Tesis Doctoral

Garantías cuantitativas para espacios de estados no tratables

Pavese, Esteban

2015-10-19

Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Garantías cuantitativas para espacios de estados no tratables

Tesis presentada para optar al título de Doctor de la Universidad de Buenos Aires
en el área Ciencias de la Computación

Esteban Pavese

Director de tesis:	Dr. Víctor Braberman
Consejero de estudios:	Dr. Sebastián Uchitel

Buenos Aires, 19 de octubre de 2015

Garantías cuantitativas para espacios de estados no tratables

Resumen: Los lenguajes basados en máquinas de estados finitos (también llamados automátas finitos) son usados de manera ubicua para la especificación de sistemas de software. La formalidad de estos modelos permite la aplicación de técnicas de validación tales como el *model checking*. De esta manera, pueden responder con seguridad si un sistema cumple o no las propiedades de interés. Al mismo tiempo, estás máquinas pueden ser utilizadas de manera *composicional*, especificando comportamientos aislados mediante varias máquinas, y estableciendo el comportamiento global mediante su composición en paralelo. Este enfoque reduce el esfuerzo de validación, ya que la validez de las propiedades en el sistema deberían ser dependientes de la validez de las propiedades en cada componente. Sin embargo, este enfoque es amenazado por la complejidad del sistema especificado, dando lugar al problema de la explosión de estados, que puede impedir la aplicación de estas técnicas.

En esta tesis presentamos un enfoque que intenta paliar este problema, proveyendo información *cuantitativa* respecto de la propiedad que se intentó validar sin éxito. Nuestro enfoque se sostiene sobre dos contribuciones distintas, donde cada una de ellas puede, además, ser aplicada en el contexto de problemas relacionados. Esta tesis se inspira en el *modelado y model checking probabilísticos*, que pueden proveer información cuantitativa respecto de la validez de una propiedad. Esta cuantificación nos sirve de validación parcial en el contexto del problema que nos interesa.

Sin embargo, un enfoque composicional tiene sus propios problemas en un contexto probabilístico. Las anotaciones probabilísticas asociadas a eventos independientes precisan ser contrastadas con estimaciones obtenidas de la observación del comportamiento a modelar. Al agregar estas anotaciones, es preciso distinguir las fuentes de estas probabilidades; en otras palabras, las probabilidades de eventos independientes deberían estar asociadas al comportamiento de los componentes que generan este comportamiento. A su vez, es preciso mantener la relación entre la validez de los componentes de manera aislada, y la validez de los comportamientos en el sistema compuesto. Los formalismos disponibles al momento, sin embargo, no proveen la seguridad de que estos resultados de validez sean preservados durante la composición. La primera contribución de esta tesis es, entonces, una extensión probabilística al formalismo de *Interface Automata*. Esta extensión asegura la preservación de comportamiento tal como es observado por la lógica probabilística pCTL.

La segunda parte de esta tesis apunta al análisis de estos modelos, en particular cuando un análisis exhaustivo no es factible, teniendo en cuenta que la complejidad del *model checking* probabilístico es aún mayor que en el caso clásico. Nuestra hipótesis en este trabajo es que una exploración parcial, pero sistemáticamente controlada, puede proveer cotas a los valores de interés con un costo computacional reducido. Los experimentos realizados sugieren que un análisis mediante este enfoque puede ser más efectivo que tanto el *model checking* exhaustivo como así también enfoques estadísticos relacionados.

Palabras clave: modelado probabilístico, verificación probabilística, simulaciones, verificación estadística, exploración parcial.

Quantitative Guarantees for Intractable State Spaces

Abstract: System specifications have long been expressed through automata-based languages, which enable automated validation techniques such as model checking. Automata-based validation has been extensively used in the analysis of systems, where they have been able to provide yes/no answers to queries regarding their temporal properties. Additionally, a compositional approach to construction of software specifications reduces the specification effort, allowing the engineer to focus on specifying individual component behaviour; and then analyse the composite system behaviour. This also reduces the validation effort, since the validity of the composite specification should be dependent on the validity of the components. However, even in a compositional approach, automatic validation techniques usually cannot cope with systems under analysis that grow complex enough. Problems such as state space explosion seriously hamper the applicability of these approaches.

In this thesis, we present an approach that can help cope with these absence of results by providing *quantitative* validation information related to the property being validated, even when the model checking approach is unable to handle the whole system. Our proposed technique stands on two different approaches, with each of them being applicable on its own to related problems. The inspiration is that *probabilistic* modelling and checking can provide quantitative information, which can in turn serve as partial validation when full checking is infeasible.

Compositional construction, however, poses additional challenges in a probabilistic setting. Numerical annotations of probabilistically independent events must be contrasted against estimations or measurements, taking care of not compounding this quantification with exogenous factors, in particular other system components' behaviour. The validity of compositionally constructed specifications requires that the validated probabilistic behaviour of each component continues to be preserved in the composite system. However, existing probabilistic automata-based formalisms do not support behaviour preservation of non-deterministic and probabilistic components over their composition. The first contribution of this thesis is a probabilistic extension to Interface Automata which preserves pCTL properties. This extension not only supports probabilistic behaviour but also allows for weaker prerequisites to interfacing composition, allowing for specification refinement of internal behaviour.

The second part of our approach aims at analysing these probabilistically enriched models, obtaining quantitative information that can be related to the validity of the property under analysis, even when a complete analysis is infeasible. Computational complexity of estimating these metrics can be prohibitive, even more so than classic model checking. Our hypothesis is that a (carefully crafted) partial systematic exploration of a system model can provide better bounds for these quantitative model metrics at lower computation cost than exhaustive exploration. Our technique combines simulation, invariant inference and probabilistic model checking to produce a probabilistically relevant portion of the model, which is then exhaustively analysed. We report on experiments that suggest that metric estimation using this technique (for both fully probabilistic models and those exhibiting non-determinism) can be more effective than (full model) probabilistic and statistical model checking.

Keywords: probabilistic modelling, probabilistic validation, model simulation, statistical methods, partial explorations.

Agradecimientos

Este escrito representa la culminación de varios años de trabajo durante los que, naturalmente, pasé por momentos muy divertidos y otros no tanto. No se equivocan los que dicen que una tesis de doctorado supone una montaña rusa emocional. En ese sentido, lo más importante a reconocer en este momento es que esta tesis no hubiese sido posible de no haber mediado muchísima gente, a quienes quiero agradecer en estos párrafos (y me disculpo de antemano si en el fragor de la escritura omito a alguien).

En primer lugar, quiero agradecer a mis directores Víctor Braberman y Sebastián Uchitel. Su guía fue invaluable dede el punto de vista académico, entusiasmándose aún más que yo cuando seguimos la buena senda, y abriendo camino cuando tocaba el turno de los obstáculos. Pero más allá de su excelencia académica, también son dos excelentes personas con las que compartimos muchos momentos dentro y fuera de la Universidad. Muchísimas gracias por todo lo que empujaron para que este trabajo saliese adelante.

Quiero agradecer también al jurado que se tomó el trabajo de leer y aportar su valioso punto de vista sobre esta tesis: Pedro D'Argenio, Holger Hermanns y Ernesto Kofman. A Holger y Pedro agradezco además por las discusiones en Saarbrücken que terminaron de redondear la primer parte de esta tesis.

Sin duda, estos años de trabajo habrían sido realmente insoportables de no ser por la presencia continuada de todos los que conforman LaFHIS. Ante todo, la *primera línea depresiva* que conformamos con Nicolás D'Ippolito, Hernán Czemerinski y Fernando Asteasuain, con la que dejamos una pesada herencia a los que siguen. A todo el resto del grupo, los que ya se fueron y los que aún están: Guido de Caso, Germán Sibay, Diego Garbervetsky, Nico Kicillof, Sergio Yovine, Hernán Melgratti, Gervasio *7K* Pérez, Guido Chari, Rodrigo Castaño, Daniel Ciolek, Ezequiel *Bishōnen* Castellano, Mariano Cerrutti, Christian Roldán, Leandro *Turco* Nahabedian, Natalia Rodríguez, Fernán Martinelli, Edgardo Zoppi, Fer Chapa, Sven Stork y al miembro honorario Ivo Krka. Mención especial Vero Rodríguez y Nati Derrosi que siempre simplificaron todo trámite, y a Dani Bonomo que además de lo anterior se convirtió en una gran amiga.

Más allá de LaFHIS, es un gran privilegio para mí ser parte del Departamento de Computación de la FCEyN UBA. No temo equivocarme al decir que todos los que en algún momento lo conformamos tenemos un gran cariño por lo que el Departamento significa.

La familia (¡extendida!) y amigos fue el otro gran soporte que puso el hombro para permitirme llegar acá. Agradezco infinitamente a mis viejos Rolando y Mariela, a mis

hermanos Pato, Julián y Damián, sus novias/esposas y a las peques Valen y Agus; a Cecilia Soria, Pablo Bruzzoni y Susi; al *clan Impávido* Willy, Moni, Sebas, Fede y Nico; a la *primada* (y tíos) en pleno (¡son muchos para nombrar!); Laura Paulin, Ana F., Marito, Dani T., Carlitos, Tom, Maty. . . y a todos aquellos que seguramente me olvido.

And last but not least, a Caro, Carito, *Carolinchen*, mi compañera de vida desde hace más de diez años, en las buenas, en las malas, y también en esta nueva etapa que se nos presenta. *Mein Schatzi, bei mir bist du sehr schön!* :)

Contents

Resumen en castellano

A continuación presentamos un resumen de esta tesis en castellano, dado que la totalidad de la misma se encuentra escrita en inglés. Aquí se resumen las ideas centrales presentadas en cada capítulo.

Capítulo 1: Introducción

En los últimos años, los sistemas de software se han vuelto ubicuos, además de encontrar aplicaciones donde una falla puede resultar crítica y causar pérdidas materiales, económicas e incluso humanas. Además, los sistemas de software han evolucionado más allá de ser meros procesadores de datos en bloque. Por lo contrario, el software se diseña, cada vez con más frecuencia, con el fin de monitorear su ambiente y responder frente a cambios del mismo. Como resultado de estas interacciones, el comportamiento del software es cada vez más complejo, dejando a su vez mayor lugar para la aparición de fallas. Así, cada vez son más deseables herramientas que permitan aseverar que un sistema de software realizará su tarea con alto grado de confiabilidad. Más aún, estas herramientas necesitan alto grado de automatización, ya que los conocimientos específicos para tales análisis no suelen ser parte de los conocimientos de los usuarios o analistas en general.

El foco de esta tesis está en estos análisis, especialmente aquellos que pueden realizar sus evaluaciones de manera temprana sobre el software, o sobre modelos lo suficientemente detallados de dicho software. Cuando analizamos estas descripciones detalladas del software, nos interesa además evaluar la validez de propiedades que son, por lo general, *temporales*, es decir que pueden predicar acerca del orden de los eventos de interés en el tiempo. Por ejemplo, si nos ocupáramos de estudiar el controlador de los sistemas de un auto de última generación, algunas preguntas válidas podrían ser *¿es cierto que al presionar el pedal de freno, los frenos en sí son accionados en a lo sumo 800 ms.?*; o bien *¿es cierto que la inyección de combustible se interrumpe siempre que el motor excede las 8000 revoluciones?* Técnicas tales como el *model checking* permiten obtener respuestas a este tipo de preguntas.

Sin embargo, un problema que amenaza estas técnicas es que rápidamente se vuelven inaplicables a medida que la complejidad del sistema bajo análisis aumenta. Esta complejidad aumenta de manera exponencial respecto del tamaño de los componentes del sistema que se desea analizar, ya que el comportamiento conjunto de estos componentes excede largamente la complejidad del comportamiento aislado de los mismos. Esto es lo que se conoce como el problema de la *explosión de estados*.

Si bien existen muchos antecedentes al respecto, que han derivado en técnicas que

buscan paliar este problema de explosión, la realidad es que es fácil, en la práctica, encontrar sistemas que rápidamente vuelven imposible este tipo de análisis. En este tipo de casos, lamentablemente, no podemos esperar mucho de las técnicas de model checking tradicionales. Estas técnicas sólo son capaces de responder si la propiedad es válida o no: si no lo es, pueden proveer un contraejemplo, mientras que si la propiedad es válida, sólo puede asegurarse mediante la exploración exhaustiva del sistema. De esta manera, si el procedimiento de exploración es interrumpido de manera temprana sin haber encontrado un contraejemplo, nada puede decirse al respecto de la propiedad.

Sin embargo, llegado este punto, ya se ha invertido mucho tiempo y trabajo. Más detalladamente, llegado este punto necesariamente se debió haber modelado el sistema de software de manera acorde, las propiedades fueron expresadas en lógicas apropiadas, y el procedimiento de model checking fue desarrollado o puesto a punto para el análisis en cuestión. No sólo esto, sino que seguramente será también el caso de que el procedimiento de model checking fue llevado a cabo parcialmente, invirtiendo una cantidad sustancial de tiempo y recursos computacionales. Sin embargo, parecería que se debe tirar todo por la borda.

Esta tesis parte de este escenario. La pregunta a responder por la tesis es *analizar si es posible, en los casos en que el model checking es incapaz de analizar ciertos modelos y propiedades en tiempo y forma, obtener, de todas maneras, algún tipo de información que sea realmente útil para el usuario que puso en marcha el proceso de verificación.*

Información cualitativa vs. información cuantitativa

Proveer una respuesta a la pregunta anterior requiere que nos movamos fuera de la clase de respuestas *cualitativas* (es decir, sí o no), y que nos fundamentemos en respuestas *cuantitativas*, es decir, que puedan proveer alguna dimensión respecto de la validez de las propiedades. Por ejemplo, podemos preguntarnos (y responder) cuestiones tales como *¿qué porcentaje del sistema se encuentra libre de fallas?* o bien *¿qué tanta confianza podemos tener en que una ejecución arbitraria no resultará en un error?*. En este sentido, podemos resumir la primera contribución de la tesis

> Esta tesis presenta un enfoque que permite obtener información cuantitativa acerca de un modelo cuya exploración completa es imposible. Más aún, esta información cuantitativa es relevante respecto de las propiedades de interés.

En este sentido, la información cuantitativa que sólo se limita a cuestiones topológicas del modelo tales como su cantidad de estados o transiciones, no son interesantes ya que no son aplicables o extrapolables a las propiedades. De alguna forma está en el medio la idea de que hay estados del sistema que son más *interesantes* que otros. Por ejemplo, si deseamos verificar el controlador de un automóvil, claramente los momentos en los que el auto se encuentra en marcha son más interesantes y críticos que aquellos en los que está detenido. Sin embargo, se requiere alguna medida acerca de este nivel de interés de los estados. En esta tesis, argumentamos que un estado es más interesante que otro si el estado es observado más frecuentemente. Esto dependerá tanto del sistema mismo como también del ambiente con el que interactúe. En esta tesis, para expresar estas dimensiones, utilizaremos teoría de la probabilidad, que nos permitirá rápidamente comparar el nivel de interés entre distintos estados a partir de comparar cuál es más probable observar durante una ejecución arbitraria del sistema.

Existen sin embargo varios problemas a la hora de intentar representar de manera coherente la componente probabilística del comportamiento de un sistema o su ambiente. Un modelado probabilístico puede hacer surgir problemas tales como

- semántica probabilística poco clara, donde es difícil descomponer la carga probabilística correspondiente a cada componente del sistema o ambiente;
- una relación poco clara entre las distribuciones probabilísticas de los componentes, y la distribución resultante en el modelo compuesto, lo cual lleva a consecuencias tales como
- una falla total en preservar comportamiento validado individualmente a través de las sucesivas composiciones de los componentes entre sí. Esto juega directamente en contra de un enfoque composicional a la hora de realizar validación y verificación del comportamiento del sistema.

Esto nos lleva a una segunda contribución de esta tesis.

> En esta tesis, presentamos un nuevo formalismo de modelado probabilístico, que permite la construcción y validación composicional e incremental de sistemas.

Análisis y verificación parcial

La introducción de este nuevo formalismo de modelado probabilístico es, sin embargo, solamente la mitad del trabajo. Si bien este nuevo formalismo permite un modelado composicional y provee una manera de introducir probabilidades de manera natural, no reduce en nada el problema de la explosión de estados. Nuestra propuesta al respecto es la introducción de variables aleatorias asociadas a una exploración parcial del sistema. Estas variables aleatorias, que serán medibles de manera eficiente, deberán guardar una estrecha relación con la validez de la propiedad que se está analizando. De esta manera, resultará que medir el valor esperado de esta variable aleatoria será equivalente a proveer cierta medida respecto de la validez de la propiedad en general. Más en particular, esta variable aleatoria buscará medir la relación entre los estados que sí han sido visitados durante la exploración parcial, y aquellos que no, que a los efectos prácticos consideraremos que violan la propiedad en su totalidad.

De esta forma arribamos a una nueva contribución de esta tesis.

> Presentamos una formalización de lo que significa realizar una verificación o validación de manera parcial, a través de formalizar qué significa, respecto del modelo completo, una exploración parcial del mismo. Además, explicamos cuál es la relación entre los resultados obtenidos de tales verificaciones parciales y los resultados que podrían obtenerse (idealmente) de una verificación total.

Sin embargo, debemos encontrar una técnica que, además de proveer estos resultados, sea técnicamente aplicable. En primer lugar, introducir las probabilidades de manera directa sobre el espacio de estados parcialmente explorado es no sólo inviable (ya que requeriría memoria adicional sobre una ya supuestamente agotada), sino que además atenta desde el punto de vista ingenieril, ya que se pierde la idea de que las probabilidades deben estar aisladas a cada componente y ser introducidas de manera composicional. Por otra parte, aún si estas preguntas pudiesen ser ignoradas, es de

suponer que los resultados obtenidos de una exploración parcial arbitraria no serán de gran utilidad. La causa de esto es que los model checkers no están realmente diseñados para trabajar de forma parcial, y por lo tanto no tienen gran cuidado en cómo realizan la exploración. Dado que necesitan la exploración en su totalidad, en general es lo mismo si realizan la exploración de manera profunda, al azar o bien con cualquier otra estrategia.

Este análisis provoca preguntas tales como

- ¿Existirá una manera de obtener sistemáticamente diferentes exploraciones parciales? ¿Podemos dar una medida de comparación entre estas distintas exploraciones parciales?
- ¿Podemos dar una medida de comparación, además, entre los resultados cuantitativos obtenidos de distintas exploraciones parciales?
- ¿Hay exploraciones que resulten sistemáticamente en mejores resultados? ¿Y qué significa que un resultado sea *mejor* que otro, en primer lugar?
- ¿Habrá alguna manera de predecir si una exploración parcial producirá mejores resultados que otra?
- Y si es así, ¿existirá una manera de construir exploraciones parciales de manera consistente, y de forma tal que los resultados obstenidos de las mismas sean consistentemente buenos?

En esta tesis presentaremos una técnica y heurísticas qeu permiten responder a las preguntas anteriores. Este enfoque combina los conceptos de simulación probabilística y estadística, inferencia de invariantes de comportamiento y verificación de modelos. Esto resume la última contribución de esta tesis.

> Presentamos una técnica automática para la exploración parcial de modelos que, mediante otro tipo de técnicas, no pueden ser explorados o verificados de manera exhaustiva. A través de esta técnica automática, además, tenemos una forma de obtener modelos parciales que, de manera consistente, apuntan a maximizar la información cuantitativa que puede extraerse de los mismos para una propiedad de interés dada.

Además, validamos estas aseveraciones mediante el uso de un conjunto de casos de estudio extraídos de la literatura relacionada con nuestro enfoque y con la verificación de software en general.

Contribuciones

Las contribuciones de esta tesis pueden resumirse como sigue

- Visto desde un punto de vista general, esta tesis provee un enfoque que permite obtener información cuantitativa respecto de propiedades cualitativas de un modelo de software. Esta información cuantitativa es de especial interés en aquellos casos en que la propiedad no puede ser verificada por técnicas al nivel del estado del arte, tales como *model checking* o verificación estadística al estilo Monte Carlo. Más aún, la información cuantitativa obtenida por nuestro enfoque está relacionada con, y es directamente interpretable en el contexto de, la propiedad que se intentó analizar en primer lugar.

- Presentamos los *Autómatas de Interfaz Probabilísticos* (PIA, por sus siglas en inglés: Probabilistic Interface Automata), con el fin de proveer un formalismo que permita el modelado de información probabilística asociada al comportamiento de un sistema de software. Los modelos PIA permiten la especificación incremental y composicional de modelos de software.

- Realizamos además una formalización del problema de verificar parcialmente un espacio de estados. Además, establecemos formalmente cuál es la relación entre los resultados obtenidos por la verificación de un espacio de estados parcial con respecto a la verificación completa de este mismo espacio de estados. Como resultado, mostramos que la verificación de espacios parciales resulta en cotas a los resultados que serían obtenidos por medio de una verificación total.

- Finalmente, presentamos un procedimiento automático que permite obtener espacios de estados parciales que, de manera consistente, proveen cotas que resultan mejores que los resultados obtenidos (dados el mismo tiempo y memoria disponibles) mediante técnicas establecidas como la verificación total del espacio de estados, o enfoques del estilo Monte Carlo.

Capítulo 2: Antecedentes y preliminares

En este capítulo se presentan conceptos sobre los cuales se construyen los resultados presentados en esta tesis. Aquí se presentan en primer lugar definiciones relativas a la teoría de la medida y de la probabilidad, y de manera seguida se introducen algunos de los formalismos de modelado de sistemas de software. Estos formalismos son aquellos en los que este trabajo se fundamenta.

Nociones de teoría de la probabilidad

Respecto de los conceptos asociados a la teoría de la medida y probabilidad, se presentan las siguientes definiciones.

Un **espacio de probabilidad** (Definición 2.1) [Fel08] está dado por la tripla $< \Omega, 2^{\Omega}, \mu >$, donde

- Ω es un conjunto llamado *espacio de eventos*;
- 2^{Ω} es el conjunto de partes de Ω, siendo sus elementos los *eventos* de interés; y
- $\mu : 2^{\Omega} \to [0, 1]$ es una función tal que
  - $\mu(\emptyset) = 0$;
  - $ii)$ $\mu(\Omega) = 1$; y
  - dada una secuencia de elementos de 2^{Ω} expresada por $(\omega_i), i \in \mathbb{N}$ y siendo que estos elementos son disjuntos de a pares, vale que $\mu(\bigcup_i \omega_i) = \sum_i \mu(\omega_i)$.

La función μ suele llamarse una función de medida de probabilidad, o más frecuentemente, una *distribución*. Dado un subconjunto ω del espacio de eventos Ω, $\mu(\omega)$ se dice la *medida* de ω.

Las nociones de espacio de probabilidades y sus distribuciones dan lugar a definiciones tales como el **producto de espacios de probabilidades** (Definición 2.3), **variables aleatorias** (Definición 2.4) y el concepto de **valor esperado** (o *esperanza*) de una variable aleatoria (Definición 2.5) [Fel08].

Nociones acerca de formalismos de modelado

A continuación, se presentan los formalismos de modelado de sistemas de software sobre los cuales se sostiene esta tesis. En primer lugar, se presentan formalismos que permiten sólo el modelado de sistemas de software puramente no-determinísticos, es decir, que no permiten la expresión de medidas probabilísticas en el comportamiento del sistema de software que se intenta modelar. Estos formalismos que se presentan en esta sección comparten la particularidad de que todos ellos son *máquinas de estados finitos*.

En primer lugar se introducen los **Sistemas Etiquetados de Transición** (LTS, por sus iniciales en inglés, *Labelled Transition Systems*) como lenguaje específico para máquinas de estados finitos. Estos LTSs (Definición 2.6) [BK08] se caracterizan mediante una tupla $< S, S_0, A, R >$ donde

- S es un conjunto finito de estados;
- S_0 es un estado distinguido de S que denominaremos *estado inicial*;
- A es un conjunto finito de etiquetas; y finalmente
- R una relación de transición tal que $R \subseteq S \times A \times S$. Esta relación de transición especifica, para un estado dado en S, los estados a los cuales puede evolucionar mediante la aplicación de alguna etiqueta en el conjunto A.

Seguidamente, a fin de permitir que las relaciones de interfaz entre componentes sean modeladas de manera explícita, se introduce la idea de segregación de las acciones del conjunto A en tres subconjuntos. Estos subconjuntos representan, respectivamente, las acciones que un componente emite (llamadas acciones de *salida* o *output*); aquellas que espera recibir (acciones de *entrada* o *input*); y finalmente aquellas acciones que toma de manera interna sin ningún tipo de interacción con su entorno (acciones *internas* u emphocultas).

Esta segregación de acciones permite explicitar relaciones entre las acciones de un componente y su entorno—que, a su vez, está definido en base a otros componentes modelados mediante el mismo formalismo. En particular, se pone de plano de forma explícita que una acción de entrada de un componente sólo sincronizará con una acción del mismo nombre, y que sea declarada como acción de salida de otro componente. De manera recíproca, una acción de salida sincronizará con una acción de otro componente si coinciden en nombre, y además el segundo componente declara que esta acción es de entrada en su contexto. Finalmente, las acciones internas no sincronizan con ninguna otra acción y pueden dispararse en cualquier momento.

Estas nociones dan lugar al concepto de los **Autómatas de Interfaz** (Definición 2.7) [HdA01]. Un Autómata de Interfaz es un LTS cuyas acciones han sido segregadas de la manera explicada anteriormente. Más formalmente, se trata de una tupla $P =< S_P, s_P^0, A_P^I, A_P^O, A_P^H, R_P >$ donde:

- S_P es un conjunto finito de estados;
- $s_P^0 \in S_P$ es un estado distinguido al que denominamos *inicial*;
- A_P^I, A_P^O, A_P^H son los conjuntos de acciones de entrada, salida y acciones ocultas, respectivamente. Asimismo, nos referimos al conjunto de todas las acciones como $A_P = A_P^I \cup A_P^O \cup A_P^H$; y finalmente
- $R_P \subseteq S_P \times A_P \times S_P$ es la relación de transición, que se comporta de manera similar a como fue definido en el caso de LTS.

Semántica de trazas de los Autómatas de Interfaz

Existen distintas formas de definir las semánticas de estas máquinas de estado. Cada manera de definir esta semántica tiene distintos grados de granularidad. La manera más simple de definir la semántica es por medio de sus **fragmentos de ejecución** (Definición 2.8), más particularmente sus **ejecuciones**, que son aquellos fragmentos de ejecución que tienen su comienzo en el estado inicial. Un fragmento de ejecución de un autómata A es una secuencia (posiblemente infinita) $\alpha = s_0 a_1 s_1 a_2 s_2 \ldots$, donde se alternan estados y acciones de A. Estos fragmentos comienzan *siempre* con un estado y, si son finitos, finalizan también con un estado. Finalmente, debe darse que cada subsecuencia $s_i a_{i+1} s_{i+1}$ dentro de un fragmento de ejecución se corresponde con una de las transiciones definidas $(s_i, a_{i+1}, s_{i+1}) \in R_P$.

Notamos $execs(A)$ al conjunto de posibles ejecuciones de un autómata A. Es importante notar que, como en un estado dado varias acciones pueden estar habilitadas simultáneamente, existirán multiples (posiblemente infinitas) ejecuciones, dependiendo de qué acción sea elegida en cada uno de estos momentos. La noción de **planificador de ejecución** (Definición 2.13), generalmente notado *scheduler*, formaliza este mecanismo de decisión y resolución de no determinismo. Esencialmente, un planificador es una función que, dada una ejecución finita (y que por tanto tiene un estado final), decide qué acción se tomará a continuación. Distintos planificadores, es decir, distintas funciones de elección de acciones, resultarán en distintas ejecuciones.

Mediante esta noción de planificador podemos refinar el conjunto de ejecuciones de un autómata, y limitarlos a un planificador determinado. De esta manera, dado un planificador σ podemos referirnos al **conjunto de ejecuciones generado** por σ como el subconjunto $execs(A, \sigma) \subseteq execs(A)$ tal que todas sus ejecuciones respetan las selecciones realizadas por σ a cada paso. Es importante notar que un planificador elimina todo el no determinismo de A, y por lo tanto elimina su comportamiento ramificado. Dicho de otra manera, para un autómata como los estudiados hasta este momento, cada planificador induce una sola ejecución posible. Veremos más adelante que al introducir el comportamiento probabilístico esto varía.

Son de especial de interés aquellos planificadores que no son desbalanceados en sus elecciones, es decir que, cuando encuentran repetidamente el mismo estado dentro de una misma ejecución, balancean sus elecciones entre las distintas acciones disponibles, sin hacer que predomine una de ellas fuera de lo normal. Estos planificadores son denominados **planificadores fuertemente ecuánimes** (Definición 2.16) [CGP99]. Más formalmente, un planificador es *fuertemente ecuánime* si las ejecuciones que genera son ecuánimes.

A su turno, una ejecución α se dice ecuánime en base a su *conjunto de recorridos*. Para cada $s \in S_P$, definimos $Recorridos(\alpha, s) = \{i \in \mathbb{N}_0 \cdot \alpha_i^s = s\}$, es decir, $Recorridos(\alpha, s)$ denota los índices en α en los que el estado s es recorrido. Análogamente se pueden definir los recorridos en base a las transiciones, es decir, $Recorridos(\alpha, (s, a, s'))$ es el conjunto de índices en α donde la transición (s, a, s') es ejecutada. En base a estas definiciones, decimos que la ejecución α es ecuánime si para cada $s \in S_P$ tal que $Recorridos(\alpha, s)$ es un conjunto *infinito*, es cierto tambien que, cada vez que (s, a, s') está habilitada en s, entonces el conjunto $Recorridos(\alpha, (s, a, s'))$ también es infinito. Las ejecuciones finitas son entonces, trivialmente, siempre ecuánimes.

Como se discutió anteriormente, una de las principales ventajas que acarrea el uso de este tipo de formalismos basados en autómatas sincronizantes es que permiten la especificación modular de sistemas de software. Es decir, estos formalismos permiten la posibilidad de especificar el comportamiento de cada componente del

sistema de software de manera aislada, para luego obtener la especificación del comportamiento global a partir de la composición en paralelo de las especificaciones de los componentes.

En el caso de los Autómatas de Interfaz, la composición en paralelo, también denominado el **producto** se define a partir de las acciones que se encuentran definidas para cada estado, y teniendo especial cuidado en la segregación de acciones ya definida (Definición 2.10) [HdA01]. Dados P y Q dos Autómatas de Interfaz, su producto es un nuevo Autómata de Interfaz $P \otimes Q$ tal que

- su conjunto de estados $S_{P\otimes Q}$ viene dado por el producto cartesiano $S_P \times S_Q$;
- su estado inicial es el producto cartesiano de los estados iniciales de P y Q; esto es $s^0_{P\otimes Q} = (s^0_P, s^0_Q)$; y finalmente
- sus conjuntos de acciones de entrada, salida y ocultas vienen dados por
  - $A^I_{P\otimes Q} = (A^I_P \cup A^I_Q) \setminus Shared(P, Q)$;
  - $A^O_{P\otimes Q} = (A^O_P \cup A^O_Q) \setminus Shared(P, Q)$; y
  - $A^H_{P\otimes Q} = A^H_P \cup A^H_Q \cup Shared(P, Q)$

Por otra parte, la relación de transición $R_{P\otimes Q}$ se define mediante el conjunto de relaciones

$$\left\{ \begin{array}{l} \{((s,t), a, (s',t)) \text{ such that } (s,a,s') \in R_P \wedge \\ \quad t \in S_Q \wedge a \notin Shared(P,Q)\} \cup \\ \{((s,t), a, (s,t')) \text{ such that } (t,a,t') \in R_Q \wedge \\ \quad s \in S_P \wedge a \notin Shared(P,Q)\} \cup \\ \{((s,t), a, (s',t')) \text{ such that } a \in Shared(P,Q) \wedge \\ \quad (s,a,s') \in R_P \wedge (t,a,t') \in R_Q\} \end{array} \right\}$$

donde $Shared(P, Q)$ es el conjunto de acciones que ambos autómatas P y Q comparten es decir, $Shared(P, Q) = A_P \cap A_Q$.

Es importante notar en este punto que los Autómatas de Interfaz introducen además la noción de **componibilidad** (Definición 2.9) [HdA01], que establece condiciones para que el producto de dos Autómatas de Interfaz P y Q tenga sentido. Esencialmente, esta definición establece que la componibilidad viene dada por la compatibilidad de sus conjuntos de acciones segregadas, es decir

- $A^H_P \cap A_Q = \emptyset$;
- $A_P \cap A^H_Q = \emptyset$;
- $A^I_P \cap A^I_Q = \emptyset$; y
- $A^O_P \cap A^O_Q = \emptyset$

La contribución particular que define a los Autómatas de Interfaz frente a formalismos similares basados en máquinas de estados finitos es la noción de **estados ilegales** (Definición 2.11) [HdA01] que pueden surgir al momento de la composición o producto. Informalmente, un estado es ilegal si viola la noción de interfaz entre los autómatas que están siendo compuestos. La interfaz es violada cada vez que una de las máquinas tiene la intención de ejecutar una de sus acciones de salida, pero sin embargo la máquina receptora no está lista a aceptar esa acción como entrada. Más formalmente, el estado (s, q) de la composición entre P y Q es *ilegal* si $\exists a \in Shared(P, Q)$ tal que $a \in A^O_P(s) \wedge a \notin A^I_Q(q)$ o, de manera simétrica

$\exists a \in Shared(P,Q)$ tal que $a \notin A_P^I(s) \wedge a \in A_Q^O(q)$. Notamos al conjunto completo de estados ilegales de la composición como $Illegal(P,Q)$.

Esta noción de estados ilegales da lugar, finalmente, a la noción de **ambiente legal** (Definición 2.12) [HdA01] para un Autómata de Interfaz P. Formalmente, un Autómata de Interfaz Q es un *ambiente legal* para otro Autómata de Interfaz P cada vez que simultáneamente se cumplen

- P and Q son componibles;
- $A_Q^I = A_P^O$; y
- ninguno de los estados en $Illegal(P,Q)$ es alcanzable en $P \otimes Q$.

Semántica de ramificación de los Autómatas de Interfaz

Es sabido que la semántica de trazas es demasiado gruesa para el modelado de sistemas de software [CGP99], siendo preferida una semántica que permita distinguir el comportamiento ramificado que se deriva de las decisiones no determinísticas de los planificadores.

Existen diversas lógicas modales que permiten evaluar este comportamiento ramificado. En esta tesis trabajamos con variaciones de la lógica CTL (*Computational Tree Logic*) [EC82]. En particular, nos resulta útil la lógica ACTL [DV90], la cual es equivalente a la lógica CTL. La principal diferencia entre ambas es que la lógica CTL se expresa a través de predicados sobre los *estados* del objeto de estudio, mientras que ACTL tiene su foco en las *acciones*. Esta particularidad es útil en nuestro contexto, ya que nos permite expresar de manera directa las restricciones relacionadas con la disponibilidad de acciones para ser sincronizadas.

Formalismos de modelado probabilístico

Este trabajo tiene un fuerte foco en la introducción de un formalismo de modelado que permite la expresión de comportamiento probabilístico dentro de un componente de software. En su manera más básica, los modelos probabilísticos que introducimos son LTSs donde la relación de transición tiene una componente que puede gobernar la decisión entre distintas elecciones posibles por medio de una distribución probabilística. Con este objetivo, nos fundamentamos en un formalismo ampliamente conocido, el de **Autómatas Probabilísticos Simples de Segala** (Definición 2.19) [SL95, Seg95] (SPAs por sus siglas en inglés *Simple Probabilistic Automata*). Como se notó con anterioridad, la principal diferencia de estos autómatas es que su relación de transición está determinada por distintas distribuciones probabilísticas. Más formalmente, un SPA es una tupla $M =< S_M, s_M^0, A_M, R_M >$ donde

- S_M es un conjunto finito de estados.
- $s_M^0 \in S_M$ es el estado inicial distinguido.
- A_M es un conjunto finito de acciones.
- $R_M \subseteq S_M \times A_M \times D(S_M)$ es la relación de transición, donde $D(S_M)$ se refiere al conjunto de posibles *distribuciones probabilísticas* sobre el espacio de eventos determinado por el conjunto de estados S_M. Como S_M se trata de un conjunto finito, resulta que las distribuciones en $D(S_M)$ son discretas.

El producto o composición en paralelo de SPAs se define de manera análoga a aquel de los Autómatas de Interfaz, con la salvedad de que, durante la sincronización de dos transiciones que son gobernadas por dos distribuciones δ_1 y δ_2, se requiere calcular además el producto de estas dos distribuciones (Definición 2.20) [SL95].

Siguiendo con los paralelismos entre ambos formalismos, también es posible definir la noción de ejecuciones de un SPA (Definición 2.21). En el caso de los SPAs, la diferencia es que las ejecuciones son secuencias $\alpha = s_0(a_1, p_1)s_1(a_2, p_2)s_2 \dots$ que también alternan estados y transiciones, pero donde estas transiciones están anotadas no sólo por su acción sino también por la probabilidad asociada al estado de destino según la distribución probabilística que la gobierna.

La noción de planificadores también está presente para determinar el conjunto de ejecuciones de un SPA (Definición 2.22). La principal diferencia es que, a diferencia de los planificadores para Autómatas de Interfaz que sólo planificaban la siguiente acción y estado, los planificadores de SPAs planifican una distribución probabilística asociada en vez de un estado único. De esta forma, un planificador no resulta en una única ejecución, sino en múltiples que dependen de la resolución de esta distribución probabilística. En particular, en vez de una única ejecución, determinan una única Cadena de Markov de Salto Discreto [Kul09] (DTMC, por sus siglas en inglés *Discrete Time Markov Chain*).

Medidas de las ejecuciones de un SPA

La combinación de un planificador σ con un SPA M define una medida probabilística δ en la σ-álgebra determinada por el conjunto de ejecuciones posibles. Este espacio de eventos dado por las ejecuciones no es finito, ni tampoco numerable, por lo que no es posible establecer una medida probabilística discreta sobre los mismos. En cambio, se hace necesario referirse a **conjuntos cilíndricos** (a veces también llamados *conos* en la literatura) de ejecuciones (Definición 2.26).

En particular, estos cilindros se definen a partir de una ejecución *finita* α de un SPA M que nos permite caracterizar sus (posiblemente infinitas) continuaciones. Dada esta ejecución finita α, el cilindro de α es el conjunto de ejecuciones $C_\alpha = \alpha' \in execs(M) \cdot \alpha \leq \alpha'$. La medida del cilindro C_α definida por un planificador σ se define como

$$\delta(C_\alpha, M, \sigma) = \prod_{i=1}^{length(\alpha)} EsPlanificada(\sigma, \alpha, i-1, \alpha_i^a) \times \delta_{plan}(\sigma, \alpha, i-1)(\alpha_i^s)$$

donde $\delta_{plan} : Sched(M) \times execs(M) \times \mathbb{N} \to D(S_M)$, y $EsPlanificada : Sched(M) \times execs(M) \times \mathbb{N} \times A_M \to (0, 1)$ son tales que $\delta_{plan}(\sigma, \alpha, n) = \sigma(\alpha_0 \dots \alpha_n)_\delta$ y

$$EsPlanificada(\sigma, \alpha, n, a) = \begin{cases} 1 & \text{si } \sigma(\alpha_0 \dots \alpha_n)_a = a \\ 0 & \text{en otro caso} \end{cases}$$

Dicho de otra manera, δ_{plan} es la distribución correspondiente a la transición que está planificada en próximo lugar, mientras que $EsPlanificada$ indica si es el caso que efectivamente la acción a es la próxima acción planificada.

Semántica de ramificación de los Autómatas Probabilísticos Simples

De manera similar a como es en el caso de los Autómatas de Interfaz, es preciso tener una semántica que preserve la estructura de ramificación de su comportamiento, con la salvedad de que en el caso de los SPAs esta ramificación viene dada tanto

por el no determinismo como también por el comportamiento inducido por las distribuciones probabilísticas de las transiciones. Así como se utilizaban las lógicas CTL y sus equivalentes para los Autómatas de Interfaz, en el caso de los SPAs utilizaremos la lógica pCTL (Definición 2.34) [HJ89], que enriquece a CTL con la posibilidad de referirse además a los valores probabilísticos de los comportamientos capturados.

La lógica pCTL y sus extensiones permiten, además, introducir la noción de **valores y estructuras de recompensa** (Definición 2.35) [QS96] para tanto estados como transiciones de una ejecución. En esta tesis en particular nos interesan en particular las recompensas para transiciones. De esta manera, es posible asociar, además de una probabilidad, un valor específico a cada ejecución del sistema modelado. Estos valores pueden codificar dimensiones del software tales como su confiabilidad, tiempos de respuesta, etc.

Dado que las ejecuciones tienen una probabilidad asociada, este valor de recompensa de la ejecución estará definido por una variable aleatoria. Siendo así, también será de interés el cálculo del valor esperado de este valor de recompensa. Formalmente, el valor de recompensa de las ejecuciones de un SPA viene dado por una estructura de recompensa, que se define a partir de una función $\rho : S \times A \times S \to \mathbb{R}_{\geq 0}$.

Entonces, dada una ejecución π de un SPA M, y una estructura de recompensas ρ sobre M, el valor de recompensa de π está dado por la suma de las recompensas de cada una de sus transiciones. Notaremos el valor de esta recompensa como $\rho(\pi)$. Una particularidad a tener en cuenta es que las estructuras de recompensa siempre asignan un valor no negativo a las transiciones. Por lo tanto, dado un prefijo π_{pref} de una ejecución π, el valor de recompensa de π_{pref} es necesariamente menor (o igual) al valor de recompensa asociado a π.

Esta posibilidad de calcular el valor esperado de la recompensa para un conjunto de ejecuciones puede ser combinada con la posiblidad de pCTL de describir conjuntos de ejecuciones asociadas a eventos de interés. En particular, podremos calcular el valor esperado de una recompensa asociada al cumplimiento de una propiedad de alcanzabilidad del sistema siendo analizado (Definición 2.36) [QS96]. Parte de nuestro trabajo se concentrará en intentar calcular cotas inferiores a estos valores de recompensa. Estas cotas son especialmente útiles en los casos en que el valor esperado real no puede calcularse debido a limitaciones causadas por el tamaño del sistema siendo analizado, o donde el tiempo necesario para el cálculo exacto es excesivo.

Capítulo 3: Autómatas Probabilísticos de Interfaz (Probabilistic Interface Automata)

En este capítulo introducimos nuestra primera contribución de esta tesis, los Autómatas Probabilísticos de Interfaz (PIA, por sus siglas en inglés *Probabilistic Interface Automata*). Este formalismo surge como combinación entre los SPAs y los Autómatas de Interfaz, a fin de resolver problemas de modelado que surgen al introducir probabilidades en el contexto de un modelado no determinístico de sistemas de software.

En primer lugar, analizamos los antecedentes al problema del modelado probabilístico. La discusión de estos antecedentes históricos se centra alrededor de un ejemplo de modelo de software al cual se desea combinar con un modelo de comportamiento probabilístico de su ambiente. En este caso, utilizamos el modelo de una máquina expendedora de café para orientar la discusión. Este modelo puede apreciarse en la Figura 1.

Esta máquina de café establece sus interacciones con el usuario por medio de una pantalla táctil. Por medio de esta pantalla, el usuario puede primero elegir su

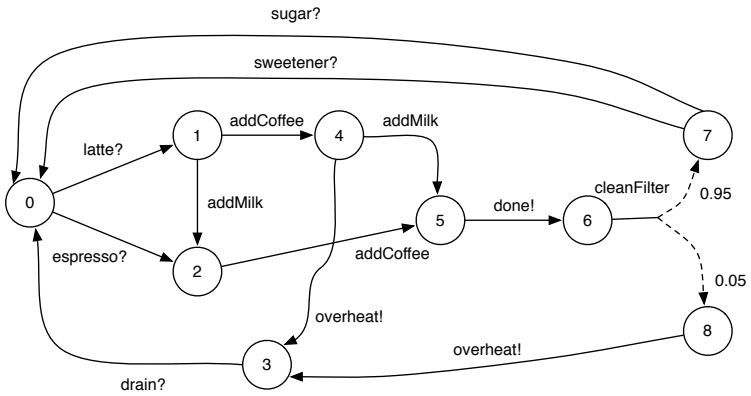


Figura 1: Máquina expendedora de café

bebida (café *espresso*, o café *latte*). Luego de que la bebida es preparada, el usuario debe seleccionar si desea azúcar o endulzante en su bebida. Finalmente, la máquina entrega la bebida al usuario. Sin embargo, la máquina tiene la particularidad de que eventualmente puede sobrecalentarse. En ese caso, es necesario que se purgue manualmente la máquina.

Con el objetivo de llevar a cabo un análisis composicional, podemos comenzar a validar este sistema aisladamente, sin depender de los otros componentes con los que interactúa. Por ejemplo, puede resultar de interés saber si es posible que la máquina se sobrecaliente *después* de que se preparó el café, ya que esta situación puede resultar peligrosa para el usuario si esa bebida se derrama y resulta en una quemadura.

Es fácil ver que esta situación puede darse; la ejecución que recorre los estados 0, 2, 5, 6, 8, 3 manifiesta este problema. No sólo eso, sino que existe siempre al menos un 0,05 de probabilidad de que esta situación se manifieste. Esta probabilidad es independiente del entorno con el cual esta máquina interactúe.

Una vez que se disponga de un modelo del comportamiento del usuario, esta probabilidad de falla podrá ser refinada. El siguiente paso del análisis es, entonces, realizar un modelo probabilístico del comportamiento del usuario. Sin embargo, notaremos que esto no es tan simple como parece, ya que algunas elecciones de modelado pueden conducir a problemas que no necesariamente resulten evidentes. Estos problemas pueden ser causados por la introducción de las probabilidades y sus interacciones, o bien por violaciones de interfaz.

Desde el punto de vista de la introducción de probabilidades, debemos tener en cuenta que históricamente se han planteado dos maneras distintas de hacerlo. Estas dos maneras han recibido los nombres de modelado *generativo* [Chr90] y *reactivo* [vGSS95].

Los modelos generativos son tales que la distribución probabilística de sus transiciones elige en cada caso tanto una acción como un estado de destino. Esto conlleva problemas a la hora de la composición en paralelo [DHK99]. En primer lugar, requieren que toda transición esté anotada probabilísticamente. Esta ausencia de transiciones no determinísticas resulta en que se está especificando, de manera solapada, la carrera entre distintas acciones para determinar cual se ejecuta antes que otra. En general este tipo de aspectos no es controlable por ninguno de los componentes, sino que depende de agentes externos como los planificadores definidos en la sección anterior. Otro problema que puede suscitarse es que son posibles las violaciones de interfaz. Una transición puede resolverse probabilísticamente en uno de los componentes determinando que cierta acción de salida debe ejecutarse. Sin embargo, el otro componente puede determinar, de manera similar, que *otra* acción será la que

deberá ser aceptada. Esta combinación no es válida, pero sin embargo se le asigna una valuación probabilística que, en definitiva, no tiene sentido desde el punto de vista del comportamiento del sistema compuesto. Estos problemas son consecuencia de la imposibilidad de los modelos generativos de modelar no determinismo, a pesar de que se vuelve aparente a la hora de la composición.

Los modelos reactivos son más apropiados para esta tarea. Estos modelos tienen transiciones donde sólo el estado destino es seleccionado probabilísticamente, como es en el caso de los SPAs introducidos anteriormente. Sin embargo, el problema de violaciones de interfaces continúa amenazando la aplicabilidad de la técnica.

El resultado de emplear cualquiera de las técnicas de modelado es que, cada uno a su manera, impide un razonamiento composicional a la hora de analizar los comportamientos temporales de los sistemas en base a sus componentes. Los comportamientos que se validan a nivel de cada componente resultan, una vez compuestos, inválidos. Esta contradicción sugiere que existe un problema al nivel del modelado de los sistemas, o al nivel de la herramienta que estamos utilizando para modelarlos. El objetivo del formalismo que introducimos en este capítulo es hacer evidentes estas fallas directamente a la hora del modelado, de manera que cualquier error de modelado sea descubierto al momento del análisis aislado de cada componente, en vez de invalidar el análisis en la última fase composicional.

Autómatas Probabilísticos de Interfaz

A partir de este análisis y la necesidad de un formalismo que evite los problemas antedichos es que proponemos como solución a los Autómatas Probabilísticos de Interfaz (PIA, por sus siglas en inglés *Probabilistic Interface Automata*). Estos autómatas surgen de una combinación de las ideas de los Autómatas de Interfaz (IA) y los SPA ya introducidos en secciones anteriores. Como tal, puede verse (Definición 3.1) que los PIA son casos particulares de los SPA, y por lo tanto pueden utilizarse en un modelado en conjunción con los mismos. Formalmente un PIA A es una tupla $M =< S_M, s^0_M, A^I_M, A^O_M, A^H_M, R_M >$ donde los conjuntos A^I_M, A^O_M y A^H_M son mutualmente disjuntos, y de forma tal que si definimos $A_M = A^I_M \cup A^O_M \cup A^H_M$ resulta que la tupla $M_{SPA} =< S_M, s^0_M, A_M, R_M >$ es un SPA como fuera definido anteriormente.

Así como podemos establecer la relación entre un PIA y su SPA embebido, podemos hacer algo similar para obtener el **IA subyacente** (Definición 3.2), que resulta de proyectar las distribuciones que constituyen las transiciones del PIA en transiciones no determinísticas. Mediante esta proyección se vuelve natural la definición de **componibilidad de PIAs** (Definición 3.3).

De la misma manera, pero proyectando sobre el SPA subyacente, podemos definir para PIAs las nociones de ejecuciones y producto en paralelo (Definición 3.4). Este producto es muy similar al caso del producto de SPAs, tal como puede verse en el ejemplo de la Figura 2. Por otra parte, heredan de los IA las nociones de estados ilegales y ambientes legales (Definición 3.5).

En este momento es importante remarcar una notable diferencia entre las nociones de ilegalidad de IAs y aquella de PIAs. La definición de ilegalidad para el caso de IAs es muy restrictiva. Requiere que, para cada uno de los componentes involucrados en la composición, las acciones de salida que se quieran ejecutar por parte de este componente sean *inmediatamente* aceptadas como acciones de entrada por el componente contraparte. Sin embargo, esto resulta tan restrictivo que impide el modelado incremental y el refinamiento de comportamiento interno de un componente. Por ejemplo, supongamos el caso en que un componente desea solicitarle un dato a otro componente. Supongamos además que este segundo componente se

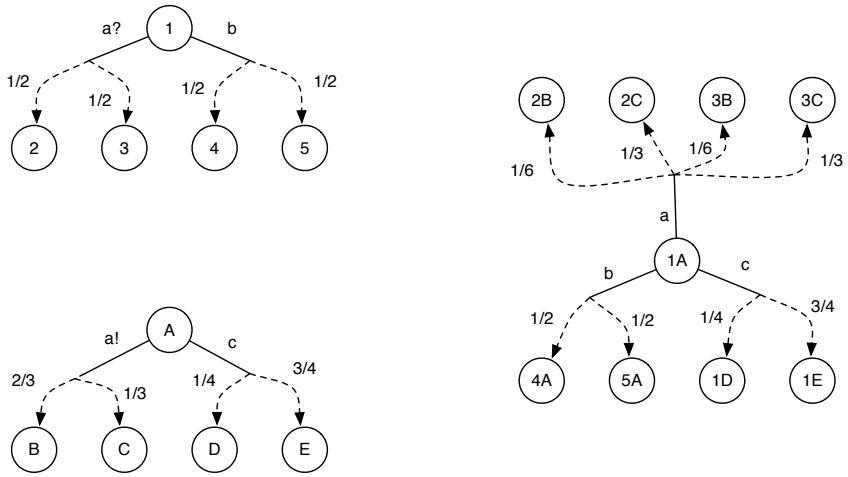


Figura 2: Producto (parcial) de Autómatas Probabilísticos de Interfaz

encuentra actualmente realizando un cómputo que lo bloquea temporariamente y, por lo tanto, no puede responder inmediatamente a este requerimiento. Este estado de la composición es, tal como está prescripto por la definición de estados ilegales de IA, ilegal. Pero supongamos además que el segundo componente, más allá de que se encuentre realizando un cómputo, es tal que *siempre* que realiza un cómputo interno eventualmente vuelve a un estado en el que acepta requerimientos, sin excepción. En ese caso, el bloqueo del requerimiento al primer componente es tan sólo *transitorio*. La eventual respuesta a ese requerimiento está *garantizada*, siendo el problema solamente que no se puede realizar de manera inmediata. Este tipo de situaciones no son permitidas por el formalismo de IA, pero sí lo son por los PIAs.

La razón detrás de esta decisión de modelado es que es usual, dentro del proceso de desarrollo de un software, que sea preciso detallar comportamiento que antes quedaba a un nivel de abstracción superior. Este proceso de refinamiento no es sólo común en el desarrollo de software, sino también deseable, ya que converge a introducir detalle en aquellas secciones del comportamiento que naturalmente van necesitando una explicación más profunda.

La Definición 3.5, entonces, no sólo permite la sincronización demorada de algunos tipos de interacciones, sino que también formaliza estas situaciones en detalle. Esta formalización se sostenie en la lógica de descripción ACTL y la noción de planificadores ecuánimes para establececer las condiciones exactas en las que la sincronización demorada está permitida.

Preservación de comportamiento probabilístico

El resultado más fuerte relacionado con el uso de PIAs para el modelado de sistemas concurrentes es el hecho de que permiten la validación composicional del comportamiento de estos sistemas. Como ya se discutió anteriormente, un enfoque de validación composicional debería permitir la validación de comportamiento aislado a nivel de cada componente en primer lugar. En segundo lugar, debería permitir la validación de estos mismos comportamientos, pero directamente sobre el modelo global resultante de la composición en paralelo de estos componentes. Finalmente, y más importante aún, ambas validaciones deben ser coherentes. Es decir, las propiedades que se validaron sobre los componentes aislados deben mantener su validez inalterada toda vez que se los compone con interfaces apropiadas.

El modelado por medio de PIA efectivamente provee esta garantía, tal como se

enuncia en el Teorema 3.1. Este teorema establece formalmente que, dados A y B dos Autómatas Probabilísticos de Interfaz tales que

- A y B son componibles; y
- su producto $A \otimes B$ es legal (es decir, no contiene estados ilegales que sean alcanzables);

entonces vale que cada vez que ϕ_A es una fórmula pCTL tal que está expresada *exclusivamente* en términos de acciones del autómata A, y A es tal que satisface ϕ_A, entonces también es cierto que $A \otimes B \models \phi_A$[1]

Este teorema tiene consecuencias inmediatas respecto de las medidas probabilísticas relacionadas con estos comportamientos. En particular, se desprende el Corolario 3.1 que establece que si la probabilidad de que el PIA A satisfaga la propiedad ϕ_A se encuentra en un intervalo $[P_{min}, P_{max}] \sqsubseteq [0, 1]$, entonces la probabilidad de que $A \otimes B$ satisfaga la misma propiedad se encuentra en el mismo intervalo (posiblemente en un intervalo estrictamente incluido en él); es decir, la propiedad probabilística se sigue verificando si ya lo hacía en primer lugar.

Capítulo 4: Validación preliminar de PIAs

En este Capítulo nos concentramos en validar tres cuestiones que nos parecen centrales a la utilidad y aplicabilidad de los Autómatas Probabilísticos de Interfaz. En primer lugar, queremos validar que modelar componentes y, por extensión, sistemas concurrentes completos por medio de PIAs no es necesariamente más complejo que hacerlo mediante formalismos ya establecidos. En segundo lugar, queremos mostrar que los modelos resultantes de aplicar el formalismo de PIA para el desarrollo de modelos de componentes concurrentes resulta en modelos legibles y no contaminados de transiciones espúreas. Finalmente, y más allá de que los resultados teóricos fuesen debidamente demostrados en el Capítulo anterior, queremos validar que efectivamente las propiedades validadas en los componentes mantienen su validez sobre la composición global.

Con el objetivo de validar estas tres cuestiones, tomamos de la literatura un ejemplo de sistema reactivo y crítico. En este caso, nos concentramos en el sistema de TeleAssistance [EGMT09], descripto de manera simplificada en la Figura 3.

El sistema de *TeleAssistance* (TA) es una aplicación web cuyo objetivo es el de proveer asistencia médica remota a pacientes que, por alguna razón, carecen de movilidad propia o precisan quedarse en sus hogares, pero que de todas maneras sufren de alguna afección tal que necesitan atención y monitoreo continuos. La interacción más básica entre el paciente y el sistema TeleAssistance comienza siempre con el envío del comando `startAssistance` hacia el sistema. Como resultado de este comando, el sistema TA entra en un ciclo reactivo en el que puede aceptar cualquiera de los siguientes pedidos de interacción:

- `stopMsg`, que indica al sistema TA que el paciente no tiene más requerimientos que realizar durante esta sesión.
- `vitalParamsMsg`. Este comando permite al paciente enviar sus parámetros vitales hacia el sistema, mediante un dispositivo dedicado a la tarea. Al recibir

[1] En rigor, el Teorema 3.1 establece algunas condiciones sobre la forma de la fórmula ϕ_A, y sobre los planificadores empleados para verificar la validez de la fórmula en A y $A \otimes B$. Sin embargo, el espíritu del teorema es tal como lo reproducimos en este resumen.

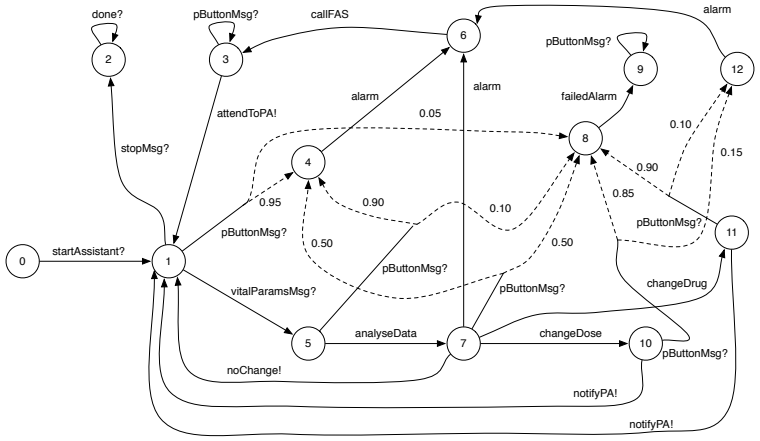


Figura 3: Un modelo del software del sistema TeleAssistance

este tipo de comando, el servidor de la aplicación se ocupa de evaluar los parámetros vitales del paciente. A continuación, y de ser necesario, el sistema sugiere una acción a tomar. Por ejemplo, el sistema puede decidir que es necesario algún cambio en la medicación que está recibiendo el paciente. De ser así, se lo comunica en forma de uno de dos comandos: `changeDrug`, que indica que el medicamento debe cambiar completamente; o bien `changeDose` que indica que, si bien el medicamento no cambiará, si lo hará la dosis a administrar. En cualquiera de estos dos casos, el dispositivo en poder del paciente es notificado, de manera que en el futuro administre los medicamentos según la nueva configuración. Además, el usuario mismo es notificado de que se realizó un cambio mediante el mensaje `notifyPA`, aunque no se le informa detalladamente la naturaleza del cambio.

- Si durante el análisis de los parámetros vitales del paciente resulta que se detecta algún tipo de anomalía que pueda amenazar la salud del paciente, el sistema eleva una alarma y solicita que un equipo de primeros auxilios (FAS, del inglés *First-Aid Squad)* sea enviado al domicilio del paciente. En este caso, el sistema indicará al paciente que espere al FAS mediante el mensaje `attendToPA`.

- También puede darse el caso que el paciente se sienta mal, más allá de que sus parámetros vitales indiquen un problema o no. Para estas situaciones, el sistema prevé la utilización del mensaje `pButtonMsg`, que permite al usuario enviar una señal de emergencia al sistema. El sistema, al recibir la señal `pButtonMsg`, dispara una alarma que eventualmente resulta también en el envío de un equipo de primeros auxilios al domicilio del paciente. Se espera que cada vez que el sistema recibe una señal de emergencia, se envíe el equipo de primeros auxilios, sin excepción.

En el contexto de este trabajo, introducimos cambios en el modelo original a fin de hacer que esta última parte no sea cierta, y existan condiciones bajo las cuales una alarma puede ser disparada, pero el equipo de primeros auxilios no sea enviado de manera correcta. Luego de realizar estos cambios, mostramos que es factible obtener especificaciones correctas mediante PIAs y que estas no precisan esfuerzo adicional respecto de especificar con formalismos ya establecidos. En particular, un modelo válido como ambiente realizado mediante PIAs puede verse en la Figura 4. Como puede verse, no se introducen dificultades adicionales ni transiciones espúreas, más allá de las necesarias para ocultar elecciones internas del modelo.

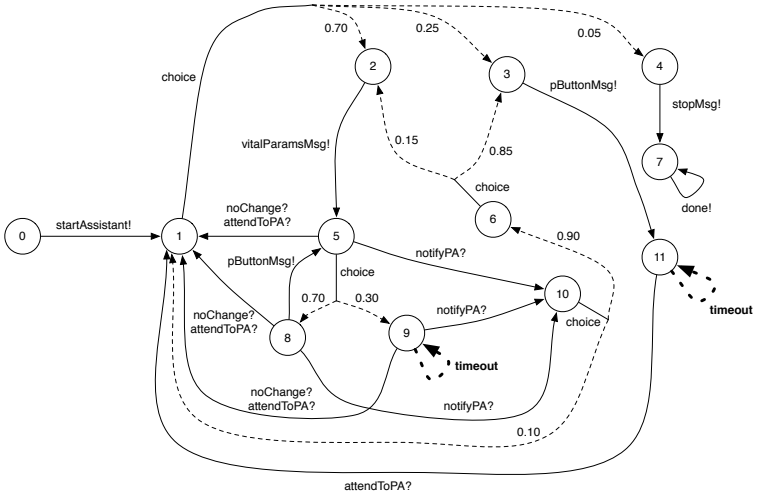


Figura 4: Un modelo PIA del paciente

A continuación, realizamos la validación de distintas propiedades y contrastamos los resultados obtenidos para las validaciones de los componentes de manera individual, contra los resultados obtenidos al analizar el sistema en su composición global. Las propiedades que evaluamos están detalladas en el Cuadro 1.

En este Capítulo, analizamos cada una de las propiedades en detalle, para ambos componentes (Sistema TA y Paciente), y mostramos que los resultados enunciados por el Teorema 3.1 efectivamente se sostienen como se esperaba.

Capítulo 5: Discusión

En este capítulo evaluamos los enfoques recientes que tienen puntos de contacto con nuestra propuesta de modelado. En particular nos focalizamos en los trabajos que tienen como objetivo el modelado de ambientes probabilísticos, a veces llamados también *perfiles de uso* [Che80, Mus93]. Podemos agrupar estos trabajos en dos grandes grupos: aquellos que apuntan a anotar modelos ya especificados con probabilidades recientemente relevadas, y aquellos que tienen como objetivo proveer herramientas de modelado con las probabilidades como parte fundamental del enfoque. Nuestro trabajo se engloba dentro de este segundo grupo.

Respecto del primer grupo, se evalúan trabajos que si bien resultan en artefactos anotados, lo hacen a nivel de composición [RM04, EGMT09]. En estos trabajos se desdibuja la relación entre el valor de las probabilidades y los componentes que las generan, haciendo imposible una verificación modular.

Dentro del segundo grupo, merece especial mención el trabajo de Delahaye et al. [DCL11] que presenta un enfoque inspirado en la anotación de contratos. La idea de contratos es una idea relacionada con aquella de interfaces, ya que establece las relaciones entre las responsabilidades de distintos componentes que interactúan. Sin embargo, este enfoque es ortogonalmente distinto al nuestro, lo cual permitiría aplicarlos de manera complementaria. En primer lugar, el trabajo citado analiza los contratos de manera aislada y resulta en una cota a la probabilidad de satisfacer cierta propiedad en el modelo compuesto. Nuestro enfoque, en cambio, establece una relación directa entre la probabilidad de satisfacción en cada componente, respecto de la probabilidad de satisfacción en el sistema compuesto. En segundo lugar, el objeto de estudio es distinto, ya que mientras en nuestro caso operamos sobre una especificación con semántica de ramificación, el trabajo citado lo hace sobre contratos,

Propiedad	Corresponde al
SP1: El paciente presiona el botón de pánico (`pButtonMsg`), pero sin embargo el equipo de primeros auxilios no es enviado al domicilio del paciente.	Sistema TA
SP2: El sistema determina que debe realizar un cambio de medicación (`changeDrug`) o de dosis (`changeDose`), y el siguiente mensaje que recibe el TA de parte del paciente genera una alarma que no resulta en el envío del equipo de primeros auxilios	Sistema TA
SP3: El TA recibe el mensaje de que el botón de pánico fue presionado durante su primera interacción con el paciente	Sistema TA
SP4: El TA recibe el mensaje de que el botón de pánico fue presionado durante *alguna* de sus primeras cinco interacciones con el paciente	Sistema TA
EP1: El paciente recibe la notificación de un cambio de medicación (`changeDrug` o bien `changeDose`) y reacciona inmediatamente presionando el botón de pánico	Paciente
EP2: El paciente presiona el botón de pánico durante su primera interacción con el sistema	Paciente
EP3: El paciente presiona el botón de pánico durante *alguna* de sus primeras cinco interacciones con el sistema	Paciente

Cuadro 1: Propiedades evaluadas para validación de PIAs

que tienen semántica de traza. Esta semántica no permite la noción de refinamiento de especificaciones, aunque sí puede modelar la composición y conjunción de sistemas.

Existen otros trabajos con la misma visión que el nuestro, basados en el principio de presunciones/garantías sobre autómatas determinísticos [KNPQ10, HKK13], y similarmente también sobre Cadenas de Markov Interactivas [HK09]. Sin embargo, estos trabajos no proveen una garantía de preservación de comportamientos a través de la composición en paralelo.

Respecto de la discusión entre modelado reactivo y generativo, existen varios trabajos que han trabajado sobre el problema. La discusión respecto del enfoque generativo se ha concentrado alrededor de determinar decisiones que permitan realizar la composición en paralelo de forma que el resultado sea un modelo formalmente válido, aunque tal vez no correcto desde el punto de vista del sistema a especificar [Chr90, DHK99].

Desde el punto de vista de la validación de interfaces, también se ha explorado la opción de utilizar los Autómatas de Entrada/Salida [LT87, WSS97]. Sin embargo, estos autómatas establecen condiciones de sincronización aún más estrictas que las de los Autómatas de Interfaz, resultando en especificaciones con transiciones espúreas que existen sólo con el objetivo de cumplir estos requerimientos foráneos al modelado en sí.

Capítulo 6: Verificación parcial eficiente

Más allá de que el formalismo de PIAs introducido en el Capítulo 4 permite el modelado composicional de sistemas de software, aún se mantiene el problema de que la verificación de este tipo de modelos se vuelve rápidamente infactible a medida

que los modelos crecen en complejidad y tamaño. En este Capítulo presentamos una técnica de verificación *parcial* que creemos puede ayudar cuando la factibilidad de realizar un análisis exhaustivo es amenazada. El enfoque que presentamos en este Capítulo es una combinación de simulación probabilística, inferencia de propiedades sobre conjuntos de observaciones, y verificación probabilística.

En el contexto de este Capítulo y aquellos que siguen, nos concentramos en la aplicación de esta técnica sobre los Autómatas Probabilísticos Simples. Vale recordar que los Autómatas Probabilísticos de Interfaz presentados anteriormente en esta tesis son un caso particular de los SPAs, por lo que la técnica presentada en este Capítulo es aplicable a PIAs, y sus resultados son equivalentemente extrapolables a los mismos.

La técnica que presentamos intenta atacar dos frentes que amenazan la factibilidad de los análisis. En primer lugar, el problema de la explosión de estados, que surge a medida que los modelos se vuelven más complejos en sus interacciones y, como consecuencia, crecen exponencialmente en tamaño.

El segundo problema que atacamos está principalmente relacionado con la resolución de sistemas probabilísticos. El análisis de este tipo de sistemas implica la resolución de un sistema de ecuaciones lineales, cuyo tamaño es equivalente a la cantidad de estados del sistema. Es claro que si es infactible almacenar el conjunto de estados en sí mismo, también será imposible analizarlo. Sin embargo, aún en los casos en que el conjunto de estados es almacenable, puede ser infactible analizarlo. Esto se debe a que métodos exactos de resolución como la eliminación gaussiana pueden tomar un tiempo excesivo sobre matrices de tamaño excesivo. Como consecuencia, suelen utilizarse métodos iterativos que apuntan a aproximar la solución en sus sucesivas iteraciones. Estas técnicas precisan un criterio para, eventualmente, detener la ejecución de la aproximación. El problema es que, en general, no hay garantía que nos permita conocer qué tan cerca del resultado real se detuvo esta ejecución. Sólo se pueden dar garantías parciales, como por ejemplo afirmar que al detener la ejecución siempre se obtiene un valor inferior (o igual) al real (es decir, cotas inferiores al valor real) en el caso del cálculo de recompensas; y cotas superiores en el cálculo de probabilidades.

La técnica que presentamos está inspirada en la idea de que analizar sólo una parte del espacio de estados puede proveer cotas más informativas (es decir, más cercanas al valor real) que las obtenidas por medio de un análisis exhaustivo. La hipótesis es que es posible identificar una porción pequeña del espacio de estados, pero significativa en términos de comportamiento y probabilidad de ocurrencia, considerando todos los estados fuera de esta porción como estados de error. La intuición es que, además de facilitar el almacenamiento de una menor cantidad de estados, los métodos iterativos sobre esta porción del espacio completo tienen, para un mismo presupuesto de tiempo de ejecución, la posibilidad de avanzar mucho más en sus iteraciones de aproximación.

Más específicamente, la técnica combina simulación, inferencia de propiedades y model checking probabilístico. Mediante la simulación obtenemos un conjunto de ejecuciones que representan el comportamiento esperado del sistema. Estas ejecuciones son analizadas y de ellas obtenemos un predicado invariante que las describe de manera global y sucinta. Finalmente, este invariante es utilizado para generar un submodelo del original, restringiéndose sólo a aquellos estados que cumplen el predicado invariante. Finalmente, este submodelo es analizado exhaustivamente mediante model checking probabilístico.

Exploraciones parciales

La base del enfoque que presentamos en este Capítulo es la noción de **submodelo** (Definición 6.1). Intuitivamente, un submodelo de un autómata probabilístico M es

otro autómata probabilístico que retiene algunos estados y transiciones de M, y donde el resto de los estados no retenidos son condensados en un estado *trampa* λ. Más formalmente, dado un modelo probabilístico $M = \langle S, s_0, A, R \rangle$, un *submodelo* de M es otro modelo probabilístico $M' = \langle S' \cup \{\lambda\}, s_0, A, R' \rangle$ tal que $S' \subseteq S$, $s_0 \in S'$, y $R' \subseteq (S' \cup \{\lambda\}) \times (A \cup \{\tau\}) \times \mathcal{D}(S' \cup \{\lambda\})$ es tal que, para todo $a \in A$

1. para cada $(\lambda, a, \mu_{R'}) \in R'$, debe valer que $supp(\mu_{R'}) = \{\lambda\}$ y $a = \tau$;
2. para cada $s \in S'$, y para cada $a \in A \cup \{\tau\}$, debe ser que $\exists \mu_{R'} \in \mathcal{D}(S' \cup \{\lambda\})$ es tal que $(s, a, \mu_{R'}) \in R' \iff \exists \mu_R \in \mathcal{D}(S)$ donde $(s, a, \mu_R) \in R$;
3. para cada $s_1, s_2 \in S'$ y cada $a \in A \cup \{\tau\}$ debe ser el caso de que $\exists \mu_{R'} \in \mathcal{D}(S' \cup \{\lambda\})$ donde $(s_1, a, \mu_{R'}) \in R' \wedge s_2 \in supp(\mu_{R'}) \Rightarrow \exists \mu_R \in \mathcal{D}(S)$ tal que $(s_1, a, \mu_R) \in R \wedge \mu_R(s_2) = \mu_{R'}(s_2)$;
4. finalmente, para cada $s_1 \in S'$ tal que $s_1 \neq \lambda$, y cada $a \in A \cup \{\tau\}$ debe darse que $\exists \mu_{R'} \in \mathcal{D}(S' \cup \{\lambda\})$ tal que $(s_1, a, \mu_{R'}) \in R' \Rightarrow \exists \mu_R \in \mathcal{D}(S)$ donde $(s_1, a, \mu_R) \in R \cdot \mu_{R'}(\lambda) = 1 - \sum_{s_2 \in supp(\mu_{R'}) \setminus \{\lambda\}} \mu_R(s_2)$.

La cláusula 1 indica que el estado λ efectivamente funciona como estado trampa, sólo aceptando además la acción interna τ. La cláusula 2 establece que cualquier transición, salvo las originadas en λ, es una transición que ya estaba presente en el modelo M. De manera similar, la cláusula 3 nota que las probabilidades de estas transiciones también son preservadas del modelo original, excepto por aquellas que fueron redirigidas al estado trampa λ. Finalmente, la cláusula 4 indica que la probabilidad asignada a las transiciones que llevan al estado λ coinciden exactamente con la probabilidad restante una vez que se tuvieron en cuenta las probabilidades de las transiciones que se conservan dentro del submodelo.

De esta forma, un submodelo preserva cierta parte del comportamiento presente en el modelo original. Como consecuencia, resulta también que existe una relación entre los planificadores que pueden utilizarse para resolver no determinismo en el modelo original, y aquellos que pueden utilizarse en el submodelo. La noción de **planificadores restringidos** (Definición 6.2) captura esta relación. Más aún, es fácil ver que cualquier planificador aplicable a un submodelo M' de M puede ser extendido a un planificador válido para M. Esta relación entre planificadores es clave para entender por qué es válido el resultado de acotación de los análisis sobre un submodelo frente al análisis de un modelo completo. Intuitivamente, si la probabilidad de cierto evento en un submodelo es p', esta probabilidad tuvo que haber sido obtenida mediante un planificador σ' del submodelo. Dado que este planificador σ' es válido también para el modelo original, el evento también debe ser posible en el modelo original. Sin embargo, como este planificador puede ser extendido a otros planificadores para el modelo original, estos planificadores extendidos podrían planificar acciones entrelazadas que reduzcan esta probabilidad. Por lo tanto, la probabilidad del evento en el modelo original será forzosamente un valor $p \leq p'$. Un análisis análogo permite mostrar que los valores de recompensas obtenidas sobre un submodelo deberán ser necesariamente menores o iguales que los valores reales de la misma recompensa sobre el modelo completo. Los Teoremas 6.2 y 6.1 capturan esta idea, y son demostrados en este Capítulo. La demostración formal sigue el argumento de planificadores restringidos que delineamos anteriormente.

Generación automática de submodelos

Si bien es cierto que cualquier submodelo conlleva estas propiedades de acotación de los valores de probabilidades y recompensas, es clave que, si nuestro objetivo es

un enfoque aplicable y eficiente, seamos capaces de obtener submodelos tales que las cotas que obtienen sean útiles. Una validación preliminar presentada en este Capítulo muestra que no cualquier submodelo es igualmente útil. Por ejemplo, submodelos obtenidos por medio de una exploración parcial DFS (profundidad primero) no es tan útil como una más abarcativa obtenida por BFS (a lo ancho primero). Pero por otra parte, las exploraciones BFS son también poco útiles en algunos casos. La clave en todos los casos es maximizar la probabilidad de, en cada transición, mantenerse dentro del espacio de estados del submodelo.

Lamentablemente, dado un tamaño deseado de submodelo, el problema de encontrar este modelo más probabilísticamente denso de ese tamaño es intratable [JD07]. En cambio, nuestro enfoque adopta una heurística para obtener estos submodelos, concentrándonos en el hecho de que esta densidad probabilística debería ser, de alguna manera, observable si evaluamos las ejecuciones que tendría el sistema durante su tiempo de vida. Nuestro enfoque apunta a aproximar estos submodelos deseables mediante una simulación (acotada) del comportamiento del sistema. Es decir, la base de nuestro enfoque implica la simulación de una cantidad considerable de ejecuciones del modelo completo. El conjunto resultante resultará testigo del comportamiento real, y por lo tanto debería cubrir buena parte del modelo probabilísticamente más denso.

Por otra parte, entendemos que es más eficiente obtener una descripción semántica de estos submodelos, en contraposición a una representación explícita sintáctica. De esta manera, además de capturar el comportamiento visto en las simulaciones, podemos además capturar comportamiento relacionado que ha probado ser útil de analizar, tales como simetrías, eventos independientes, y diferentes planificaciones de los mismos eventos pero en distintos órdenes [BK08], que pueden contribuir significativa y positivamente, a la probabilidad de mantenerse dentro del submodelo.

Más formalmente, esta descripción semántica está dada por el concepto de **invariante** de un conjunto de ejecuciones (Definición 6.3). Mediante esta noción de invariante, podemos hablar del **submodelo inferido por un invariante** (Definición 6.4). Formalmente, dado un modelo probabilístico $M = \langle S, s_0, A, R\rangle$ y ψ una fórmula (posiblemente un invariante obtenido de las ejecuciones); el submodelo inferido por ψ es el submodelo $M' = \langle S' \cup \{\lambda\}, s_0, A', R'\rangle$ of M tal que

a) cada $s' \in S'$ es tal que $s' \models \psi$;

b) para cada $s'_1 \in S', s'_1 \neq s_0$, vale $s_0 \xrightarrow{\alpha} s'_1$; y finalmente

c) para todo estado $s'_2 \in S \setminus S'$ tal que existe otro estado $s'_1 \in S, (s'_1, a, \mu_R) \in R$ con $\mu_R(s'_2) > 0$, vale que $M, s'_2 \not\models \psi$.

Dicho de otra manera, si un estado s'_2 no es parte del submodelo, pero es alcanzable desde otro estado s'_1 que sí se encuentra en el submodelo, debe ser que s'_2 viola la propiedad ψ. Es decir, el submodelo está conectado de manera maximal desde el estado inicial a través de la fórmula ψ.

Automatización de la técnica

Vale notar que en todo momento nuestro objetivo es obtener una técnica completamente automática, que no requiera de la intervención de un operador humano en ninguna de sus fases. Esto implica la necesidad de obtener invariantes de manera automática. Para esto, nos valemos de la posibilidad de realizar, de forma automática y sistemática, repetidas ejecuciones simuladas acotadas sobre el sistema real. Luego de obtener este conjunto de ejecuciones finitas, utilizamos la herramienta

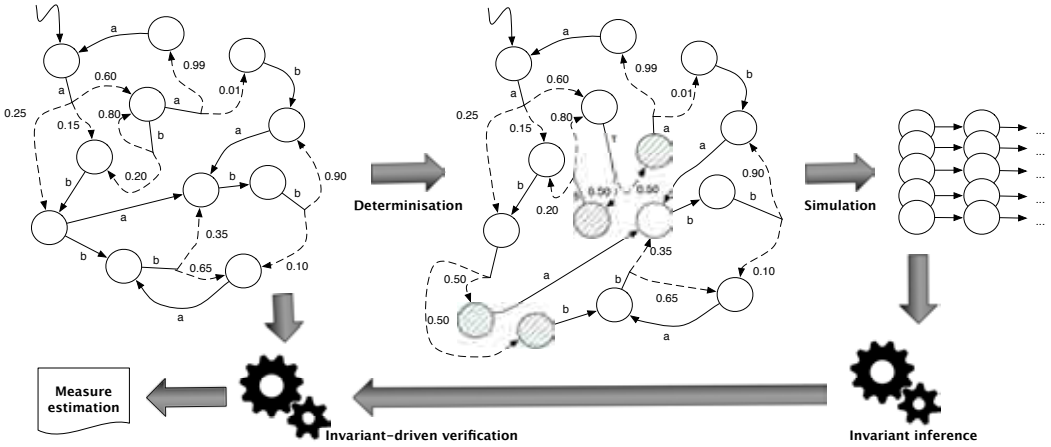


Figura 5: Procedimiento del análisis basado en exploraciones parciales

Daikon [EPG+07], un motor de inferencia de propiedades sobre observaciones, para obtener los predicados invariantes que se mantienen verdaderos a través de todos los estados explorados durante estas simulaciones. Estos predicados, a su vez, son utilizados para construir, también de manera automática, un autómata observador que, mediante el monitoreo de la validez de la propiedad durante la construcción del sistema compuesto, permite generar un submodelo inferido acorde.

La figura 5 describe el procedimiento general de nuestra técnica propuesta en sus distintas fases.

Capítulo 7: Validación empírica

En este capítulo ponemos nuestro enfoque a prueba. Esto comprende tres preguntas fundamentales.

El primero de estos interrogantes tiene que ver con la capacidad de nuestra técnica de proveer mejores cotas (es decir, más precisas) que los enfoques de model checking establecidos al momento. Realizamos esta experimentación tanto para el cálculo de probabilidades como también así para el de recompensas.

En segundo lugar, comparamos nuestro enfoque y sus resultados contra mecanismos de verificación basados en muestro estadístico, es decir, métodos Monte Carlo.

Finalmente, comparamos nuestro enfoque automático con aquel donde podemos introducir algo de experiencia e intervención humanos. Esta intervención está especialmente enfocada en la generación de predicados invariantes, ya que un ingeniero con conocimiento del dominio de la propiedad que se está analizando puede ser capaz de proveer invariantes que nuestro enfoque automático tal vez no puede hallar. Nos interesa en este caso comparar el esfuerzo entre la generación automática y la manual, además de comparar los resultados obtenidos por cada técnica.

Todas estas preguntas fueron aplicadas sobre varios casos de estudio, es decir, distintos sistemas de software y los ambientes con los que ellos interactúan. En primer lugar, modelamos un sistema de encolado de tareas en tandem, es decir, en dos colas sucesivas. El evento de falla de interés en este caso es el hecho de que ambas colas pueden quedar simultáneamente llenas, bloqueando cualquier otra tarea.

En segundo lugar, analizamos un protocolo de envío de datos entre dos computadoras. Este protocolo no es confiable, por lo que se utilizan bits de control a fin de no repetir datos innecesariamente en casos en que los paquetes enviados se pierden. Analizamos dos variantes de este protocolo y su entorno. En el primer caso, el cliente

Caso de estudio	Modelo del sistema	Determinismo del ambiente	Propiedades
Cola en tandem	LTS no determinístico	Determinístico	Tiempo promedio a la falla Probabilidad de falla en tiempo acotado
Bounded Retransmission Protocol	DTMC	Determinístico	Tiempo promedio a la falla Probabilidad de falla en tiempo acotado
Bounded Retransmission Protocol	DTMC	No determinístico	Tiempo promedio a la falla Probabilidad de falla en tiempo acotado
IEEE 802.3 CSMA/CD	SPA	No determinístico	Tiempo promedio de *turnaround*
Red infectada	SPA	No determinístico	Tiempo promedio a infección total Probabilidad acotada de infección total Probabilidad acotada de infección parcial

Cuadro 2: Resumen de los casos de estudio analizados

envía archivos de diverso tamaño, eligiendo de manera probabilística el tamaño de archivo a enviar. En la segunda variante, esta elección de tamaño es no determinística y no está cuantificada. El evento de interés en ambos casos es la superación de cierto límite de reintentos, tras lo cual se aborta el envío del archivo en cuestión.

El tercer caso de estudio que trabajamos es el del protocolo de detección y evasión de colisiones en el envío de datos por medio de redes *wireless*. Este protocolo está descripto por el estándar IEEE 802.3 y prevé la utilización de tiempos de espera en el caso de colisiones. Estos tiempos de espera se incrementan de manera exponencial en el caso de detectar nuevas colisiones. En este caso, el evento de interés no es un error, sino que nos interesa saber cuánto tiempo debe transcurrir en promedio para que dos terminales, que compiten por el medio de transimisión, puedan enviar con éxito sus datos.

Finalmente, modelamos el caso de una red de computadoras donde una de ellas se encuentra infectada por un virus y puede contagiar a sus vecinas. En este caso, los eventos de interés son la infección de un nodo determinado de la red, y la infección de la red por completo.

El Cuadro 2 resume los casos de estudio analizados en esta tesis.

Planteo experimental

En cada caso, tomamos modelos de la literatura que fueron analizados con anterioridad en la comunidad científica. En algunos de estos casos, modificamos los modelos existentes a fin de hacerlos más complejos e interesantes si estos eran demasiado pequeños para poder realizar un análisis exigente de la técnica. Además, realizamos modelos de sus entornos operativos, en los casos en que éstos no estaban disponibles. En cada caso, chequeamos exhaustivamente que la composición de todos los componentes sea válida respecto de las restricciones planteadas por los Autómatas Probabilísticos de Interfaz.

Las propiedades de interés fueron modeladas, en cada caso, mediante fórmulas

de estado a fin de poder verificar de manera automática su alcanzabilidad. Adicionalmente, establecimos estructuras de recompensas adecuadas en los casos en los que, además de medir la probabilidad del evento, nos interesaban otras dimensiones asociadas (por ejemplo, el tiempo promedio hasta alcanzar el evento en cuestión).

En aquellos casos en los que fue factible aplicar la técnica de model checking exhaustivo, lo hicimos a fin de obtener una cota inicial a los resultados buscados. En los casos en los que los modelos pudieron ser analizados de manera analítica lo hicimos, mientras que en aquellos que no, aplicamos el model checker PRISM [HKNP06]. Es importante recordar que, a diferencia de un cálculo analítico, el model checker realiza aproximaciones numéricas para llegar al resultado deseado. Dado que esta aproximación puede no converger en tiempo, notamos el tiempo de corte de la ejecución y advertimos que este resultado debe tratarse como una cota, y no como el resultado real.

Finalmente, pusimos a prueba nuestro enfoque en cada caso de estudio. Obtuvimos, para cada caso, distintos invariantes variando los parámetros iniciales de simulación, es decir, la cantidad de ejecuciones simuladas y su longitud. Utilizamos la herramienta Daikon v4.6.4 [EPG+07] para producir invariantes. La herramienta fue configurada a fin de que los invariantes obtenidos fuesen conjunciones de términos de la forma $x \sim y$, donde x e y son o bien variables del modelo o constantes numéricas; y $\sim \in \{<, \leq, =, \geq >\}$. En estos casos de estudio, un *estado* del modelo está representado por las distintas valuaciones que pueden llegar a tomar estas variables del modelo.

Los invariantes obtenidos fueron utilizados para construir de manera automática un modelo *observador* O que monitorea en todo momento la validez del invariante. Este observador es un autómata en sí mismo que, al ser compuesto con el modelo del sistema M, sincroniza con todas sus acciones y fuerza a evolucionar hacia el estado λ cada vez que el estado de destino original resultaría en una violación del invariante obtenido. Gracias a esta manera monitoreada de construir la composición, el modelo que obtenemos es con seguridad un submodelo del sistema original.

El Cuadro 3 resume de manera sucinta los resultados obtenidos para cada caso de estudio para las preguntas que nos planteamos en esta tesis.

Capítulo 8: Discusión

Respecto de la técnica presentada anteriormente, tal vez el punto que merezca más trabajo es el de poder determinar, de manera automática, los parámetros de simulación (cantidad y longitud de trazas) que maximicen, durante la fase de análisis del modelo parcial, la utilidad de los resultados obtenidos. Afortunadamente, se desprende de los ejemplos estudiados que, en general, se requiere una cantidad discreta de trazas y una longitud también moderada. Además, se puede ver que la velocidad con la que se obtienen estimados iniciales es muy veloz, mientras que estas estimaciones tienden a estancarse una vez establecida esta estimación inicial. Esto supone una oportunidad para, de manera rápida, poder comparar la efectividad de dos configuraciones de simulaciones distintas. Tales comparaciones pueden incluso paralelizarse a fin de obtener rápidamente la mejor combinación tomada de un conjunto de valores posibles dados.

Por otra parte, es válido notar que esta técnica es completamente ortogonal, y por lo tanto se beneficia, de otras técnicas de optimización de verificación de modelos [KKZ05, HKNP06, SVA05b, You05, KNP06, DG97].

Existen otros enfoques que buscan también realizar mediciones sobre modelos incompletos. Por ejemplo, el trabajo de [ZVB11, CBvB12] apunta a proveer una

Tandem Queue (mean time to failure)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	$4{,}2 \times 10^5$	TO	$\mathbf{7 \times 10^7}$	**TO**	N/A	TO	$5{,}5 \times 10^7$	TO
Tandem Queue (bounded reachability probability)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	0,0000	TO	0,0713	TO	N/A	TO	$\mathbf{2{,}28 \times 10^{-6}}$	**19 hs**
Fully probabilistic BRP (mean time to failure)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	OOM	TO	$\mathbf{2{,}5 \times 10^7}$	**TO**	N/A	TO	$1{,}69 \times 10^7$	TO
Fully probabilistic BRP (bounded reachability probability)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	OOM	TO	0,0680	22 hs	N/A	TO	**0,01319**	**7,9 hs**
Non-deterministic BRP (*minimum* mean time to failure)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	OOM	TO	$\mathbf{5{,}6 \times 10^6}$	**TO**	N/A	TO	9999	126,25 s
Non-deterministic BRP (*maximum* mean time to failure)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	OOM	TO	$\mathbf{9{,}8 \times 10^6}$	**TO**	N/A	TO	9965,87	46,26 s
Non-deterministic BRP (*minimum* bounded reachability probability)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	OOM	TO	**0,02382***	**8,6 hs***	N/A	TO	0,01239	17,5 hs
Non-deterministic BRP (*maximum* bounded reachability probability)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	OOM	TO	0,71205	TO	N/A	TO	**0,01321**	**16,2 hs**
WLAN (*minimum* mean turnaround time)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
1725,00	1725,00	628,00 s	**1725,00**	**0,98 s**	N/A	N/A	1665,63	490,05 s
WLAN (*maximum* mean turnaround time)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
4301,65	4301,65	54149 s	**4300,67***	**2 s***	N/A	N/A	3846,17	1085,87 s
Constrained Virus (*minimum* mean time to total infection)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
5200,00	OOM	TO	500,54	2771 s	N/A	N/A	**999,32**	**414 s**
Constrained Virus (*minimum* mean time to corner infection)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
1200,00	OOM	TO	599,54	1452 s	N/A	N/A	**999,32**	**1242 s**
Constrained Virus (*maximum* bounded probability to total infection before 5200 steps)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
0,51872	OOM	TO	**1,0000**	**$\sim$ 0 s**	N/A	N/A	**1,0000**	**$\sim$ 0 s**
Constrained Virus (*maximum* bounded probability to corner infection before 1200 steps)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
0,53898	OOM	TO	0,97997	1004 s	N/A	N/A	**0,75805**	**420 s**

Cuadro 3: Resumen de (mejores) resultados para cada técnica y caso de estudio. TO indica corte de ejecución tras 24 horas. N/A denota resultados que no pudieron ser obtenidos por superar el tiempo, o que no son confiables debido a la técnica subyacente.

medida de avance de un verificador hacia la resolución de una pregunta respecto de una propiedad dada. Sin embargo, esta medida no está relacionada con el dominio del problema ni tampoco con el predicado que se desea validar.

Desde el punto de vista de los análisis estadísticos del estilo Monte Carlo, el mayor problema que los afecta es que sólo pueden funcionar para propiedades acotadas en el tiempo, esto es, de la forma $\psi \mathcal{U}^{\leq T} \rho$ donde T es un tiempo fijado de antemano. En los casos donde la propiedad no es acotada en el tiempo, el hecho de que las observaciones simuladas sí sean acotadas impide clasificar a cada simulación como válida o no respecto de la propiedad. Trabajos tales como [SVA05a, RP09, LP06, BGH09, RK08, MSW12, MSW13] apuntan a sesgar las simulaciones con el fin de poder determinar estos valores de verdad. Sin embargo, el impacto de este sesgado no puede ser cuantificado en general, lo cual amenaza la validez estadística de estos enfoques. Otros enfoques interesantes son aquellos que apuntan a la simulación estratificada [RC05, VAVA94], donde las simulaciones son sucesivamente recomenzadas desde puntos intermedios, con la esperanza de que se aproximen al evento de interés. Estas técnicas requieren, sin embargo, un análisis exhaustivo que permita predecir dónde realizar estos recomienzos. Por otra parte, debe tenerse en cuenta el hecho de que este recomienzo puede introducir un sesgo. Debe, en todo caso, medirse el impacto de este sesgo.

En contraste, el trabajo en [YCZ11] propone dos técnicas que no dependen de un sesgo en la simulación. Sin embargo, una de ellas requiere un número excesivo de muestras simuladas, tanto que no pueden ser obtenidas en un tiempo razonable; mientras que la segunda requiere un procesamiento que precisa que el modelo completo sea explorado de antemano, justamente uno de los puntos que deseamos evitar con nuestra técnica.

Otro punto a analizar es nuestra decisión de reemplazar no determinismo por distribuciones equiprobables durante la simulación. Si bien este enfoque es correcto, puede no ser óptimo respecto de la calidad de las cotas que pueden obtenerse. En este sentido, son interesantes los trabajos que intentan desviarse hacia los planificadores extremos [HMZ+12, BFFHH11, LPD+14], que proveen, de manera acorde, los resultados mínimos y máximos.

Capítulo 9: Conclusiones

En esta tesis, trabajamos sobre el problema de la verificación de sistemas, con un foco en la verificación cuantitativa, y con el objetivo de producir resultados aún cuando una exploración exhaustiva no es factible. La técnica que proponemos permite, mediante el modelado apropiado de ambientes operativos, cuantificar estas exploraciones parciales y obtener información parcial referida a la propiedad de interés.

Nuestra técnica propuesta puede, en algunos casos, proveer cotas sobre la probabilidad o las recompensas asociadas a propiedades de alcanzabilidad sobre el sistema bajo análisis. Esto puede ser útil para argumentar que el sistema, a pesar de no haber sido evaluado de manera exhaustiva, de todas maneras cumple con garantías mínimas respecto de su confiabilidad u otras prestaciones.

La resolución del no determinismo en estos sistemas es aún un tópico de interés para investigaciones futuras. En particular, el mecanismo de resolución que utilizamos a lo largo de este trabajo hace que todas estas decisiones sean uniformes, lo cual hace que las exploraciones parciales también lo sean, y no se reduzcan lo suficiente en tamaño. En particular, creemos que puede ser interesante aplicar estrategias de simulación que emulen un planificador que fuerce la ejecución fuera de los estados de error de interés, a fin de maximizar el tiempo de ejecución dentro de estos submodelos.

Además, es interesante focalizar la investigación en la búsqueda de planificadores que permitan encontrar rápidamente cotas que se aproximen a los valores extremos de probabilidades y recompensas. Asimismo, creemos que es interesante extender el uso de la técnica hacia otros formalismos de modelado, incluyendo eventualmente la aplicación sobre código fuente o programas binarios.

Desde un punto de vista más ingenieril, el trabajo se sostiene sobre un nuevo formalismo de modelado de comportamiento probabilístico y no determinístico. En este sentido, presentamos los Autómatas Probabilísticos de Interfaz como una alternativa adecuada para el modelado composicional, ya que garantiza la conservación de las propiedades a nivel componente dentro de la composición. Además, al presentar este formalismo relajamos además las restricciones más fuertes respecto de sincronización que eran planteadas por los Autómatas de Interfaz tradicionales. Sin embargo, creemos que requerir ecuanimidad absoluta para los planificadores puede aún resultar demasiado restrictivo. Este área será foco de trabajo futuro a fin de relajar aún más estas restricciones.

Finalmente, hemos presentado evidencia experimental que nos permite aseverar con alto grado de confianza que la técnica propuesta resulta de utilidad en los casos en que un análisis exhaustivo no es factible. Sin embargo, existen áreas donde profundizar el trabajo, tales como la inferencia de los parámetros óptimos de simulación, y la realización de experimentos que permitan argumentar que el uso de invariantes obtenidos de manera automática es más útil (o demanda menos esfuerzo) que el uso de invariantes obtenidos por medio de un enfoque de inspección y análisis manual.

Part I

Prelude

CHAPTER 1

Introduction

1.1. Foreword

The document that you, the reader, have in your hands is succinctly defined as being a "Ph.D. thesis". Whatever that may *actually* mean, the fact is that this thesis, not as an exception but rather as a generality, is quite a complex document. The main reason for this complexity is that, although this is a whole, and (hopefully) coherent document, the ideas and concepts presented here have not been written either in one go nor linearly. Rather, the process has been quite the contrary. This document is the result of a journey of several years of research, during which there have been many side roads, backtracks, detours and even U-turns. During this period of work, many complex concepts were drawn up, ideas were both produced as well as fed on, experiments were set up and evidence was gathered. The aim of this document is then to present all of this work in a way that can be read (and hopefully understood and better yet, enjoyed) by several different audiences, who come with different backgrounds.

The goal of this foreword and introduction is to ease up the process of approaching this document. Here, we will provide a short and informal summary of the topics, problems and solutions tackled by this thesis. In the spirit of keeping this summary simple, formality, rigour, detailed explanations and citations are missing from this introduction. However, the experienced reader should not fret, as every concept introduced here will be properly defined further on in the thesis. As a result, this section is by far the most readable and easily approachable.

This introduction will conclude with a roadmap of the remainder of the document. Some readers may be interested in reading the whole thesis, while others may be interested only in a fraction of it. This introduction, along with its roadmap at the end, should give every reader, no matter her background or interests, a good notion of where to find the topics she would like to know more of.

So now, without further ado, we begin this introduction by discussing why model checking of systems is an important topic, and which are the particular challenges we tackle in this thesis.

1.2. Motivation

In the last years, software systems have become pervasive; and also have come to perform tasks that are increasingly more critical. The presence of software systems in our world ranges from small, easily updateable everyday devices like tablets and smartphones, to the more critical, and harder to modify or upgrade, like spaceflight and plant controllers. These systems are highly reactive, that is, they are not designed to perform a batch computing task until completion. Rather, they are designed to respond to external events such as environment sensing and user interaction, possibly taking into account previous events and the own system's reactions.

As a result of this reactivity and the intricacy of the possible interacting environments, the complexity of these systems increases accordingly. As the system grows more complex, so does the possibility of introducing errors that may prove crippling to the system's ability to perform its intended task. As a result, techniques that can ensure that a software system will perform its task flawlessly are desirable. Since deployment and fixing costs increase as the development process matures and the system is deployed, validation and verification techniques that can be applied earlier in the process are valuable.

Since the focus of this thesis is in techniques that can be applied early in the process, the deliverable under analysis will not be the actual implemented software system, but rather an abstract description of it. This abstraction is a formal one, since it has a definite syntax and unambiguous semantics, which makes it amenable to a rigorous analysis. These descriptions that we will work on take the form of Labelled Transition Systems (LTS), which will be defined further on in Chapter 2. For now, it will suffice to say that an LTS is a set of system *states*. From each state it is possible to traverse to another state, through the triggering of an *action*.

When analysing these descriptions, it will be interesting to answer questions regarding their ability to perform the required task. For example, if we were to analyse the controller of a car's engine and braking system, we could ask questions such as "does pressing the brake pedal always result in the wheels being braked?", or "is the gas injection cut off every time that the engine surpasses 8000 revolutions per minute?". Useful techniques are those that can answer these questions with definite *yes* or *no* answers; and that when answering *no*, are able to provide a counterexample to back this negative claim. Following the previous example, such a tool could answer *yes* to the second question, but provide a negative answer for the first, while informing the engineer that the wheels are not braked if the emergency brake was already applied.

Model checking is an example of these techniques. Given a software model $\mathcal{M}$ such as one expressed by an LTS and a property ϕ (which is expressed in some modal logic, usually one that can reason about the time ordering of events), model checking is an effective procedure to answer whether $\mathcal{M}$ satisfies ϕ, usually noted $\mathcal{M} \models \phi$. Unfortunately, the problem of *state explosion* seriously hampers applicability. This problem stems from the fact that even small modifications or additions to one aspect of the model can impact greatly on the complete model, by generating an inordinate number of states that the model checking procedure needs to explore. From a theoretical point of view, it is known that the problem of model checking, for the type of systems and properties we are interested in, lies in the PSPACE-complete class of problems.

There has been much research aimed at palliating the state explosion problem. Some techniques, such as partial order reduction, attack the problem by optimising the system exploration, avoiding the generation of states that are known to be, in some way, equivalent to others already visited. Other techniques have the goal of

minimising the memory requirements while still being able to generate the whole space of states. For example, symbolic representations of states manage to avoid explicitly enumerating the states by the use of boolean conditions that successively refine the state space, eventually refining it to single states. Incremental state space building (also called *on-the-fly* state generation), which generates states as their analysis is needed, also helps in reducing the generated state space, especially in the case where a counterexample can be found without examining the whole system. All these techniques can complement each other and contribute to making model checking applicable to larger domains.

Yet, there is a limit to how much these techniques can help, and it is often the case that complex systems grow large enough that no combination of techniques is able to reduce the state space so that its analysis is made feasible. Even if the whole system state space could be generated, we must take into account that verification procedures are also costly in execution time. It is usually the case that the time budget for verification activities does not allow for such costly procedures, and verification tasks would need to be cut short.

In such cases, what can we expect from a technique such as model checking? Unfortunately, not much. Recall that a successful model checking procedure has two possible outcomes: either a *yes* answer, meaning that the state space was *completely* explored and no violation to the property was found; or else a *no* answer, which comes accompanied by a counterexample exhibiting the property violation. Positive answers always need the whole state space to be explored, while the negative ones may not, as a counterexample may be found at any point in the analysis. Therefore, if the model checking procedure was terminated (either because the memory was exhausted, or the time budget was consumed), and no definite answer was provided, we can only know for sure that, up to the point where termination was forced, the procedure *did not* find any property violations (if it had, it would have had the evidence necessary to provide a counterexample). For most applications, such a vague answer is not enough, as there could be countless ways for the software to fail that are present in the portion of the state space that the procedure did not analyse.

This is clearly a lost opportunity. The software model has already been formally modelled, requirements were elicited and expressed in a suitable logic, and a model checking procedure is in place that could, in theory, answer the satisfaction question for this model and property. Even more, if the procedure was actually put to work, it expended a (possibly large) time budget, and still failed to provide an answer.

This thesis is kicked off by this scenario. The main question that we set ourselves to answer is this: *can we, when faced with models and properties for which model checking procedures have failed (be it for memory or time reasons), nevertheless infer some information that is actually useful for the user that intended to verify that model and property?* In a very simple sense, we want to fulfil the promise in the box below in Figure 1.1.

In the following we summarise the steps we took to answer this question, explore the contributions' box in Figure 1.1 in more detail, and we direct the reader to specific chapters discussing each concept and idea in detail.

1.2.1. Quantitative vs. qualitative information

The model checker procedure we described up to now is such that, when it terminates successfully, gives as a result some *qualitative* information; that is, it answers whether the model under analysis satisfies the given property, or not. As we have discussed in the previous section, the problem is that whenever the procedure fails to terminate properly, it returns no answer. Further, if we required the model checker

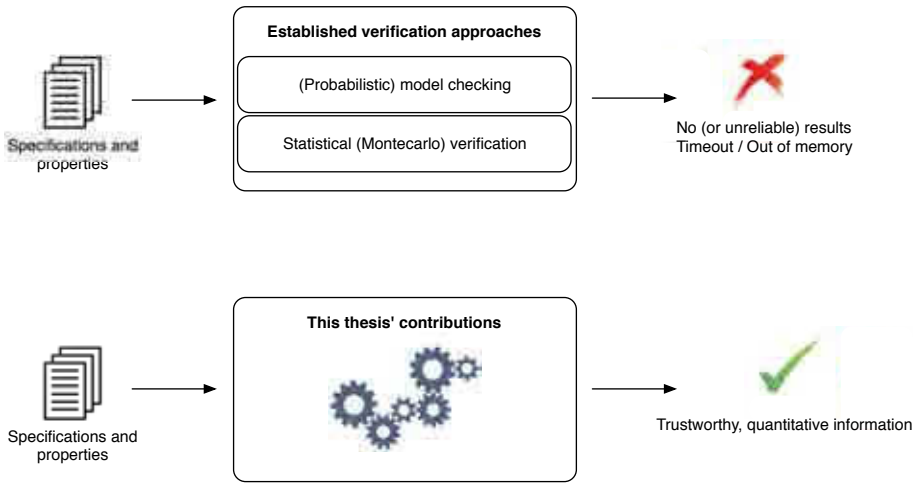


Figure 1.1: Expected contributions of this thesis

to return some *safe* qualitative information, that is, an answer that errs on the side of caution, the only possible answer would be *no, the model under analysis potentially does not satisfy the property.*

Even though qualitative questions do not convey much information in this case, there are some *quantitative* questions that can be asked about the portion of the model that was explored before failing. For example, "how much of the whole system was found to be free of property violations?"; or "how *confident* can we be that the system does not exhibit a violation, given that the model checking procedure did not see one so far?". The answer to these questions can provide some interesting feedback on the failed verification effort. This brings us to the first contribution of this thesis.

> The work presented here provides a way to obtain useful quantitative information about the validity of a property even if a complete, automatic model check is infeasible.

Quantitative answers that are based solely on state space size or on its topology are usually not very informative. For example, we could have an educated guess about the expected size of the full model (even though it has not been built) and answer that a certain percentage of this expected size was analysed without finding evidence of property violations. But, what would such an answer *actually mean* to the engineer posing the question? For instance, recall the example of verifying a model of a car's controller. This controller acts over the engine and braking system, and has anti-lock capabilities (ABS). Assume as well that the property of interest is that every time that the brake pedal is pressed, the wheels should not lock. Now, suppose we tried to model check this system, along with its property, and that we ran out of memory after exploring 75% of the state space. As was discussed earlier, no failure was found within this explored state space. But, what useful information can we obtain from this failed model check? What if it turns out that most of the explored states only depict situations where the car is already at zero speed, or with its engine off and emergency brake applied? We have a sense that some states are *more important* or *more interesting* than others; and those explored in this case are clearly the least interesting.

What would be a good measure of how interesting a given state is? Within this

thesis we will argue that a state will be of greater interest if this state is *more likely* to be witnessed during the actual execution of the system. There are several factors that could impact this likelihood. First, given the high reactivity of these systems, the likelihood of a given state being witnessed in actual operation will be closely related to the likelihood of the events that trigger a transition to such a state. Further, the likelihood of events may itself be influenced by the responses of the system. For example, a system that controls the elevators on an intelligent building is more likely to receive requests to go down from upper floors at morning (when people leave to work), and conversely more likely to receive requests to go up in the afternoon (when people return). This information is usually captured in what is called an *operational profile*, about which we will expand later on.

A second source of likelihood information is the system itself. Just as some environmental events may be more likely, so may be some of the system's actions. For example, and going back to the elevator system above, it may be desirable to balance the work load of the elevators. One possible way to do this would be, once a request is received, to choose the elevator randomly between the elevators closest to the floor where the request originated.

To convey likelihood information such as the one described above, we will enrich our system and environment models with notions of probability theory. We will also allow non-deterministic information to be conveyed by these models, since it is often the case that the likelihood of different, concurrent actions, cannot be quantified. In Chapter 2 we will summarise and quickly refresh the notions we will employ in the course of this thesis.

1.2.2. Modelling probabilistic information

Of course, we are not the first to discuss probabilistic models in the context of software engineering. Several different modelling formalisms exist, which differ mostly in two ways. First, they may differ in the nature of the probability distributions that they allow. For example, some formalisms such as Continuous Time Markov Chains model probabilistic transitions, as its name implies, through continuous time distributions. Alternatively, Discrete Time Markov Chains model transitions as discrete probability steps.

A second way in which they differ is more oriented towards the interactive nature of software systems. Some models allow the probabilistic choices to discern between different possible actions of the model. Others restrict the probability distribution to just choose the result of taking a single action. Yet another kind of models establishes a strict alternation between actions that may have probabilistic results and actions for which their outcome is not probabilistically quantified and is left non-deterministic.

The problem that we identify in this thesis, however, is that none of these formal models allow for a meaningful compositional approach to system model construction. The main problem is that measuring the likelihood of a system component making a choice, independently of the behaviour of its environment (which may include other components), can be notoriously difficult. Quantifying this isolated choices properly may require a careful decomposition of probabilities that were estimated or measured from the actual setting of the system. These compound probabilities need to be decomposed into conditional ones, and these should in turn be incorporated into a component description that will form part of the composite system.

We characterise the problems that arise from trying to perform this decomposition as a result of a lack of an appropriate treatment of the notion of action controllability in combination with probabilistic descriptions. This leads to problems such

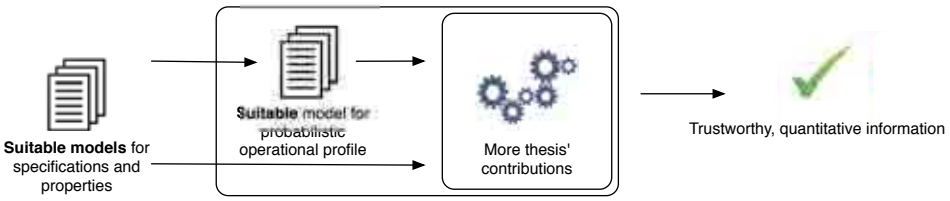


Figure 1.2: First contributions, adding probabilistic environment information and improving on probabilistic modelling

as

- probabilistic semantics that are unclear, in the sense that their provenance cannot be easily identified;
- unclear relation between the probability distributions of the components and those of the composite model; and, as a consequence
- a lack of preservation of the probabilistic behaviour properties of a component when in combination with the behaviour of concurrently running components.

In this thesis we propose a novel formalism for probabilistic reasoning in such a way that individual component behaviour is guaranteed to be preserved over a composition. This approach achieves the goal by combining, and adding to, notions taken from Input-Output Probabilistic Automata and Interface Automata. This analysis leads to another contribution of this thesis.

> We present a formalism that supports compositional construction and validation of probabilistic models.

We can now refine Figure 1.1 as seen in Figure 1.2.

This new automata-like formalism, Probabilistic Interface Automata, is presented and discussed in Part II of this thesis.

1.2.3. Partial verification

Up to this point, we have discussed the underpinnings of a formalism that allows meaningful description and analysis of probabilistic and non-deterministic behaviour in a setting where models are constructed incrementally and through parallel composition. However, this is only half the work we need to do, since we still have not tackled the original problem we posed at the beginning of this introduction; that of providing quantitative information from a partial, failed model checking effort.

In this thesis, we propose measuring a partial model exploration as a random variable. In particular, the measure that we will assign to a property evaluated over a partial model is the expected value of this random variable for an arbitrary execution trace that traverses *outside* the explored state space. The rationale for this definition is that, since there were no property violations observed over the explored state space, we can safely assume that *every unexplored state* is a property-violating state. Defining the measure as this random variable, this value represents a lower bound on the actual value of the random variable expectation, if it were calculated over the complete model.

From a software engineering point of view, we will note that several reliability measures studied in the software reliability community can be characterised as random variables similar to the one we describe here. However, in contrast to software reliability approaches based on testing and simulation, here we aim to exploit the rigorous and extensive explorations that model checking tools and techniques are capable of, thanks to their fine-grained control of the model exploration strategy and efficient techniques for identifying already visited states.

This results in an additional contribution of this thesis

> We present a formalisation of what it means to perform a verification over a partial exploration of a system model; an analysis of why it makes sense to do so; the relationship between the results of a partial verification and a full one; and the expected benefits of performing partial vs. full verifications.

1.2.4. Efficient partial verification

Once the problem of verifying a partial state space is formalised and we know the relationship between the answer to the full model verification and the partial one, we set out to finally obtain the desired results.

However, both technical and practical reasons hamper this approach as we described it. From the technical point of view, the task of quantitatively annotating a partial state space exploration *a posteriori* is unfortunately not feasible. Suffice to say that, if our initial verification effort failed because of memory exhaustion, it is very unlikely that we still have enough memory to add this quantitative information to the mix.

From a practical point of view, and even if such annotation procedure were feasible, the obtained results are bound to not be very informative. The main reason for this is that model checkers are not really designed towards partial explorations. Their task is to finish the exploration in full, with no regard of how they get to it in the meantime. This means that partial explorations are generated with no rhyme nor reason; they are just as good as being completely random in their exploration order.

Analysis of this situation causes some questions to arise naturally.

- What if we have a way to yield many, different, partial state space explorations?
- Will the results obtained from verifying different partial state spaces be comparable?
- Which state spaces yield *better* results? (And what does *better* actually mean?)
- Is there a way to tell whether a partial state space will perform better than another (as in, they will provide more useful feedback results)?
- If so, can we consistently obtain *good* state spaces, in the sense that through their partial verification we can consistently obtain meaningful information?

In Chapter 6 we will show that not all partial state spaces are created equal. We argue that there is a relationship between the quality of the feedback results obtained by the analysis of a given partial state space, and how much of the relevant behaviour is captured by this state space. In this sense, the more relevant behaviour it captures, the better the results obtained.

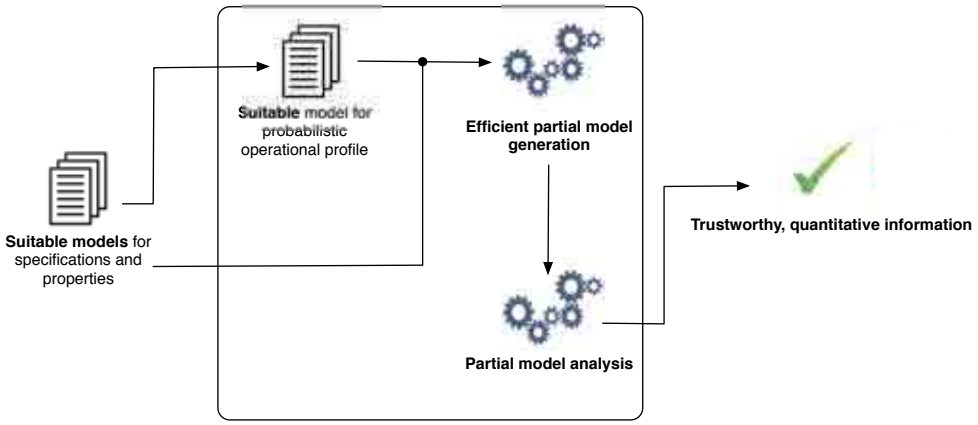


Figure 1.3: Detail of thesis contributions

We will also present a heuristic technique that combines probabilistically-guided simulations, invariant behaviour inference and model checking to obtain partial state spaces that *i*) are consistently small in relation to the full state space size (less than 5% of the projected full state space); and that *ii*) these partial state spaces, when subjected to the verification effort, consistently obtain meaningful results, that is, that they can be used to argue a reliability case for the whole system model.

Finally, we bring all of our results together. We show that Probabilistic Interface Automata are a natural and sensible way to model reactive software systems and their interaction with a probabilistic environment, even if the reactive system exhibits probabilistic behaviour itself. These models are amenable to automated verification techniques, but they can grow large enough to make whole system analysis infeasible. In such cases, our partial verification technique can be applied, including the automated generation of partial state spaces that have the potential of providing useful results.

We also study other approaches that aim at obtaining results while avoiding the construction of the complete model. In particular, we focus on statistical approaches, which are usually referred under the umbrella term of Monte Carlo verification. This techniques have minimal memory requirements, as they only need to keep a single execution path in memory. We discuss the characteristics of these approaches, and we compare the results obtained with our technique against the application of these statistical methods.

This wraps up the final contribution of this thesis

> We present an automated technique for exploring a system model in order to obtain a partial model such that it attempts to maximise the information conveyed by the partial verification approach.

Further, we validate this approach through several case studies and compare the results against established approaches. The blown-up contributions of this thesis are captured by Figure 1.3.

1.2.5. Contributions of this thesis

This thesis puts forth the following contributions, which condense those we mentioned along this introduction.

- In summary, this thesis provides an approach for obtaining quantitative information on properties that cannot be verified in general using state-of-the-art approaches such as model checking or Monte Carlo statistical verification. Further, this quantitative information is meaningful in terms of the property being verified in first place.
- In order to specify probabilistic behaviour, we present Probabilistic Interface Automata, a formal model suitable to be used in an incremental, compositional model construction setting.
- We formalise the problem of partial state space probabilistic verification, and its relationship with the probabilistic model checking of full models. We show that verification over partial explorations provides meaningful bounds on the expected results over full models.
- Finally, we present an automated approach to efficiently obtain partial state spaces that consistently provide better results than both full state space model checking and Monte Carlo approaches.

1.2.6. Roadmap

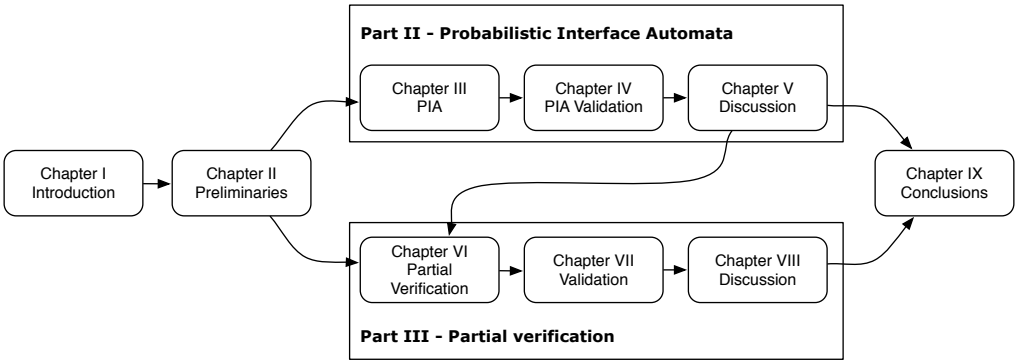


Figure 1.4: Organization of this thesis.

This thesis has two distinct, mostly independent parts. As a result, it can be read in three different ways depending on the interests of the reader. These three ways are depicted in Figure 1.4.

The first way to read it is to simply follow the thesis sequentially. In this way, first a common background is presented in Chapter 2, and then we progress to Part II where we present our modelling formalism, Probabilistic Interface Automata, and discuss related probabilistic modelling work. This first part is based on [PBU09] where Probabilistic Interface Automata were studied as a probabilistic model for environments interacting with non-deterministic system models.

After presenting these automata, we move on to our partial verification framework, validate our approach through some case studies from the literature, and discuss similar and related verification approaches. This second part is based on ideas first presented in [PBU10] and then expanded in [PBU13]. Finally, we offer our conclusions and outlook on the subjects tackled by this thesis.

Two alternative ways of reading the thesis are offered for those readers that are only interested in the formal aspects of the modelling formalism; or those that mainly want to learn about our partial verification approach. These readers may choose to read either Part II or Part III only, without sacrifice of a coherent reading. If the

reader would choose to approach Part III only, some backtracking to background definitions may be needed; whenever such back references are necessary, these will be clearly identified in the text.

CHAPTER 2

Preliminaries

This chapter summarises most of the concepts and notation that we will use throughout this thesis. First, we will introduce and recall some definitions related to probability theory. Later on we will define labelled transition systems and particular extensions of them, both non-deterministic as well as probabilistic.

2.1. An introduction to probability theory

In this section we provide a summary of notions from measure and probability theory that we will use throughout the thesis. Readers experienced with the subject may skim through this chapter in order to familiarise themselves with the notation we employ for different concepts. These definitions are by no means a complete introduction to probability theory, and they are kept simplified for the benefit of the casual reader. Further, we only focus on discrete probability spaces, since this thesis does not deal with continuous probabilistic processes. The interested reader is referred to the classic introduction by Feller [Fel08], from which we borrowed the definitions below.

Definition 2.1 (Probability space)**.** *A* probability space *is a triple* $< \Omega, 2^{\Omega}, \mu >$ *where*

- Ω *is a countable set called the* sample space*;*
- 2^{Ω} *is the powerset of* Ω*, and its elements are called* events*; and*
- $\mu : 2^{\Omega} \to [0, 1]$ *is a function such that*
  - $\mu(\emptyset) = 0$*;*
  - $\mu(\Omega) = 1$*; and*
  - *given* $(\omega_i), i \in \mathbb{N}$ *a sequence of elements in* 2^{Ω} *such that they are all disjoint pairwise, then* $\mu(\bigcup_i \omega_i) = \sum_i \mu(\omega_i)$*.*

The function μ is usually called a *probability measure* or, more often, a *probability distribution*. Given a subset ω of a sample space Ω, $\mu(\omega)$ is referred as the *measure* of ω.

Definition 2.2 (Support set)**.** *Given a distribution μ on a sample set Ω, the* support set *of μ, noted $supp(\mu)$ is the smallest closed set $S \subseteq \Omega$ such that its complement with respect to Ω has measure zero.*

There is a particular case which occurs when there is a single element ω in Ω such that $\mu(\omega) = 1$, and $\mu(\omega') = 0$ for any other ω'. In that particular case, we say μ is a *Dirac* distribution.

Definition 2.3 (Product of probability spaces)**.** *Let $P_1 =< \Omega_1, 2^{\Omega_1}, \mu_1 >$ and $P_2 =< \Omega_2, 2^{\Omega_2}, \mu_2 >$ be two probability spaces. We can then consider the* product probability space *defined as $P_1 \otimes P_2 =< \Omega_1 \times \Omega_2, 2^{\Omega_1 \times \Omega_2}, \mu_1 \otimes \mu_2 >$, where for each $\omega_1 \times \omega_2 \in 2^{\Omega_1 \times \Omega_2}$ it holds that $\mu_1 \otimes \mu_2(\omega_1 \times \omega_2) = \mu_1(\omega_1) \times \mu_2(\omega_2)$.*

Sometimes we will be interested not only in the probability of an event ω, but also on the value of functions over these events. For example, suppose the set Ω depicts the possible outcomes of flipping ten coins. A function of interest, for example, could be the one which calculates the number of *heads* for a given event. The value of such a function is related to the probability space where it is applied. This gives rise to the notion of *random variables*.

Definition 2.4 (Random variable)**.** *Let $< \Omega_1, 2^{\Omega_1}, \mu_1 >$ and $< \Omega_2, 2^{\Omega_2}, \mu_2 >$ be two probabilistic spaces. A* random variable *X is a function $X : \Omega_1 \to \Omega_2$. X is said to be evaluated on* outcomes *Ω_1 and have* range *Ω_2.*

In the context of this thesis, Ω_2 will always be $\mathbb{R}$. In this sense, the idea of a random variable is to convey some numerical value to an outcome ω.

Definition 2.5 (Expected value of a random variable)**.** *Let $< \Omega, 2^{\Omega}, \mu >$ be a probability space and X a random variable on $\mathbb{R}$ (that is, $X : \Omega \to \mathbb{R}$). The* expected value *of X, noted $E[X]$ or $\overline{X}$, is the weighted average of X based on μ. That is, $\overline{X} = \sum_{\omega \in \Omega} \mu(\omega) \times X(\omega)$.*

2.2. Formalisms for system modelling

In the course of this thesis we are interested in the modelling and verification of properties over *reactive* systems. That is, we focus on systems that, rather than perform a batch task without interference from outside entities, actually interact with their environment by reacting to some events and providing some of its own. These events that the system generates, in turn, elicit responses on the entities outside the reactive system. In that sense, these outside entities can be regarded as reactive systems themselves.

2.2.1. Non-deterministic models

Labelled transition systems are a widespread form of modelling such systems. One of the main advantages of these models is that, along with a semantics of asynchronous execution but synchronisation on shared events, they allow for incremental modelling of systems by the way of parallel composition.

We will start first by defining models that do not exhibit any probabilistic behaviour. We will refer to these models by the umbrella term of *non-deterministic* model, since the choice between different reactions to a given event will be resolved by choosing one without any quantitative information about this choice (i.e., the choice is non-deterministic).

Definition 2.6 (Labelled Transition System [Kel76])**.** *A Labelled Transition System (LTS) is a tuple* $M =< S, S_0, A, R >$ *where*

- S *is a finite set of* states*;*
- $S_0 \subseteq S$ *is the set of* initial states*. Without loss of generality, we usually consider this set to be unitary and note* s_0 *the unique initial state;*
- A *is a finite set of event labels, also usually referred as* action labels*; and*
- $R \subseteq S \times A \times S$ *is the* transition relation *that specifies, for each state, to which state the system evolves as a result of a given event. Since* R *is a relation, there could exist none, or several different, transition destinations for a same action.*

In order to establish a proper protocol of interaction between components that run concurrently and synchronise through shared actions, it is useful to segregate the action set A into three mutually disjoint sets. These sets will represent the *input* actions that a component reacts to, the *output* actions that it generates, and the *hidden* or *internal* actions that represent internal computation and are not visible from outside the component.

This thesis bases its approach and results on the Interface Automata formalism [HdA01]. Later on we will provide a discussion on this choice, as well as considering other suitable formalisms that could have been used in its place.

Definition 2.7 (Interface Automata [HdA01])**.** *An* Interface Automaton *is a tuple* $P =< S_P, s_P^0, A_P^I, A_P^O, A_P^H, R_P >$ *where:*

- S_P *is a finite set of states.*
- $s_P^0 \in S_P$ *is a distinct* initial *state.*
- A_P^I, A_P^O, A_P^H *are finite and mutually disjoint sets of* input*,* output *and* hidden *actions respectively. We denote the set of all actions* $A_P = A_P^I \cup A_P^O \cup A_P^H$*.*
- $R_P \subseteq S_P \times A_P \times S_P$ *is the transition relation.*

We will write $A_P^I(s)$, $A_P^O(s)$ and $A_P^H(s)$ for a state $s \in S_P$ to denote the subset of actions in A_P^I, A_P^O and A_P^H, respectively, that are *enabled* at s. An action $a \in A_P$ is said to be enabled at state $s \in S_P$ if there exists $t \in S_P$ such that $(s, a, t) \in R_P$. Alternatively, we may say that the transition (s, a, t) itself is enabled if the previous condition holds. Analogously, we denote $A_P(s)$ the subset of actions enabled at state s, regardless of them being input, output or hidden actions. Without loss of generality, we require that for each state $s \in S_P$, there exists $s' \in S_P, a \in A_P$ such that $(s, a, s') \in R_P$.

In essence, an Interface Automaton is a labelled transition system where its action set has been further subdivided to distinguish the input, output and hidden actions. As we will see, this does not make a syntactic difference, but it does semantically. Also, note that we have reduced the original set of initial states to a single one without loss of generality.

Definition 2.8 (Execution fragments and executions)**.** *An* execution fragment *of an Interface Automaton* P *is a (possibly infinite) sequence* $\alpha = s_0 a_1 s_1 a_2 s_2 \ldots$ *of alternating states and action labels. Execution fragments always start with a state and, if finite, also end with a state. Each subsequence* $s_i a_{i+1} s_{i+1}$ *within an execution fragment of* P *is such that* $(s_i, a_{i+1}, s_{i+1}) \in R_P$*.*

Given an execution fragment α, first(α) denotes the first state of the fragment, while tail(α) denote the execution fragment from its second state. tail(α) might be empty if α is finite and consists of only one state. If α is finite, last(α) denotes its final state.

An execution *of an Interface Automaton P is an execution fragment α of P such that first(α) $= s^0_P$, the initial state of P. As executions are execution fragments themselves, they can also be finite or infinite.*

We will also note *fragments*(P) and *fragments*$*$(P) to denote the set of execution fragments of P and the set of *finite* execution fragments of P, respectively. Accordingly, we will note *execs*(P) and *execs*$*$(P) for the set of executions and finite executions of P. For convenience, we also define *length* : *fragments*(P) $\to \mathbb{N} \cup \infty$ to be the number of states traversed by the execution fragment. We also define projectors α^s_i and α^a_i that return the i-th state and i-th transition label respectively. Note that α^s_i is defined from 0 through *length*(α) $- 1$, while α^a_i is defined from 1 through *length*(α) $- 1$.

The notation $\alpha \leq \alpha'$ will be used to indicate that the execution fragment α is a *prefix* of execution fragment α'; that is, for each $0 \leq i \leq$ *length*(α) $- 1$, $\alpha^s_i = \alpha'^s_i$ and for each $1 \leq j \leq$ *length*(α) $- 1$, $\alpha^a_j = \alpha'^a_j$. Accordingly, *suffix*(α, i) is defined for every $i <$ *length*(α) and obtains the execution fragment that results of dropping the first i states and actions from an execution fragment. Therefore, for an execution fragment $\alpha = s_0 a_1 s_1 a_2 s_2 a_3 s_3 \ldots$, *suffix*($\alpha, 0$) $= \alpha$, *suffix*($\alpha, 1$) $= s_1 a_2 s_2 a_3 s_3 \ldots$ and so on.

As additional notation, we will note the existence of a finite execution fragment α from s_0 to s_n by $s_0 \xrightarrow{\alpha} s_n$.

Parallel composition

As we discussed earlier, we will use Interface Automata to build more complex system models in an incremental fashion. The notion of action segregation in Interface Automata allows for establishing a synchronising communications protocol between components, as output actions on one component will synchronise with input actions on another one. The notion of *composability* of Interface Automata captures this idea formally.

Definition 2.9 (Composability [HdA01])**.** *Let P and Q be two Interface Automata. We say P and Q are* composable *if it holds simultaneously that*

- $A^H_P \cap A_Q = \emptyset$;
- $A_P \cap A^H_Q = \emptyset$;
- $A^I_P \cap A^I_Q = \emptyset$; *and*
- $A^O_P \cap A^O_Q = \emptyset$

Furthermore, when discussing the interaction of two Interface Automata P and Q, it is usual to refer to its shared set of actions, *Shared*$(P, Q) = A_P \cap A_Q$. Note that if P and Q are composable, then *Shared*$(P, Q) = (A^I_P \cap A^O_Q) \cup (A^O_P \cap A^I_Q)$. We recall the definition of Interface Automata product and illegal states.

Definition 2.10 (Product [HdA01])**.** *Let P and Q be two composable Interface Automata. Their product $P \otimes Q$ is another Interface Automaton such that*

- *Its set of states $S_{P\otimes Q}$ is defined as $S_P \times S_Q$;*

- *its initial state is* $s^0_{P\otimes Q} = (s^0_P, s^0_Q)$*; and*
- *its segregated action sets are* $A^I_{P\otimes Q} = (A^I_P \cup A^I_Q) \setminus \mathit{Shared}(P,Q)$*;* $A^O_{P\otimes Q} = (A^O_P \cup A^O_Q) \setminus \mathit{Shared}(P,Q)$ *and* $A^H_{P\otimes Q} = A^H_P \cup A^H_Q \cup \mathit{Shared}(P,Q)$*.*

Finally, its transition relation $R_{P\otimes Q}$ *is defined by the set*

$$\begin{aligned}
&\{((s,t),a,(s',t)) \text{ such that } (s,a,s') \in R_P \wedge \\
&\quad t \in S_Q \wedge a \notin \mathit{Shared}(P,Q)\} \cup \\
&\{((s,t),a,(s,t')) \text{ such that } (t,a,t') \in R_Q \wedge \\
&\quad s \in S_P \wedge a \notin \mathit{Shared}(P,Q)\} \cup \\
&\{((s,t),a,(s',t')) \text{ such that } a \in \mathit{Shared}(P,Q) \wedge \\
&\quad (s,a,s') \in R_P \wedge (t,a,t') \in R_Q\}
\end{aligned}$$

Since the behaviour of a composite Interface Automaton is directly related to the behaviour of each of its components, there is a close relationship between the executions (and executions fragments) of a composite system, and those of its components. However, this depends on the semantics of the interface. The action segregation introduced in the definition of Interface Automata is essentially a description language tool. Although it has no bearing in the previous formal definitions, it introduces the notion of *illegal composition states*. Intuitively, a composition state will be regarded as illegal if, somehow, it violates the enabledness of the intended actions of each component.

Definition 2.11 (Illegal states [HdA01])**.** *Given two composable Interface Automata* P *and* Q*, their product's illegal states are defined by the set* $\mathit{Illegal}(P,Q) \subseteq S_P \times S_Q$*. For any* $s \in S_P$*,* $q \in S_Q$*,* $(s,q) \in \mathit{Illegal}(P,Q)$ *if* $\exists a \in \mathit{Shared}(P,Q)$ *such that* $a \in A^O_P(s) \wedge a \notin A^I_Q(q)$*, or conversely* $\exists a \in \mathit{Shared}(P,Q)$ *such that* $a \notin A^I_P(s) \wedge a \in A^O_Q(q)$*.*

Informally, the idea behind illegal states is that for a composition to be legal, component systems should not be able to block each other's enabled *output* actions. We will abuse notation and say that the product $P \otimes Q$ of two Interface Automata P and Q is *legal* if the product has no *reachable* illegal states.

The notions of composability and illegal states make it possible to define what a *valid environment* for a given Interface Automaton is.

Definition 2.12 (Valid environment [HdA01])**.** *Given an Interface Automaton* P*, another non-empty Interface Automaton* Q *is a* valid environment *for* P *if all of the following hold simultaneously:*

- P *and* Q *are composable;*
- $A^I_Q = A^O_P$*; and no state in* $\mathit{Illegal}(P,Q)$ *is reachable in* $P \otimes Q$*.*

Non-determinism and schedulers

It is important to note that the distinct execution fragments generated by an Interface Automata depend on how the choice between different transitions is resolved. That is, whenever two or more actions can be chosen in a state, the choice of which action to take is left unspecified, and can only be resolved by an external agent. In order to distinguish this choice from the probabilistic choices that will appear later in this thesis, we will refer to these choices as *non-deterministic* choices. Note that this is slightly different from a common meaning of *non-determinism* which is limited to the choice between different transitions with the same label. Throughout

this thesis we use the term *non-deterministic* to distinguish those choices that are *not probabilistic* in nature.

In order to characterise this external agent, and thus the different non-deterministic choices and the execution fragments that they induce, we will introduce the notion of a *scheduler*.

Definition 2.13 (Scheduler)**.** *A* scheduler σ *for an Interface Automaton* $P =< S_P, s^0_P, A_P, R_P >$ *(also called an* adversary*) is a total function* $\sigma : execs*(P) \to R_P$*, such that* $\sigma(\alpha)$ *is a transition starting from* $last(\alpha)$*; and whenever* $\sigma(\alpha) = (last(\alpha), a, s)$ *it must be that* $(last(\alpha), a, s) \in R_P$*. The notation* $Sched(P)$ *refers to the set of all possible schedulers for the automaton* P*; while* $\sigma(\alpha)_a$ *and* $\sigma(\alpha)_s$ *refer to the scheduled action and destination state given an execution* α*, respectively.*

The idea behind schedulers is that they drive the execution of the automaton by resolving all possible non-determinism. As such, they restrict the set of possible execution fragments. Extending this notion, a set of schedulers defines a set of possible executions and execution fragments.

Definition 2.14 (Scheduler-generated executions)**.** *Given an Interface Automaton* P*, a scheduler* σ_P *and an execution* $\alpha \in execs(P)$*, we say that* σ_P generates α *over* P *if and only if for each* $0 \leq i < length(\alpha)$ *it holds that* $\sigma_P(\alpha^s_0 \alpha^a_1 \ldots \alpha^s_i) = (\alpha^a_{i+1}, \alpha^s_{i+1})$*.*

Note that once a scheduler σ is set for an Interface Automaton P, this scheduler eliminates all possible branching behaviour. That is, it generates a single infinite execution fragment, along with its infinite set of finite prefixes.

Some schedulers will not be very useful to our approach, as they may model invalid behaviours. In particular, we are interested in schedulers that are *fair* in their choices of non-determinism resolution, as they have desirable properties which will be discussed later. The following definitions deal with our requirements for fairness, which have been adapted from [BGC09, Var85, BK98].

Definition 2.15 (Fair executions)**.** *Let* α *be an infinite execution over an Interface Automaton* P*. For each* $s \in S_P$*, let* $Traversals(\alpha, s) = \{i \in \mathbb{N}_0 \cdot \alpha^s_i = s\}$*, that is* $Traversals(\alpha, s)$ *denotes the indexes in* α *where state* s *is traversed. Similarly, define* $Traversals(\alpha, (s, a, s'))$ *to be the indexes in* α *where the transition* (s, a, s') *is taken.*

We say that α *is a* fair *execution if for each* $s \in S_P$ *such that* $Traversals(\alpha, s)$ *is an* infinite *set, it holds that whenever* (s, a, s') *is an enabled transition from* s *(that is,* $(s, a, s') \in R_P$*), then the set* $Traversals(\alpha, (s, a, s'))$ *is also infinite.*

Informally, an execution is *fair* if every time that it passes through a state t infinitely often, then it also progresses over each of its enabled transitions infinitely often. In other words, whenever a transition is enabled and the execution has the opportunity to take it, a fair execution cannot avoid taking it indefinitely. We will extend this notion of fairness to schedulers.

Definition 2.16 (Strictly fair schedulers [CGP99])**.** *A scheduler* σ *is* strictly fair *(also called* strong fair*) if the infinite execution it generates is itself* fair*.*

The reasons behind the choice of words on defining schedulers as *strictly* fair in Definition 2.16 will be made more clear once we examine schedulers for probabilistic models.

Logics for property description

Several temporal logics have been put forth for reasoning about the protocols described by automata-like formalisms. As we will see later when we discuss property preservation, we need to preserve the branching structure of components within the composition. We will therefore express these behaviour properties with the logic CTL (*Computational Tree Logic*) [EC82], or some variants of it. ACTL [DV90] (not to be confused with the universal fragment of CTL) in particular is a temporal logic equivalent to CTL. The main difference is that, while CTL focuses its predicates on *states*, ACTL does so on the set of *actions*. ACTL will become useful to us, as it allows us to express directly the restrictions that pertain to the availability of actions for synchronisation. This will allow us to expand the notion of composability when we present our Probabilistic Interface Automata formalism in Part II.

Definition 2.17 (ACTL Syntax [DV90])**.** *The set of ACTL formulae is defined as the smallest set of* state formulae *such that*

- $True$ *is a state formula;*
- *if* ϕ_1 *and* ϕ_2 *are state formulae, then* $\neg\phi_1$ *and* $\phi_1 \wedge \phi_2$ *are also state formulae;*
- *if* ψ *is a* path formula*, then* $\neg\psi$ *is also a path formula;*
- *if* ψ *is a path formula then* $\exists\psi$ *is a state formula;*
- *if* ϕ_1 *and* ϕ_2 *are state formulae and* a *is an action label, then* $X_a\phi_1$*,* $\phi_1 \mathcal{U} \phi_2$ *and* $\phi_1 \mathcal{U}_w \phi_2$ *are path formulae.*

Definition 2.18 (ACTL Semantics [DV90])**.** *Let* $M =< S_M, s^0_M, A^I_M, A^O_M, A^H_P, R_M >$ *be an Interface Automaton. The semantics of an ACTL formula are given by a* satisfaction relation*, which is defined for* M *over execution fragments* $\alpha \in fragments(M)$ *for path formulae* ψ *(noted* $M, \alpha \models \psi$*), and over states* $s \in S_M$ *for state formulae* ϕ *(noted* $M, s \models \phi$*). The satisfaction relation is defined inductively as follows, where* ϕ_1, ϕ_2 *denote state formulae and* ψ *denotes a path formula, and* $a \in A_M$*:*

$$
\begin{array}{lcl}
M, s \models True & & always\ holds \\
M, s \models \neg\phi & \Leftrightarrow & \neg(M, s \models \phi) \\
M, s \models \phi_1 \wedge \phi_2 & \Leftrightarrow & M, s \models \phi_1 \wedge M, s \models \phi_2 \\
M, s \models \exists\psi & \Leftrightarrow & \exists\alpha \in fragments(M)\ such\ that \\
 & & \alpha^s_0 = s \wedge \alpha \models \psi \\
M, \alpha \models \neg\psi & \Leftrightarrow & \neg(M, \alpha \models \psi) \\
M, \alpha \models X_a\phi & \Leftrightarrow & length(\alpha) > 1 \wedge \alpha^a_0 = a \wedge \\
 & & M, \alpha^s_1 \models \phi \\
M, \alpha \models \phi_1 \mathcal{U} \phi_2 & \Leftrightarrow & (\exists 0 \leq j < length(\alpha))(\forall 0 \leq i < j) \\
 & & M, \alpha^s_i \models \phi_1 \wedge M, \alpha^s_j \models \phi_2 \\
M, \alpha \models \phi_1 \mathcal{U}_w \phi_2 & \Leftrightarrow & (M, \alpha \models \phi_1 \mathcal{U} \phi_2) \vee \\
 & & (\forall 0 \leq i < length(\alpha)) M, \alpha^s_i \models \phi_1
\end{array}
$$

We will abuse notation and, given a finite set of actions A, note $X_A\phi$ as an equivalent to $\bigvee_{a \in A} X_a\phi$. Also, we can further refine the satisfiability notion to ask whether a formula ϕ is satisfiable by an Interface Automaton M when under a given scheduler σ. The satisfiability semantics are kept almost the same, except that whenever we need to check for fragments in $fragments(M)$, we must restrict them to those *generated* by σ.

2.2.2. Probabilistic models

The probabilistic models that we will work within this thesis are automata-like, so they are essentially LTSs where the transition relation is enriched with probabilistic information. In order to convey these probabilistic semantics, we will use as foundation a well-known probabilistic formalism, that of Segala's Simple Probabilistic Automata [SL95, Seg95]. Again, we will discuss on this choice and other alternatives when we show the distinct problems of introducing probabilities in our setting, and when we survey related work.

As we will see, SPAs extend classic LTSs by modifying the transitions so that they no longer reach a single state, but a probabilistic distribution over a set of destination states instead.

Definition 2.19 (Segala's Simple Probabilistic Automaton (SPA))**.** *A* Simple Probabilistic Automaton *is defined by a tuple* $M =< S_M, s^0_M, A_M, R_M >$ *where*

- S_M *is a finite set of states.*
- $s^0_M \in S_M$ *is a distinct initial state.*
- A_M *is a finite set of actions.*
- $R_M \subseteq S_M \times A_M \times D(S_M)$ *is a transition relation, where* $D(S_M)$ *denotes the set of* probabilistic distributions *over the sample set of states* S_M*. Note that since* S_M *is finite,* $D(S_M)$ *turns out to be a discrete distribution.*

We will note $R_M(s)$ to denote the set of all transitions that originate on state s, that is, those tuples in R_M where the first component is s. Similarly, we will note $R_M(s, a)$ to note the set of transitions originating in s through action a. For convenience and without loss of generality, we will assume that for all states $s \in S_M$, the transition relation is such that $R_M(s) \neq \emptyset$ [dA97].

In a manner similar to other automata-based behaviour description formalisms, Simple Probabilistic Automata can be constructed compositionally as the product of other, smaller Simple Probabilistic Automata.

Definition 2.20 (Simple Probabilistic Automata product [SL95])**.** *Let* $M_1 =< S_1, s^0_1, A_1, R_1 >$ *and* $M_2 =< S_2, s^0_2, A_2, R_2 >$ *be two Simple Probabilistic Automata. Their product* $M_1 \otimes M_2$ *is defined to be another Simple Probabilistic Automaton* $M =< S_{M_1 \otimes M_2}, s^0_{M_1 \otimes M_2}, A_{M_1 \otimes M_2}, R_{M_1 \otimes M_2} >$*, such that*

- $S_{M_1 \otimes M_2} = (S_1 \times S_2)$
- $s^0_{M_1 \otimes M_2} = s^0_1, s^0_2$
- $A_{M_1 \otimes M_2} = A_1 \cup A_2$
- *given* $(s, t) \in S_1 \otimes S_2$*,* $a \in A_1 \cup A_2$ *and* $\delta \in D(S_{M_1 \otimes M_2})$*,* $R_{M_1 \otimes M_2}$ *is such that* $((s, t), a, \delta) \in R_{M_1 \otimes M_2}$ *if and only if any of the following is satisfied:*
  1. $a \in A_1 \wedge a \notin A_2 \wedge \forall s' \in S_1(\exists \delta_1 \in D(S_1)$ *such that* $(s, a, \delta_1) \in R_1 \wedge \forall s' \in S_1, \delta((s', t)) = \delta_1(s'))$
  2. $a \in A_2 \wedge a \notin A_1 \wedge \forall t' \in S_2(\exists \delta_2 \in D(S_2)$ *such that* $(t, a, \delta_2) \in R_2 \wedge \forall t' \in S_2, \delta((s, t')) = \delta_2(t'))$
  3. $a \in A_1 \cap A_2 \wedge \exists \delta_1 \in R_1(s, a) \wedge \exists \delta_2 \in R_2(t, a)$ *such that* $\forall s' \in S_1, t' \in S_2, \delta((s', t')) = \delta_1(s') \times \delta_2(t')$.

As is the case for Interface Automata, SPAs are composed through an asynchronous product, but synchronising on shared actions. This distinction is made clear when defining the transition relation for the product SPA. Clauses 1 and 2 state that, whenever an action is not shared by both processes, the possible distributions governing transitions in the product are exactly those that come from each component process. Clause 3 describes the synchronising nature of the Simple Probabilistic Automata product. The distributions for transitions where the action label is shared are computed as the product of the distributions for each of the components. Note that, when composing states from different components, if at any of these states the shared action is not enabled (i.e., the state does not provide an outgoing transition through the shared action), then no distribution is present and the product cannot be computed. In that case, the product state does not have an outgoing transition on the shared action—it does not synchronise.

The definitions for execution fragments and complete executions still apply to Simple Probabilistic Automata, as we are still interested in the possible traces of the Simple Probabilistic Automaton.

Definition 2.21 (SPAs' execution fragments and executions)**.** *An* execution fragment *of a Simple Probabilistic Automaton M is a (possibly infinite) sequence $\alpha = s_0(a_1, p_1)s_1(a_2, p_2)s_2 \ldots$ of alternating states and transitions, where these transitions are annotated by their governing action and associated probability. Execution fragments always start with a state and, if finite, also end with a state. Each sequence $s_i(a_{i+1}, p_{i+1})s_{i+1}$ within an execution fragment of M is such that there exists a probabilistic distribution δ such that $(s_i, a, \delta) \in R_P$, and $\delta(s_{i+1}) = p_{i+1}$.*

Given an execution fragment α, $first(\alpha)$ denotes the first state of the fragment, while $tail(\alpha)$ denotes the execution fragment from its second state. $tail(\alpha)$ might be empty if α is finite and consists of only one state. If α is finite, $last(\alpha)$ denotes its final state.

An execution *of a Simple Probabilistic Automaton M is an execution fragment α of M such that $first(\alpha) = s^0_M$, the initial state of the automaton. As executions are execution fragments themselves, they can also be finite or infinite.*

As was the case for Interface Automata, we will also note $fragments(M)$ and $fragments*(M)$ to denote the set of execution fragments of M and the set of *finite* execution fragments of M, respectively. Additionally, we will note $execs(M)$ and $execs*(M)$ for the set of executions and finite executions of M. We also define $length : fragments(M) \to \mathbb{N} \cup \infty$ to be the number of states traversed by the execution fragment. For additional convenience, we define projectors α^s_i, α^a_i and α^p_i that return the i-th state, i-th transition label and i-th associated probability respectively. Note that α^s_i is defined from 0 through $length(\alpha) - 1$, while α^a_i and α^p_i are defined from 1 through $length(\alpha) - 1$. Finally, we will note $\alpha \leq \alpha'$ to indicate that the execution fragment α is a finite *prefix* of execution fragment α'. Again, $suffix(\alpha, i)$ is defined for every $i < length(\alpha)$ and obtains the execution fragment that results of dropping the first i states and probability-action pairs from an execution fragment. Therefore, for an execution fragment $\alpha = s_0(a_1, p_1)s_1(a_2, p_2)s_2(a_3, p_3)s_3 \ldots$, $suffix(\alpha, 0) = \alpha$, $suffix(\alpha, 1) = s_1(a_2, p_2)s_2(a_3, p_3)s_3 \ldots$ and so on.

As additional notation, we will note the existence of a finite execution fragment α from s_0 to s_n by $s_0 \xrightarrow{\alpha} s_n$. Note the notation is different to quickly distinguish probabilistic execution fragments from purely non-deterministic ones.

The notion of schedulers for resolving non-determinism is also preserved, but note that instead of scheduling an action and a destination state, it schedules a distribution on destination states instead.

Definition 2.22 (Scheduler for Simple Probabilistic Automata)**.** *A* scheduler σ *for a Simple Probabilistic Automaton* $M =< S_M, s^0_M, A_M, R_M >$ *(also called an* adversary*) is a total function* $\sigma : execs*(M) \to A_M \times D(S_M)$*, such that if* $\sigma(\alpha) = (a, \delta)$ *it must be that* $(last(\alpha), a, \delta) \in R_M$*.*

It is noteworthy, however, that resolving non-determinism via a scheduler for an SPA does not, as was the case for Interface Automata, produce a unique execution. Rather, resolving non-determinism induces a fully probabilistic process, specifically a Discrete Time Markov Chain (DTMC) which, in turn induces a set of execution fragments. For more insight on these probabilistic processes the reader may refer to [Fel08, Kul09]. We will need, however, a couple of concepts related to the DTMCs obtained by fixing a scheduler. The following definitions on DTMCs have been lifted from [Kul09].

Definition 2.23 (Irreducibility)**.** *Let A be a Discrete Time Markov Chain (in particular, A could have been obtained as the result of fixing a scheduler* σ *for a SPA M). The DTMC A is said to be* irreducible *if, for every pair of states* s, s' *in its state space, there exists an execution fragment* α *such that* $s \xrightarrow{\alpha} s'$*.*

In other words, a DTMC is irreducible if it is possible to get from any state to any other state.

Definition 2.24 (Periodicity)**.** *A state s in a Discrete Time Markov Chain A is said to have a* period k *if*

$$k = gcd\{length(\alpha) \cdot \alpha \in fragments*(A) \wedge length(\alpha) > 0 \wedge s \xrightarrow{\alpha} s\}$$

where gcd denotes the greatest common divisor.

Put more plainly, a state s on a DTMC A has period k if every execution fragment that starts at s and traverses back to s has a length that is a multiple of k. If $k = 1$, s is said to be *aperiodic*. If *every* state s of A is aperiodic, then the whole DTMC A is said to be aperiodic.

Theorem 2.1 (Periodicity and reachability)**.** *Let A be a DTMC and* s, s' *two states in its state space such that there exist* α, $\alpha' \in fragments*(A)$ *and* $s \xrightarrow{\alpha} s'$, $s' \xrightarrow{\alpha'} s$*. Then, if s has period k and* s' *has period* k'*, it holds that* $k = k'$*.*

Corollary 2.1 (Periodicity and irreducibility)**.** *If A is an irreducible DTMC, then all of its states have the same period.*

Corollary 2.2 (Aperiodicity check on irreducible DTMCs)**.** *Let A be an irreducible DTMC. Then, A is aperiodic if and only if any of its states is aperiodic.*

Definition 2.25 (Ergodicity)**.** *If a Discrete Time Markov Chain A is irreducible and aperiodic, it is said to be* ergodic[1]*.*

Both irreducibility and aperiodicity can be easily tested in the underlying graph of the DTMC [JS99]. Ergodic DTMCs have desirable properties, which we will turn to further in the thesis. For now, we continue introducing other concepts related to schedulers and the *measures* they induce.

The combination of a scheduler σ and an SPA M defines a probability measure δ on the σ-algebra generated by the set of execution fragments defined by the scheduler

[1]The knowledgeable reader may know that this is not exactly so. Irreducibility and aperiodicity are sufficient for ergodicity, but not necessary. However, we will not need that level of detail. The exact definition of ergodicity requires additional definitions and can be found in [Kul09].

σ. Note that the sample set of execution fragments is neither finite nor countable, therefore distributions over this sample space will not be discrete. We need to resort to measuring by using *cylinder sets* (also called *cones* in the literature) of execution fragments.

Definition 2.26 (Cylinders and probability measure [Seg95])**.** *Given a finite execution fragment α of an SPA M, the cylinder of α is the set of execution fragments $C_\alpha = \alpha' \in fragments(M) \cdot \alpha \leq \alpha'$. The measure of a cylinder C_α under scheduler σ is defined as*

$$\delta(C_\alpha, M, \sigma) = \prod_{i=1}^{length(\alpha)} IsSched(\sigma, \alpha, i-1, \alpha_i^a) \times \delta_{Sched}(\sigma, \alpha, i-1)(\alpha_i^s)$$

where $\delta_{Sched} : Sched(M) \times fragments*(M) \times \mathbb{N} \to D(S_M)$, and $IsSched : Sched(M) \times fragments*(M) \times \mathbb{N} \times A_M \to (0,1)$ are such that $\delta_{Sched}(\sigma, \alpha, n) = \sigma(\alpha_0 \ldots \alpha_n)_\delta$ and

$$IsSched(\sigma, \alpha, n, a) = \begin{cases} 1 & \text{if } \sigma(\alpha_0 \ldots \alpha_n)_a = a \\ 0 & \text{otherwise} \end{cases}$$

In other words, δ_{Sched} obtains the distribution corresponding to the next scheduled transition, while IsSched checks whether in fact a is the next scheduled action.

Cylinder measure as defined in Definition 2.26 can easily be extended for sets of non-overlapping cylinders. Given a SPA M, a scheduler σ, and a set Γ of finite execution fragments such that for every $\alpha_i, \alpha_j \in \Gamma$ neither is a prefix of the other, we can define the measure of the set Γ (noted $\delta(\Gamma, M, \sigma)$) as follows:

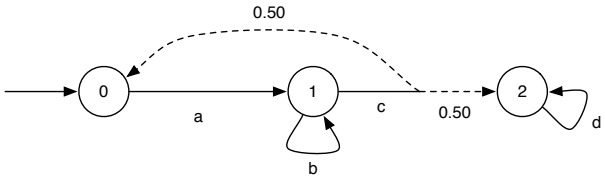
$$\delta(\Gamma, M, \sigma) = \sum_{\alpha \in \Gamma} \delta(C_\alpha, M, \sigma)$$

The notion of cylinders is essential for the definition of the σ-algebra underlying SPAs, since it gives us a way to *measure* sets of traces, where nevertheless each individual trace has zero probability. As we will see later, this concept will have a strong relation with the logics we will employ to reason about SPA behaviour.

With the leverage of the previous definitions, we can characterise the set of execution fragments generated by a scheduler σ on an SPA M.

Definition 2.27 (Simple Probabilistic Automaton scheduled fragments)**.** *Let M be a Simple Probabilistic Automaton, and σ a scheduler for M. The set of* scheduled execution fragments *of M through σ is the set of execution fragments $fragments(M, \sigma) \subseteq fragments(M)$ such that $\alpha \in fragments(M, \sigma) \Leftrightarrow (\forall \alpha' \in fragments*(M) \cdot \alpha' \leq \alpha \Rightarrow \delta(C_{\alpha'}, M, \sigma) > 0)$.*

In other words, $fragments(M, \sigma)$ is the set of the execution fragments of SPA M that may be generated probabilistically given a scheduler. Each scheduler for an SPA generates a (possibly infinite) set of executions and execution fragments, instead of a single execution as was the case for automata that do not exhibit probabilities. Therefore, schedulers alone are not enough to exercise complete control over the executions of an SPA, as probabilities also have an influence on possible behaviour. In particular, this implies that the notion of scheduler fairness needs to be adjusted. Consider for example the case of the SPA depicted in Figure 2.1, and two possible schedulers σ_1 and σ_2 that behave roughly as described beside the automaton. In both cases, a non-fair execution is possible – $0a1b1b1 \ldots b1b1b1 \ldots$ in the case of scheduler



$$\begin{cases} \sigma_1(0,\alpha) &= a \\ \sigma_1(1,\alpha) &= b \\ \sigma_1(2,\alpha) &= d \end{cases}$$

$$\begin{cases} \sigma_2(0,\alpha) &= a \\ \sigma_2(1,\alpha) &= c \\ \sigma_2(2,\alpha) &= d \end{cases}$$

Figure 2.1: A Simple Probabilistic Automaton and two unfair schedulers. σ_2 is *probabilistically fair*

σ_1, and $0a1c0a1c0a1\ldots c0a1c0a1\ldots$ in the case of scheduler σ_2. Under the previous definition, neither of these schedulers are themselves fair. However, note that the probability of the non-fair executions under σ_2 is actually zero, while those under σ_1 have nonzero probability.

This important distinction leads to the definition of *probabilistically fair* schedulers. Once again, this definition has been put forth previously in [BGC09, Var85, BK98].

Definition 2.28 (Probabilistically fair schedulers)**.** *A scheduler σ is* probabilistically fair *for an SPA M if it either is strictly fair, or else the measure of the subset of non-fair executions within its scheduled fragments set $fragments(M,\sigma)$ is zero.*

In other words, a probabilistically fair scheduler generates fair execution fragments *almost surely*, while they *almost never* produce unfair execution fragments. For the remainder of this thesis, when we refer to *fair* schedulers for SPAs, we will be implicitly referring to *probabilistically fair* ones, unless specifically noted.

Simulations for probabilistic automata

The notion of simulations [Mil89] is useful to compare the behaviours of automata, and is a step forward to establishing equivalence between them. In the context of probabilistic automata the concept of simulations has also been studied [SdV04]. In this work we will leverage on the particular notion of probabilistic branching simulations [Seg95]. We will later employ these simulations to show that the probabilistic formalism that we propose in this thesis establishes a close relationship between their parallel composition and these simulations.

Before we can define probabilistic branching simulations properly, we need to understand the basic blocks with which they are built. Probabilistic branching simulations must show that the probabilistic information is simulated between different automata. The main mechanism through which this is achieved is by showing that a probability distribution on the simulated system can be embedded into a probability distribution over the system that simulates it.

Definition 2.29 (Distribution embedding [SL95]). *Let $\mathcal{R} \subseteq S \times T$ be a relation between two sets S and T; and let $\delta_S \in D(S)$ and $\delta_T \in D(T)$ be two distributions on each of those sets. We say δ_S and δ_T are in relation $\sqsubseteq_{\mathcal{R}}$, noted $\delta_S \sqsubseteq_{\mathcal{R}} \delta_T$ if there exists a weight function $w : S \times T \to [0, 1]$ such that*

1. *for each $s \in S$, $\sum_{t \in T} w(s, t) = \delta_S(s)$;*
2. *for each $t \in T$, $\sum_{s \in S} w(s, t) = \delta_T(t)$;*
3. *for each $(s, t) \in S \times T$, $w(s, t) > 0 \implies s \mathcal{R} t$.*

The notion of distribution embedding bears a close relationship to embedding a probabilistic transition of one system into a combination of several transitions on the other, and vice versa. The notion of combined steps captures this relationship.

Definition 2.30 (Combined step [SL95]). *Let M be an SPA and $s \in S_M$ an arbitrary state in S. Let $\delta_C \in D(A_M \times S_M)$. We say (s, δ_C) is a* combined step *of M if there exists a weight function $w : R_M(s) \to \mathbb{R}$ such that the following hold:*

- *$\sum_{(t,a,\delta) \in R_M(s)} w((t, a, \delta)) = 1$; and*
- *for every $s' \in S_M$ it holds that $\delta_C(s') = \sum_{(t,a,\delta) \in R_M} w(t, a, \delta)\delta(s')$.*

In other words, a combined step of M at state s is a convex combination of the transitions allowed by M at state s. We will note $s \xrightarrow{a,p}_C s'$ every time that there exists a combined step $C = (s, \delta_C)$ such that $\delta_C(a, s') = p$.

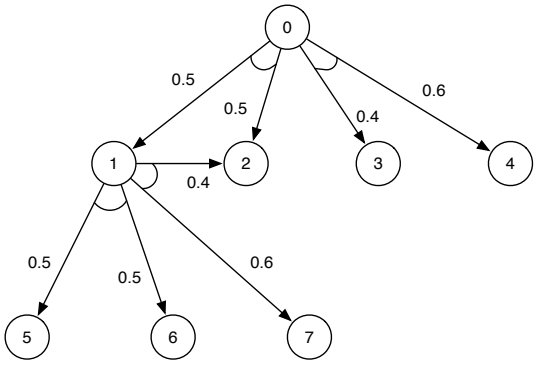
A related notion is that of *weak* combined steps. A weak combined step is essentially a product of many combined steps where at most one of them is via a non-internal action, while the rest are internal.

Definition 2.31 (Internal combined step [SL95]). *Let M be an SPA, $s \in S_M$ and $\delta_{IC} \in D(S_M)$. (s, δ_{IC}) is an* internal combined step *if either*

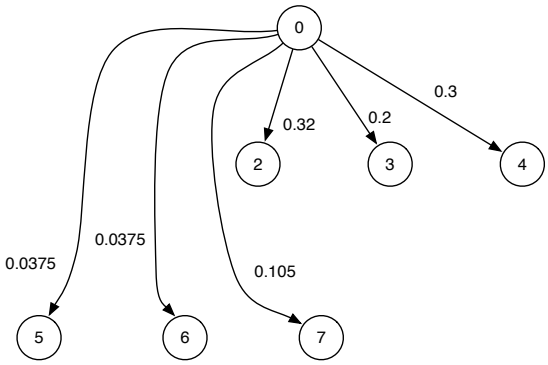
1. *$\delta_{IC}(s) = 1$; or*
2. *there exists a combined step (s, δ_C) such that for every $(a, t) \in A_M \times S_M$ such that $\delta_C(a, t) > 0$ it holds that*
   a) *$a \in A_M^H$;*
   b) *there exists an internal combined step $(t, \delta_{(a,t)})$ noted step(s, a, t); and*
   c) *for every state $s' \in S_M$, $\delta_{IC}(s') = \sum_{(a,t) \in A_M \times S_M} \delta_C(a, t) * \delta_{s,a,t}(s')$; where $\delta_{s,a,t}$ is the distribution given by the combined step step(s, a, t).*

In other words, an internal combined step is a combination of subsequent combined steps where each combined step is such that it assigns non-zero probabilities only to internal actions. Figure 2.2 shows an example of an internal combined step. In this case, all actions are hidden so no labels on transitions are necessary. Different transition distributions are told apart by the arc between the transitions. The combined transition depicted is obtained through an embedded distribution. This embedded distribution is the result of combining the distributions from state 0 with a factor of 0.5 on each distribution; and from state 1 using factors 0.3 (distribution shown on left) and 0.7 (distribution shown on right). In this case the combined step "skips" state 1.

There is a combination of combined steps and internal combined steps that is of important interest, which is the case when a state can be reached by any combination of exactly one action in $A_M^I \cup A_M^O$ and countably many interleavings of actions in A_M^H in between. We shall denote these as *weak combined steps*.



(a) Original transition distributions



(b) Combining internal distributions

Figure 2.2: An internal combined step

Definition 2.32 (Weak combined step [SL95])**.** *Let M be an SPA, $s \in S_M$ and $a \in (A_M^I \cup A_M^O)$. (s, a, δ_C) is a* weak combined step *if and only if there exists a combined step (s, δ'_C) such that every time that $\delta_C(action, state) > 0$ the following hold:*

1. *$action = a \vee action \in A_M^H$; and*
2. *if $action = a$ then either $\delta'_C(state) > 0$ or else there exists an* internal *combined step denoted $step(s, a, state) = (state, \delta'_{IC})$;*
3. *otherwise, if $action \in A_M^H$, there exists a weak combined step denoted*

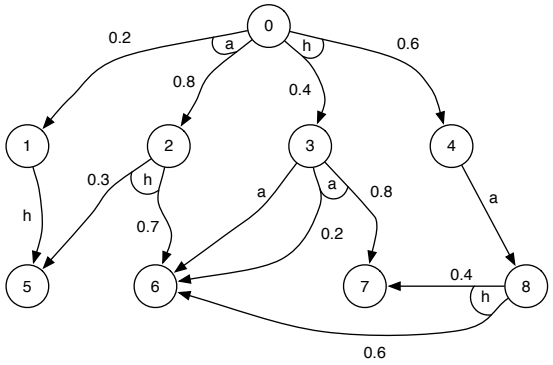
$$step(s, action, state) = (state, a, \delta_C)$$

4. *and finally, for every state $t \in S_M$ it holds that*

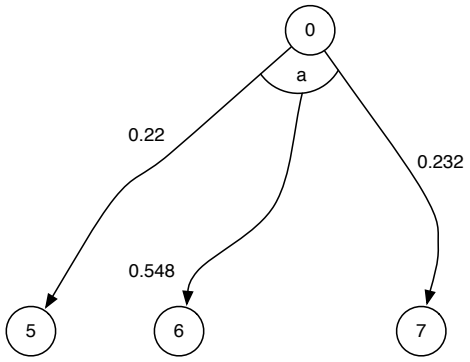
$$\delta_C(t) = \sum_{(action,state) \in A_M \otimes S_M} \delta'_C(state) * \delta_{s,action,state}(t)$$

where $\delta_{s,action,state}$ is the distribution of $step(s, action, state)$.

Figure 2.3a shows an example of distributions that can be combined as a weak combined step. Inside the arc corresponding to a distribution we note the triggering action. a is an action that is presumably shared with an external environment, while h is an internal action to the component we are modelling in this case. Figure 2.3b shows the resulting weak combined step. In this case, we obtained this step by



(a) Original transition distributions



(b) Combining a action and internal distributions

Figure 2.3: A weak combined step on action a

combining the first two transitions (originating from state 0) with factors of 0.5 each; on state 3 we use factors 0.3 and 0.7. In this case, the combination is far more complex, as hidden actions may appear before or after the action a, and even multiple times. However, it can easily be seen that the resulting step is much more simpler as well.

Definition 2.33 (Probabilistic branching simulation (PBS) [SL95])**.** *Given two Simple Segala Automata M_1 and M_2, a* probabilistic branching simulation *is a relation $\mathcal{R} \subseteq S_{M_1} \times S_{M_2}$ such that*

1. *the initial state of M_1 is related through $\mathcal{R}$ with the initial state of M_2;*
2. *for each $s_1 \mathcal{R} s_2$ and each possible transition $(s_1, a, \delta_1) \in R_1$ then:*
   a) *if $a \in A_{M_2}$, there exists a weak combined step (s_2, a, δ_2) such that the distribution δ_1 can be embedded into δ_2 through $\mathcal{R}$, that is, $\delta_1 \sqsubseteq_{\mathcal{R}} \delta_2$.*
   b) *if $a \notin A_{M_2}$, there exists an internal combined step (s_2, δ_2) such that $\delta_1 \sqsubseteq_{\mathcal{R}} \delta_2$.*
3. *every time that $s_1 \mathcal{R} s_2$, it must be that if $s_2 \xrightarrow{a_i}$ for a set of actions $a_i \in A_{M_1}$, then $s_1 \stackrel{a}{\Longrightarrow}$ as well for at least one of these actions a_i; where $s \xrightarrow{a}$ denotes that there is a transition from s with action a; and $s \stackrel{a}{\Longrightarrow}$ denotes that s can* weakly transition *to some other state on action a. That is, it either has a enabled, or there is a path of internal transitions to a state where a is enabled.*

> *In other words, whenever s_2 weakly enables some actions, at least one of them must be weakly enabled in s_1. This establishes a liveness condition*[2].

Whenever there exists such a simulation relation $\mathcal{R}$ between M_1 and M_2 we will say that M_2 simulates *M_1, and note it $M_1 \sqsubseteq_{\mathcal{R}} M_2$ (or succinctly $M_1 \sqsubseteq M_2$ if we do not care about the particular relation $\mathcal{R}$).*

Logics for property description

In order to express and analyse properties over probabilistic models such as SPAs, these automata are coupled with modal logics whose formulae express said properties. For the specific case of probabilistic models, the temporal logic pCTL [HJ89] has been introduced as an extension of the well known temporal logic CTL. Essentially, pCTL replaces path quantifiers present in CTL for *probabilistic quantification bounds* on the related path formulae.

Definition 2.34 (pCTL Syntax and Semantics)**.** *pCTL formulae are built from state and path formulae, just as CTL. Let AP be a finite set of atomic propositions. If ϕ stands for a state formula, and ψ for a path formula, then pCTL formulae are built as follows*

$$\phi \to true \,|\, a \in AP \,|\, \neg\phi \,|\, \phi \wedge \phi \,|\, P_{\sim p}\psi$$
$$\psi \to X\phi \,|\, \phi U \phi \,|\, \phi U^{\leq k} \phi$$

In the above, $\sim \in \{<, \leq, =, \geq, >\}$ and $p \in \mathbb{R}, p \in [0,1]$. Given an SPA Q and a mapping of states to atomic propositions $V : S_Q \to 2^{AP}$ defining the subset of atomic propositions that are valid for each state, we can define the satisfiability of pCTL formulae for a state $s \in S_Q$, a scheduler $\sigma \in Sched(Q)$ and an execution fragment $\alpha \in fragments(Q)$ as follows

$$\begin{aligned}
Q, s, \sigma \models true &\Leftrightarrow \text{true} \\
Q, s, \sigma \models a &\Leftrightarrow a \in V(s) \\
Q, s, \sigma \models \neg\phi &\Leftrightarrow \neg(s, \sigma \models \phi) \\
Q, s, \sigma \models \phi_1 \wedge \phi_2 &\Leftrightarrow (s, \sigma \models \phi_1) \wedge (s, \sigma \models \phi_2) \\
Q, s, \sigma \models P_{\sim p}\psi &\Leftrightarrow \textstyle\sum_{\alpha \in \psi_{sat}} \delta(C_\alpha, \sigma, Q) \sim p \\
&\quad \text{where } \alpha \in \psi_{sat} \text{ iff } \alpha, \sigma \models \psi \text{ and} \\
&\quad \text{for every other } \alpha' \in \psi_{sat} \text{ neither} \\
&\quad \alpha \leq \alpha' \text{ nor } \alpha' \leq \alpha. \\
\alpha, \sigma \models X\phi &\Leftrightarrow \alpha^s_1, \sigma \models \phi \\
\alpha, \sigma \models \phi_1 U^{\leq k} \phi_2 &\Leftrightarrow \exists 0 \leq i \leq k \cdot \alpha^s_i, \sigma \models \phi_2 \wedge \\
&\quad \forall 0 \leq j < i \cdot \alpha^s_j, \sigma \models \phi_1 \\
\alpha, \sigma \models \phi_1 U \phi_2 &\Leftrightarrow \exists 0 \leq k \cdot \alpha, \sigma \models \phi_1 U^{\leq k} \phi_2
\end{aligned}$$

It is interesting to note that satisfiability verification of a pCTL formula can be reduced to a reachability problem coupled with an optimization problem if more than one scheduler is possible [BdA95]. Informally, given a path formula ϕ, a typical pCTL state formula takes the form of a restricted classic CTL state formula, but where path quantifiers have been replaced by the probabilistic operator $P_{\sim a}$. Thus, a state formula $P_{\leq a}\phi$ (resp. $P_{\geq a}\phi$), is true at a given state of the system if its possible evolutions from that state satisfy the formula ϕ with probability at most (resp. at least) a.

Note that satisfiability depends heavily on schedulers. Under two different schedulers, the same pCTL formula may be satisfiable or not. This plays a critical role

[2] In [SL95] liveness is required on *every* action, although it is stated that it can be relaxed in the way we state here.

especially in the case of probabilistic operator formulae (that is $P_{\sim p}\psi$) as two different schedulers may assign distinct probabilities. In general, the scheduler is left unknown when evaluating the satisfiability of a formula. Therefore, it is more interesting to know if a formula holds for *any* possible scheduler. In that case, for a probabilistic formula ψ, there will exist a scheduler $\sigma^{\psi}_{\min}$ that induces a *minimum* probability on the satisfiability of the formula; and another one $\sigma^{\psi}_{\max}$ (not necessarily a different one) that induces a *maximum* probability. Then, we will usually employ a different form of the probabilistic operator to query whether the minimum or maximum probabilities satisfy our requirements. We will usually replace the operator $P_{\sim p}$ by two other operators $P^{\min}_{\sim p}$ and $P^{\max}_{\sim p}$, which are evaluated globally for every scheduler. Satisfiability will be defined as follows:

$$\begin{aligned} s \models P^{\min}_{\sim p}\psi \quad &\Leftrightarrow \quad s, \sigma^{\psi}_{\min} \models P_{\sim p}\psi \\ s \models P^{\max}_{\sim p}\psi \quad &\Leftrightarrow \quad s, \sigma^{\psi}_{\max} \models P_{\sim p}\psi \end{aligned}$$

It is important to note that there is a close relationship between pCTL satisfiability and the notion of cylinders defined in Definition 2.26. We can see from the semantics definition of pCTL that $s, \sigma \models P_{\sim p}\psi$ if the measure of the set of traces that satisfy ψ holds the relation $\sim p$. We have already established that cylinders induce a σ-algebra (in particular, a measure). The set of traces that satisfy ψ can be characterised by a (possibly infinite, but numerable) set of disjoint cylinders, based on the prefixes of the traces. Therefore, the set of traces that satisfy ψ has a definite measure induced by the cylinders that characterise it.

Finally, note that in the context of this work we will focus on a restriction of pCTL, namely its weak fragment, which we denote as WpCTL. A WpCTL formula is restricted in the sense that the X and $U^{\leq p}$ operators are prohibited. Such a restriction is reasonable when the aim of the approach is to allow further refinement by modelling internal computation of components. The *next* and *bounded until* operators, which we choose to avoid, distinguish models based on these internal computations. However, from the point of view of an external observers, such internal computation should not be discernible.

Reward structures

In addition to pCTL property specification, *reward structures* are used to convey some sense of value to traces from probabilistic models, that can then be weighed by their corresponding probability. For example, a transition reward structure that assigns a value of 1 to each transition is a standard way of defining overall time steps cost for the traces of a model. This provides a good way to model discrete time, and reliability measures such as mean time to failure can be easily interpreted over this notion of time.

The value of a reward is a random variable itself, as the accumulation of rewards over traces will depend on the probability of the transitions taken. By weighing the values of this modelling of time over the (possibly infinite) set of traces and their probabilities, we can obtain the expectation –or bounds to this expectation– of running time for an arbitrary execution.

Definition 2.35 (Reward Structures [QS96])**.** *Given a probabilistic model* $M = \langle S, s_0, A, R\rangle$*, a* transition reward structure *is a function* $\rho : S \times A \times S \to \mathbb{R}_{\geq 0}$.

Given a trace π of a probabilistic model M, and a reward structure ρ over M, the *path-reward* of π is the sum of the reward of each of its transitions. We will abuse notation and note $\rho(\pi)$ to note the path-reward of π based on reward structure ρ.

It is important to note that a reward structure assigns a non-negative reward value to transitions. Therefore, if we were to take any prefix π_{prefix} of a trace π, the path-reward of π_{prefix} will necessarily be *at most* that of π.

We will note $\Pi_{S_{end}}(M)$ (where S_{end} is a set of states) to refer to the possibly infinite set of all execution traces of M, but where they have been pruned so that the last state of each trace is one of those in S_{end}, and no other state in S_{end} exists in the trace before the end. Note that $\Pi_{S_{end}}(M)$ may contain traces of infinite length (i.e., those that never reach a state in S_{end} and therefore have not been pruned). This definition will allow us to define the value of a reward structure for *reachability* properties.

Definition 2.36 (Reachability reward values [QS96])**.** *Let $M = \langle S, s_0, A, R\rangle$ be a probabilistic model, $S_{reach} \subseteq S$ be a set of states from M, σ a scheduler for M and ρ a reward structure over M. The reachability reward value for S_{reach} under the conditions above is a random variable $X_{reach}(S_{reach}, M, \sigma)$ on $\mathbb{R}_{\geq 0} \cup \{+\infty\}$ such that the probability p of $X_{reach} = k$ is defined as*

$$Pr(\sigma, X_{reach} = k) = \sum_{\pi \in \Pi_{S_{reach}}(M), \rho(\pi) = k} Pr(\pi, \sigma, M)$$

In the definition above, X_{reach} is a random variable denoting the reward value for a random execution trace until it reaches a state in S_{reach}. As such, it may be of interest to know its *expected* or *mean* value, that is, the expected value taking into account every possible execution trace. We will note this expected value $\overline{X_{reach}}$. Note that S_{reach} may contain states for which there is a non-zero probability that they won't be reached at all. In such a case, it will happen that $\Pi_{S_{reach}}$ will contain some infinite paths. More so, these infinite paths may themselves accumulate infinite reward. In such cases, the mean $\overline{X_{reach}}$ is defined to be ∞.

Simulations and property preservations

There is a close relationship between automata that can be shown to be in a probabilistic branching simulation, and the sets of WpCTL formulae that they satisfy. However, since an automata that simulates another will probably have more behaviour than the simulated one, it is necessary to take into account some precautions regarding fairness if we wish to study these sets of properties. As we will see, this idea has a close relationship to that of probabilistically fair schedulers 2.28.

Definition 2.37 (Probabilistically convergent automata [SL95])**.** *A Simple Probabilistic Automaton M is* probabilistically convergent *under a set of schedulers Sch if for every state $s \in S_M$ and $\sigma \in Sch$, the probability of diverging (that is, performing infinitely many internal actions and no input or output actions) from state s is 0.*

Proposition 2.1 (Convergence of SPAs)**.** *Let M be a Simple Probabilistic Automaton and Sch a set of probabilistically fair schedulers for M. Then, it holds that M is probabilistically convergent under Sch.*

Proof. The proof is immediate from the definition of probabilistically fair schedulers. The only way for an infinite sequence of internal actions to have a measure larger than zero is that there is only a finite number of probabilistic choices with probability less than 1. For having such a situation be possible, there should be only a finite number of non-deterministic choices made in favour of input/output actions instead of internal actions. However, such a choice would be in direct violation of probabilistically fair schedulers, therefore no probabilistically fair scheduler may result in a divergent automaton. □

Finally, we recall a central theorem from [Seg95] regarding probabilistic branching simulations and convergent SPAs.

Theorem 2.2 (PBSs preserve WpCTL [SL95])**.** *Let M_1 and M_2 be two SPAs and such that $M_1 \sqsubseteq M_2$. Let $\phi = P_{\geq p}\psi$ be a WpCTL formula with no recursive $P_{\sim p}$ operators. Then, it holds that $M_2 \models \phi \implies M_1 \models \phi$, where the formula satisfaction is considered only under fair schedulers.*

In other words, Theorem 2.2 states that, under the conditions described, if the minimum probability of M_1 satisfying ψ is p, then the minimum probability of M_2 satisfying ψ is at least as much. Note that the theorem also applies to maximum probabilities, since the minimum probability $p_{\min}$ of satisfying a given formula is equal to $1 - p^{\neg}_{\max}$ where $p^{\neg}_{\max}$ is the maximum probability of satisfying the negation of that same formula.

This notion of probabilistic branching simulations and property preservation is central to the first Part of this thesis, as will become clearer in Chapter 3.

This sums up the preliminary concepts that we use throughout this thesis. In the next part, we tackle the problem of defining a new probabilistic modelling formalism that is suitable for the incremental specification of systems through composition, and that is amenable to incremental verification as well.

Part II

Compositional probabilistic modelling

CHAPTER 3

Probabilistic Interface Automata

In this chapter we introduce our new probabilistic modelling formalism, Probabilistic Interface Automata (PIA). But first, we start by showing why we believe there is a need for this new modelling tool in the context of software engineering.

3.1. Why a new formalism?

In the previous chapters we have hinted that there already exist several modelling formalisms, both non-deterministic and probabilistic. In this section we present a simple example to motivate the problem of compositional construction and analysis of probabilistic models. We also highlight the main issues related to the modelling of non-determinism and probabilities that threaten the compositional construction approach.

As a motivating example, we present a simple system model that will help us illustrate the problem. This discussion will be focused on the system model for a coffee machine, which is presented in Figure 3.1.

This coffee machine has a digital tactile screen with which it interacts with the user, showing the user various options at different times during operation. First, the coffee machine offers the user, through the screen, a beverage choice between either an espresso or a latte. Once the user chooses her selection, the machine clears its screen

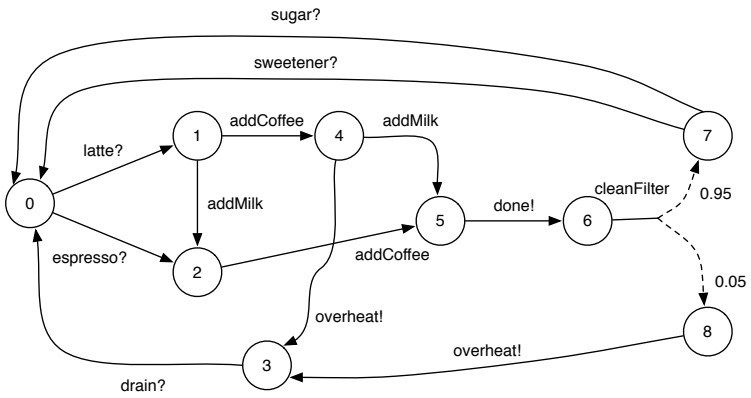


Figure 3.1: A simple coffee machine.

and possibly shows a message telling the user to wait for beverage preparation. At this point, in a way unknown to the user, the machine prepares the beverage. Then, the machine informs the user it has finished the preparation. After the beverage has been prepared, the screen prompts for the addition of sugar or sweetener. Once the choice is made by the user, the machine finally delivers the prepared drink. However, this coffee machine is known to overheat sometimes. If this happens, it is required that the user performs a manual drainage on the machine. We have some information about the conditions under which the machine overheats, so we add this information to the model.

We can already validate some behaviour on this coffee machine model. Note that we can do so without the need of having a model of the user behaviour yet. For example, we may be interested in knowing whether the machine can overheat *after* it has added coffee to the cup, as at this point the coffee may boil and spill violently towards the user, posing a safety hazard.

By observing the trace that traverses states 0, 2, 5, 6, 8, 3, we see that such an error is clearly possible. Moreover, note that there is always *at least* a 0.05 chance that this behaviour will manifest itself. This probability is completely independent of user choices; the user has no way to avoid this undesired behaviour (other than abstaining from using the machine altogether). The probability of this risky scenario could be even greater if the machine always overheats at state 4, but we do not have the probabilistic information to quantify this claim. All we know is that the likelihood of the unsafe behaviour lies between 0.05 (this is certain) and 1 (if it were the case that the machine does overheat at state 4).

For the sake of argument, assume for a moment that it is not economically viable to fix this behaviour unless its likelihood surpasses some probability threshold $p_{overheat} > 0.05$. Once we have the user model and compose it with our coffee machine, we could answer whether this threshold is met or not. For example, if the user were such that she never orders a latte, then the system model will never traverse state 4. In that case, the probability of overheating is exactly 0.05, and therefore there is no need for a fix. However, if she does order lattes, then it could theoretically overheat every time this happens.

In other words, if we want to analyse the economical viability of fixing the machine, we are interested in quantifying the occurrence of this error based on the expected behaviour of the environment interacting with the coffee machine.

In order to achieve this objective, we set out to produce a probabilistic model of the user's behaviour. However, not every modelling formalism will suit our compositional construction approach. Some choices may lead to problems which may not be immediately obvious, and these may arise from both the probabilistic aspect of the modelling and the non-deterministic as well.

3.1.1. Issues arising from probabilistic modelling

There exist two main approaches for modelling probabilities over transitions of a behaviour model; namely modelling them via a *generative* [Chr90] approach or a *reactive* [vGSS95] one.

Generative models are characterised by having a transition relation that defines, for each source state, a distribution on the cartesian product of the set of states and actions. That is, for each transition, both an action an a destination state are probabilistically selected by the same distribution. This choice of distribution modelling leads to some well-known problems when trying to compose a generative model with another [DHK99].

The first problem is that the generative paradigm requires all transitions to be

probabilistically annotated. This is true even in the case of states that may transition because of both input and output actions. Probabilistically quantifying such choices would encode the probabilities of the resolution of this race between actions. This race is usually an aspect that is outside the control of either component, since the race between actions is actually a race between two independent components that are running asynchronously.

A second problem arises if a component specifies a certain probability for an output action that is not accepted by its counterpart; or conversely provides a probability for witnessing an input action that actually may never be triggered by the environment. In such a case, the probability of that action being observed should be obviously zero in the composition, yet the component specified a non-zero probability. This contradiction needs to be resolved at composition time. Although some solutions have been proposed to redistribute this missing probability [DHK99], they are all arbitrary in the sense that they need to *guess* what the component would have done if the action were not present.

These problems can be explained technically in terms of a lack that generative models have in modelling non-determinism, and a lack of clear semantics for the concurrent composition in such cases. Not allowing non-determinism means that these models are unsuitable for use when it comes to modelling external actions the environment must act in response to.

Alternatively, the environment can be modelled under what is called the *reactive* paradigm [vGSS95]. Under this paradigm, for each action on each state there is a probabilistic distribution that defines the next state. In turn, the action at each state is chosen in a non-probabilistic fashion (even allowing for non-determinism between different distributions for a same action), and only then the destination state is determined probabilistically. Reactive models, contrary to their generative counterpart, do allow for non-determinism, but do not allow probabilistic choice between different actions. There is a workaround for this, however, using hidden probabilistic internal actions that evolve the model towards states that are either accepting input actions, or generating outputs. State 6 in the coffee machine model of Figure 3.1 shows an example of this workaround.

The use of a reactive probabilistic model solves many of the issues of the generative paradigm. However, in general, reactive probabilistic models allow for behaviour that does not necessarily consider input/output restrictions between components. Recall the property that the machine may overheat after dispensing coffee. We have already seen that this property holds with probability at least 0.05 for our modelled system. Yet, we can model a user environment that chooses to *never* synchronise on the `overheat` action, effectively blocking it. Oddly enough, the result obtained using standard composition [vGSS95] and analysis [HKNP06] is that the probability of the composite system overheating in an unsafe way is now zero, which means the error has probability 0. This would make the probability of this erroneous behaviour to be below the lower bound to error (0.05) that we had established when validating the machine model in isolation. The reason for such an unintuitive result is that the environment constrains the occurrence of a transition that should be controlled by the coffee machine. In other words, just because a component chooses *not to acknowledge* a certain action cannot be considered as a guarantee that the event modelled by the action *will not happen*.

The result in the analysis of the previous scenario is quite unintuitive. There is a property that holds for the machine, and that does not depend on any environment to hold; but when composed with a certain environment it does not hold any more. Such a contradiction indicates that something is wrong with the way we have modelled

either the system or the environment; in the way we composed them together, or in the probability computation. Again, this lack of behaviour preservation makes our goal of performing early probabilistic validation impossible.

It is important to note that, contrary to the case of generative modelling, these problems do not relate strictly to the probabilistic annotations. Rather, they arise as a consequence of the inappropriate treatment of the notion of controllability. However, they do have an impact in terms of preservation of component properties. As such, we will make use of reactive modelling for the introduction of probabilities into the environment, but will need to resolve the synchronisation issues to ensure that components cannot restrict what other components are intended to control.

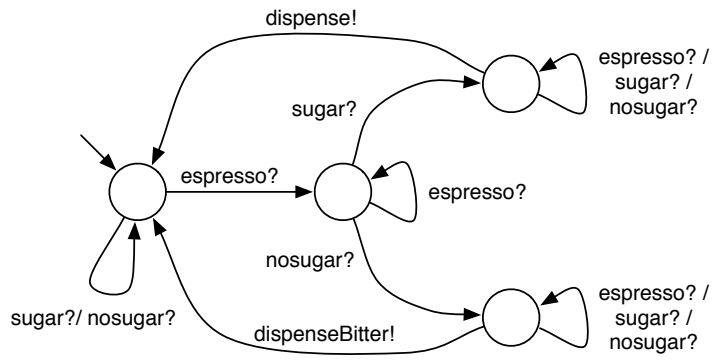
3.1.2. Issues arising from system-environment action controllability

Most of the aforementioned synchronisation semantics problems have been tackled by introducing a semantic distinction between *input*, *output* and *internal* (also called *hidden*) actions. These sets of actions represent those that the component can listen to (in the case of input actions) and emit (in the case of output actions). The set of internal actions represents those that cannot be observed from outside the component, and do not take part of the *interface* of the component. The most well-know approaches to modelling that take this action segregation idea are those of Input/Output Automata [LT87] and Interface Automata [HdA01], which we have already introduced in Chapter 2. Input/Output automata have the same action label segregation as Interface Automata. Additionally, they require that each component is input-enabled, that is, that they accept every possible input at every state. As we have already pointed out, Interface Automata relaxes this condition a little by only enforcing that input synchronisations are always possible, but do not force an input to be enabled at a given state if it is known that it will not be triggered at that state.

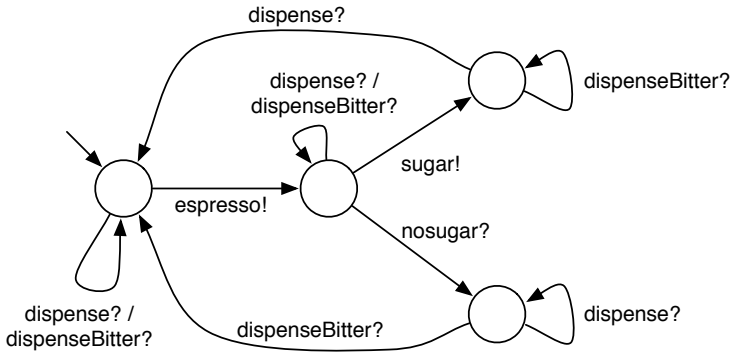
Strict input-enabledness introduces two modelling problems. First, it clutters models with unnecessary transitions. For example, we can look at the models in Figures 3.2a and 3.2b. In this figure, the I/O automaton A models a coffee machine that is much simpler to the one discussed above, since it does not allow for ordering lattes, nor exhibits the problematic overheating behaviour. In turn, I/O automaton E models a potential environment that will interact with A. It is noteworthy that the requirement for input enabledness does make the modelling more cumbersome.

The second problem is that input-enabledness restrictions are unrealistic for modelling some systems. It is usually the case that a component will accept some inputs in one state, while it will accept a different set of inputs in another. In fact, it may not accept any inputs at all until it finishes some internal computation, at which point it will accept new inputs. The need for immediate synchronisation with intended output actions hampers an iterative refinement approach where this internal behaviour is gradually modelled. As an example of how this problem arises, refer back to Figures 3.2a and 3.2b. An engineer may now decide that the level of abstraction used to depict the behaviour of A is too high, and she may decide to model some of the internal behaviour of the component. In particular, the engineer decides it would be interesting to note that the machine needs to heat the water for the beverages prior to preparing them. The result of this decision is a new model depicted in Figure 3.3a. However, this new model is now not an I/O automaton respective to the environment model, as the grey state is blocking inputs from the environment that, at this point, may choose the beverage and later whether to add sugar or not.

In order to turn this model into a valid Input/Output automaton it becomes necessary to take into account that the environment model expects a single push of the `espresso` button to prepare the drink, and a second one for the sweetener choice.



(a) Coffee machine model



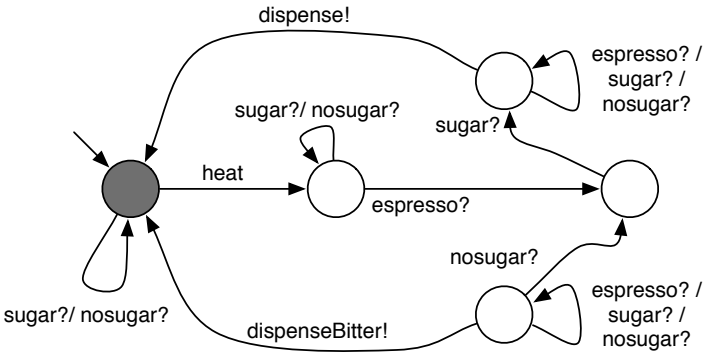
(b) Environment model

Figure 3.2: I/O models for the simple coffee machine

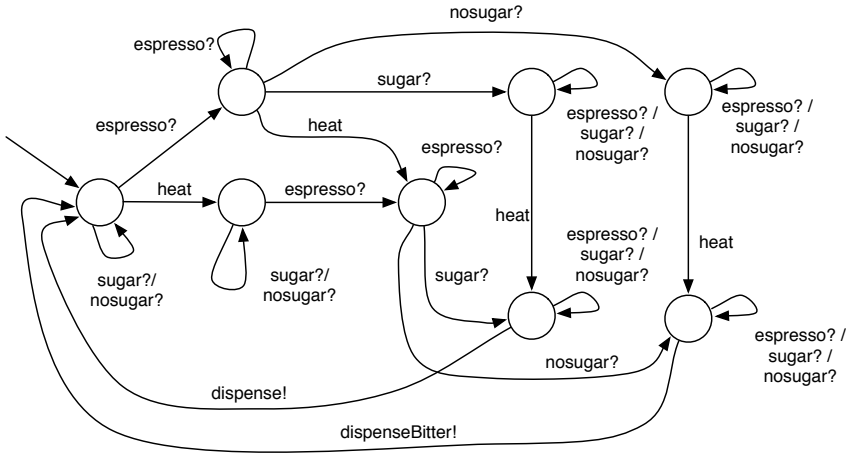
Simply adding loops and ignoring the environment `espresso`, `sugar` and `nosugar` actions is insufficient, as the environment would now be expecting the beverage to be dispensed, and such an action would never happen. The model depicted by the automaton shown in Figure 3.3b fulfils both this requirement and I/O synchronisation. It is easy to see that it is overly complex because of this need to remember user choices that may have happened during the internal actions of the machine. This complexity arises even for the very simple behaviour exhibited for this machine. Of course, an alternative modelling could consider signalling the environment that although the input actions are enabled, they are being ignored. However, such a decision involves a rework on the environment itself. Worse, such changes are a result of trying to fit a methodology rather than an attempt at modelling the actual interaction.

Interface Automata

An alternative formalism, but one that still retains the notion of segregating interfacing actions, *Interface Automata* [HdA01] has been proposed. The Interface Automata formalism stipulates that the composition of a pair of components will be legal only if components do not block each other, that is, if every time that one component intends to exercise one of its output actions, the other component enables such action (as part of its own input actions). In this case, it is not necessary to spuriously enable input actions, as only those that are actually needed are mandatory to be enabled. In this sense, Interface Automata allow for succinct modelling of interfacing protocols than their Input/Output counterpart, which assumes input-



(a) A refined model that violates I/O automata rules



(b) A complex refinement satisfying I/O requirements

Figure 3.3: Approaches to refinement of the coffee machine model

enabledness. However, similarly to I/O automata, they do require that the non-blocking behaviour be immediate, that is, whenever a component wants to emit one of its output actions, the corresponding input action must be immediately enabled at its counterpart component.

Except for the immediacy restrictions depicted above, Interface Automata seem to be a natural choice for modelling synchronisation and controllability. From an engineering point of view, it is natural to model the restriction of certain actions at selected states as long as these restrictions are compatible with the behaviour of the component that controls them.

In this way, assumptions about the behaviour of cooperating models can be encoded directly, easing the task of modelling interactions such as protocols enforcing ordered method calls, internal uninterruptible behaviour or system exceptions, among other useful system properties. This results in more concise models, as the engineer is released from the obligation of having to explicitly model responses for interactions that are known to not occur in the reality being modelled.

It is important to note, however, that specifying a similar formalism to the one we will present, but using Input/Output automata-like modelling is feasible. The choice of Input/Output Automata over Interface Automata is of no consequence regarding the solutions to the problems described in the previous sections, and the way to resolve them would be similar in both cases.

Regarding the immediate enabledness requirements discussed above, a formalism

that allows for modelling such delayed synchronisation is thus desirable. Of course, an important requirement for such a model is that it can be guaranteed that for every possible future behaviour, the synchronisation point will always be available. Such guarantees will require some restrictions on unfair behaviour of the system under analysis that may hamper such guarantees. We will study these guarantees when we present our modelling formalism in Section 3.2.

3.1.3. Combining probabilities modelling and synchronisation semantics

Summarising the previous sections, in order to model the probabilistic behaviour of the environment and compose it with a non-probabilistic behaviour model of the system to obtain meaningful quantitative results, a formalism is needed that can *i*) allow for modelling of both non-deterministic behaviour and probabilistic behaviour, *ii*) address notions of controllability and monitorability of actions by the environment and system (including synchronisation notions and delayed behaviour), and *iii*) preserve probabilistic properties of the environment after composition.

In the following sections we propose a formalism which distinguishes output/-controlled and input/monitored actions, and also supports probabilistic and non-deterministic behaviour. Our formalism is inspired on *probabilistic reactive* models for introducing probabilities, as we discussed above. Synchronisation will be modelled inspired on Interface Automata. This combination allows for satisfying objective *i*) in the above paragraph, as well as *ii*).

However, challenges arise from the combination of these two formalisms. The previous discussion hints at some of these challenges, and we elaborate on our solution on the next sections. We focus especially on the mechanisms that allow us to ensure that *iii*) is satisfied.

We will also tackle the problem of the need for immediate synchronisation.To this end, we will introduce a notion of fairness for executions of these automata that allows us to distinguish those cases where future synchronisation of delayed actions is guaranteed from those where it is not. Further, we will also present a suitable composition operator for these automata and in Theorem 3.1 we demonstrate the required results of property preservation.

3.2. Probabilistic Interface Automata

In this section we present our new modelling formalism. This automata-like formal model is designed to overcome the shortcomings other probabilistic modelling formalisms have, as was discussed in Section 3.1.

3.2.1. Definitions, relations with IA and SPA

Leveraging on the definitions presented in previous sections, we can attain our aim of merging the notion of Segala's Simple Probabilistic Automata with that of Interface Automata. As a way to attain this objective, we define *Probabilistic Interface Automata* based on SPAs.

Definition 3.1 (Probabilistic Interface Automata)**.** *A* Probabilistic Interface Automaton *(PIA) is a tuple of the form* $M =< S_M, s^0_M, A^I_M, A^O_M, A^H_M, R_M >$ *where the sets* A^I_M*,* A^O_M *and* A^H_M *are mutually disjoint, and such that defining* $A_M = A^I_M \cup A^O_M \cup A^H_M$ *yields a Simple Probabilistic Automaton* $M_{SPA} =< S_M, s^0_M, A_M, R_M >$*.*

Therefore, a Probabilistic Interface Automaton is an SPA that shares the input, output and hidden action semantics from Interface Automata. Note that since a Probabilistic Interface Automaton must induce an SPA, then $R_M \subseteq S_M \times A_M \times D(S_M)$. Note also that a Probabilistic Interface Automaton A has an *underlying Interface Automata*, noted $A\downarrow$ and defined as follows:

Definition 3.2 (Underlying IA)**.** *Given a Probabilistic Interface Automaton E, we define its* underlying Interface Automaton *as the classic Interface Automaton $E\downarrow = < S_{E\downarrow}, s^0_{E\downarrow}, A_{E\downarrow}, R_{E\downarrow} >$ such that*

- $S_{E\downarrow} = S_E$;
- $s^0_{E\downarrow} = s^0_E$;
- $A_{E\downarrow} = A_E$; *and*
- *for all $s, s' \in S_{E\downarrow}$, $a \in A_{E\downarrow}$, $(s, a, s') \in R_{E\downarrow}$ if and only if there exists a distribution $\delta \in R_E(s, a)$ such that $\delta(s') > 0$.*

Simply put, the underlying Interface Automaton of a Probabilistic Interface Automaton is a non-deterministic automaton with the same state and transition edge structure, but where all probabilities have been *forgotten* and replaced by non-deterministic transitions, leaving all other information unchanged. Conversely, it is also worth noting that a classic Interface Automaton can be embedded in a Probabilistic Interface Automaton by restricting R_M to Dirac distributions. This definition is akin to that of *underlying graph* of Markov chains [Seg95], but this definition makes explicit the fact that the obtained graph is an Interface Automaton.

The notion of underlying Interface Automaton turns out to be useful for a natural way to define Probabilistic Interface Automata composability.

Composability and product

Definition 3.3 (Composability)**.** *Given P and Q two Probabilistic Interface Automata, we will say that P and Q are* composable *if their underlying Interface Automata $P\downarrow$ and $Q\downarrow$ are themselves composable (refer to Definition 2.9).*

The concepts of execution fragments and schedulers that were introduced in Chapter 2 still apply to Probabilistic Interface Automata. Since these automata can be directly embedded into an SPA, we will refer to the SPA definitions for these concepts while working with PIAs. Probabilistic Interface Automata product, however, does express some differences regarding the composition of the transition relation. The definition of illegal states in Probabilistic Interface Automata is of special interest, as the synchronisation conditions on PIAs are much more relaxed than those of classic Interface Automata. Of course, this relaxation does have an impact on the modelled behaviour of the components and the composition. We will analyse this relation with further detail later in this chapter, and we will establish a link between these synchronisation conditions and our objective of behaviour preservation.

Definition 3.4 (Product)**.** *Given P and Q two composable Probabilistic Interface Automata, their product $P \otimes Q$ is defined by the Probabilistic Interface Automata*

$$P \otimes Q = < S_{P\otimes Q}, s^0_{P\otimes Q}, A^I_{P\otimes Q}, A^O_{P\otimes Q}, A^H_{P\otimes Q}, R_{P\otimes Q} >$$

where $S_{P\otimes Q}$,$s^0_{P\otimes Q}$, $A^I_{P\otimes Q}$, $A^O_{P\otimes Q}$ and $A^H_{P\otimes Q}$ are defined in the same way as Interface Automata composition. Its transition relation $R_{P\otimes Q} \subseteq S_{P\otimes Q} \times A_{P\otimes Q} \times D(S_{P\otimes Q})$ however, is constructed in the same way as it was constructed for SPAs (refer to Definition 2.20).

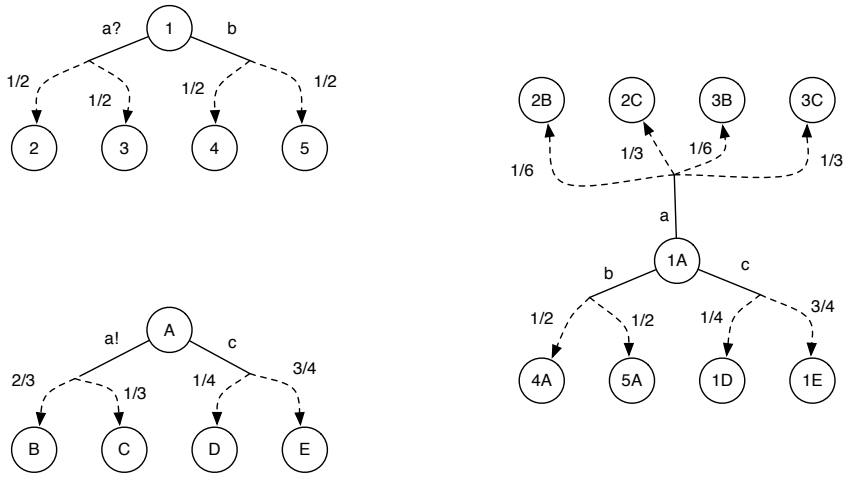


Figure 3.4: Probabilistic Interface Automata (partial) product. Only the composite state 1A is shown.

Note that we are overloading the operator $\otimes$ to refer to all of IA, SPA and PIA compositions. The specific meaning in each case, however, can be easily understood from the context in which we use the operator. Refer to Figure 3.4 for an example of two-state composition, where $a?$ makes explicit that a is an input action for the automaton, and $a!$ denotes it is an output. Action labels that are left without annotation are internal.

Recall that we would like the definition of Probabilistic Interface Automata to exceed a syntactic notion and actually have an interesting semantics, as otherwise its usefulness would be drastically reduced. We will see to this objective in Theorem 3.1.

Note that the probabilistic composition operator and the underlying Interface Automata operator are distributable over one another. That is, if P and Q are two Probabilistic Interface Automata, then $(P \otimes Q)\downarrow = P\downarrow \otimes Q\downarrow$.

Illegal states and valid environments

The notions of illegal states and valid environments can also be extended for Probabilistic Interface Automata. In essence, they share the same definition, except for an important difference in the illegal states concept. As we discussed earlier in this chapter in Section 3.1, the original criteria for defining illegal states in the case of Interface Automata is too stringent, as it requires *immediate* enabledness of output actions in the component to be composed with.

In the following definition, we will make use of ACTL formulae over the underlying Interface Automaton of a given Probabilistic Interface Automaton P. Refer back to Definitions 2.17 and 2.18 on Chapter 2 for a refresher on ACTL.

Definition 3.5 (Illegal states)**.** *Given two composable Probabilistic Interface Automata P and Q, their product's illegal states are defined by the set $Illegal_{ProbIA}(P,Q) \subseteq S_P \times S_Q$. For any $s \in S_P$, $t \in S_Q$, $(s,t) \in Illegal_{ProbIA}(P,Q)$ if it is the case that either*

1) *for any action $a \in A_P^O \cap Shared(P,Q)$ enabled in s (respectively, actions $b \in A_Q^O \cap Shared(P,Q)$ enabled in state t) it must be that the ACTL formula*

$$\forall(X_a\, True) \vee (X_{A_Q \setminus Shared(P,Q)}\, True)\mathcal{U}(X_a\, True)$$

does not hold for $Q\downarrow$ *at state* t *under fair schedulers (respectively*

$$\forall(X_b\,True) \vee (X_{A_P \setminus Shared(P,Q)}\,True)\mathcal{U}(X_b\,True)$$

does not hold on $P\downarrow$ *at state* s*); or*

II) *s is such that its only enabled actions on P are a subset A_s of $A_P^I \cap Shared(P,Q)$ (respectively, enabled actions at t on Q are a subset A_t of $A_Q^I \cap Shared(P,Q)$) and the ACTL formula $\forall(X_{A_Q \setminus Shared(P,Q)}True)\mathcal{U}(X_{A_s}True)$ does not hold on $Q\downarrow$ at state t (respectively the formula $\forall(X_{A_P \setminus Shared(P,Q)}True)\mathcal{U}(X_{A_t}True)$ does not hold on P at state s) when being evaluated, restricting evaluation only to fair schedulers.*

Note that the semantics of the $\mathcal{U}$ operator above is that of a *strong until*. The difference between *weak until* ($\mathcal{U}_w$) and *strong until* is subtle and merits a reminder: an execution α satisfies the path formula $\psi\mathcal{U}_w\phi$ (that is, $\alpha \models \psi\mathcal{U}_w\psi$) if there exists an index i such that $\alpha_i^s \models \psi$ and $\forall 0 \leq j < i \cdot \alpha_j^s \models \phi$; or alternatively $\alpha_k^s \models \phi$ for every $k \geq 0$. The *strong until* is more stringent in the sense that it does not allow the second alternative, and it needs the step α_i^s such that $\alpha_i^s \models \psi$ to exist. In other words, the *strong until* demands the formula ψ to be true at some point, while *weak until* does not, as long as ϕ is never violated.

The illegal state definition for Probabilistic Interface Automata relaxes that of Interface Automata, so that synchronisation does not need to be available at each state, but may be *finitely* delayed, under certain conditions. Intuitively, the first clause (I) enforces the claim that states will only be legal if they allow an output action to be taken immediately; or else, if the current state is momentarily blocking it, it is such that every possible continuation of the trace from that state involves only internal actions of the blocking component until it allows the blocked behaviour to happen. However, it still is required that the synchronisation be carried out, regardless of any internal actions the delaying component takes. It must be noted that this future synchronisation delayed by a component cannot depend on action requirements by its counterpart. That is, a component may delay synchronisation *only* through the execution of internal actions, and every possible fair continuation of such execution fragments must eventually synchronise. Such restrictions are essential to further probabilistic analysis, because failure to eventually accept such behaviours would result in missing behaviour from the environment, along with its probability. Note that we refer to *fair* executions in the sense of *probabilistic fairness*. In other words the probability distributions that govern the transitions may allow for an indefinite delay of the required synchronization, but the probability of selecting this delay indefinitely should be zero (i.e., such a situation should *almost never* arise).

Clause (II) in turn, describes that states that only allow for *shared input* actions are such that they must eventually always receive one of these input actions in order to advance. These are states that need to receive an input in order to advance (because the states themselves do not generate outputs and do not perform internal actions), and must be guaranteed to eventually receive one of these inputs and cannot be kept stuck forever. This second restriction essentially imposes an advancing condition on quiescent states of the components. On the one hand, this forces the counterpart component to actually have one of those actions as an output to be processed by the blocked component. On the other hand, fairness conditions are vital to ensure, additionally to the fact that the action must be available, that again the action is taken at some point in the future and is never indefinitely delayed.

These restrictions allow us to relax the stringent immediate blocking semantics, and let us model components' internal behaviour in a way that doesn't interfere

with the synchronising semantics. Also, note that these conditions are not necessarily exclusive to Probabilistic Interface Automata. They can be used to relax the Interface Automata illegal states condition as well.

3.2.2. PIAs and property preservation

In the case of Probabilistic Interface Automata, WpCTL is a viable logic for property observation, since we can leverage on their underlying SPA structure and the scheduler definition (recall Definition 2.13). For a refresher on WpCTL, refer back to Definition 2.34.

The main contribution of Probabilistic Interface Automata to software engineering practices is to convey the notion that the product of two interfacing probabilistic models is not merely a syntactic convenience, but that it does maintain a semantic relationship between the individual models, their composition, and their observable properties. The following theorem and its corollary see to this objective.

Theorem 3.1 (WpCTL property preservation)**.** *Let A and B be two composable Probabilistic Interface Automata such that their product $A \otimes B$ is legal (that is, it contains no reachable illegal states). Let ϕ_A be a WpCTL property such that ϕ_A is expressed only in terms of the alphabet of actions in A. Then, if $A \models \phi_A$ under fair schedulers, then it holds that $A \otimes B \models \phi_A$ under fair schedulers as well.*

Informally, the theorem provides a validation for the compositional view of the component-composite model relation, as properties formulated early in the validation process do not lose their meaning once the components are integrated into a whole composite model. Intuitively, this is true, since the composition does not add new behaviour and neither does it prohibit allowed behaviour by the environment.

We delay for a moment proving the theorem and present a useful corollary regarding the extreme probabilities (minimum and maximum) of satisfaction of a given WpCTL property.

Corollary 3.1 (Maximum and minimum scheduler probability)**.** *Let A and B be defined as in Theorem 3.1. Further, let $\phi_A = P_{\leq p}\psi_A$, where p satisfies that for any other formula $\rho_A = P_{\leq p'}\psi_A$ where $p' > p$, it holds that $A \models \phi_A$ but $A \not\models \rho_A$. In other words, p is the maximum probability of satisfying ψ_A on A.*

Similarly, let $\phi_{A\otimes B} = P_{\leq q}\psi_A$ such that $A \otimes B \models \phi_{A\otimes B}$; and q is such that any other $\rho_{A\otimes B} = P_{\leq q'}\psi_A$ for $q' > q$ is not satisfied by $A \otimes B$. That is q is the maximum probability of satisfying ψ_A on $A \otimes B$. Then, it holds that $q \leq p$.

This same corollary applies analogously to the minimum *probabilities of satisfying ψ_A.*

Proof. Suppose $q = p + r$ with $r > 0$. Then $A \otimes B \models P_{\leq p+r}\psi_A$. Because of Theorem 3.1, it must be then that $A \models P_{\leq p+r}\psi_A$. But p was the *maximum* probability of satisfying ψ_A on A. Contradiction. □

We can now go back to the proof of Theorem 3.1.

Proof. Recall Theorem 2.2. This theorem expresses a property over two SPAs M_1 and M_2 that are related under a probabilistic branching simulation such that $M_1 \sqsubseteq M_2$. If that condition holds, then for a WpCTL formula ϕ it also holds that $M_2 \models \phi \implies M_1 \models \phi$. Since in our setting A, B and $A \otimes B$ are PIAs, they are also SPAs. If we were to show that there exists a probabilistic branching simulation $\mathcal{R}$ such that $A \otimes B \sqsubseteq_{\mathcal{R}} A$, the theorem would be proved as a consequence of Theorem 2.2.

We will show that $\mathcal{R}$ indeed exists by construction. We define $\mathcal{R} \subseteq S_{A\otimes B} \times S_A$ such that $(s,t)\mathcal{R}r$ if and only if $s = r$. We informally recall the four conditions of PBSs definition (Definition 2.33) and show they are satisfied by $\mathcal{R}$ and we'll prove each formally.

First, we check that the initial state of $A \otimes B$ is related through $\mathcal{R}$ with the initial state of A. The initial state of $A \otimes B$ is (s_0^A, s_0^B), the product of the initial states of A (s_0^A) and B (s_0^B). By definition of $\mathcal{R}$, $(s_0^A, s_0^B)\mathcal{R}s_0^A$.

Second, we check the simulation conditions on internal actions of $A \otimes B$ and those shared with A. Now take an arbitrary reachable state $(s,t) \in S_{A\otimes B}$. By definition of $\mathcal{R}$ it holds that $(s,t)\mathcal{R}s$. Consider the possible steps originating on (s,t) at $A \otimes B$, that is $R_{A\otimes B}((s,t)) \subseteq A_{A\otimes B} \times D(S_{A\otimes B})$. Let (a,δ) be an arbitrary transition on this set.

Proving for an action a invisible to A If $a \in A_{A\otimes B} \setminus A_A$, then a is an action invisible to A (internal to $A \otimes B$). In this case we need to see that there exists an internal combined step (s, δ_{IC}) for A, such that $\delta \sqsubseteq_{\mathcal{R}} \delta_{IC}$. Define $\delta_{IC} = Dirac(s)$, that is, $\delta_{IC}(s) = 1$ and 0 everywhere else. To prove $\delta \sqsubseteq \delta_{IC}$, we refer back to Definition 2.29. We need to show the existence of a weight function $w : (S_A \times S_B) \times S_A \to [0,1]$ such that

1. $\forall r \in S_A$, $\sum_{(x,y)\in S_A\times S_B} w((x,y),r) = \delta_{IC}(r)$;
2. $\forall (x,y) \in S_A \times S_B$, $\sum_{r\in S_A} w((x,y),r) = \delta(x,y)$; and
3. $w((x,y),r) > 0 \Rightarrow (x,y)\mathcal{R}r$.

We define the weight function w as follows:

$$w((x,y),r) = \begin{cases} \delta(x,y) & \text{if } x = r \\ 0 & \text{otherwise} \end{cases}$$

We prove each condition on w individually. First, let $r \in S_A$. We compute $\sum_{(x,y)\in S_A\times S_B} w((x,y),r)$.

$$\begin{aligned} &\textstyle\sum_{(x,y)\in S_A\times S_B} w((x,y),r) = \\ &\quad = \textstyle\sum_{y\in S_B} w((r,y),r) \text{ as } w \text{ is defined as 0 otherwise} \\ &\quad = \textstyle\sum_{y\in S_B} \delta(r,y) \end{aligned}$$

Now, recall that δ is a distribution arising from a transition on an action invisible to A. Therefore if the originating state was (s,t), only states of the form (s,t_i) will have nonzero probability for δ. So, if $r \neq S$, $\sum_{y\in S_B} \delta(r,y) = 0 = \delta_{IC}(r)$ as δ_{IC} was 0 everywhere but s. If $r = s$, then $\sum_{y\in S_B} \delta(r,y) = \sum_{y\in S_B} \delta(r,y)$ which sums over the whole support set of δ, so equals to 1, which in turn is $\delta_{IC}(s)$.

Conversely, take an arbitrary $(x,y) \in S_A \times S_B$. Now, $\sum_{r\in S_A} w((x,y),r) = w((x,y),x)$ as w is zero otherwise. And $w((x,y),x) = \delta(x,y)$ by definition.

Finally, it is easy to see that if $w((x,y),r) > 0$ it must be that $r = x$. By definition of $\mathcal{R}$, $(x,y)\mathcal{R}x$, so the final point is proven.

Proving for a shared action a In this case, we need to show the existence of a weak combined step (s, a, δ_{WC}) on A. Action a is obviously enabled on s as otherwise a would not synchronise and a would not be enabled on (s,t) either. Since a is a shared action, the distribution δ on $A \otimes B$ must have arisen from the product of a distribution δ_A on a transition from A, and a distribution δ_B on B. That is, for any $(x,y) \in S_A \times S_B$, $\delta((x,y)) = \delta_A(x) \times \delta_B(y)$.

In this case, we define $\delta_{WC} = \delta_A$, while w is defined in the same way as it was defined before. The conditions on w are proven in the same way as in the previous case.

Proving the liveness condition on simulations Finally, in order for $\mathcal{R}$ to be a probabilistic branching simulation, we need to show that whenever $(s,t)\mathcal{R}s$ and s enables a set of actions $A_A(s)$, then (s,t) weakly enables a set of actions $A_{A\otimes B}(s,t)$ with at least one action in common. The proof is a direct consequence of the fact that $A \otimes B$ has no illegal states. Assume $s \xrightarrow{o} s'$ for at least one *output* action o. Because of condition i) on illegal states, every internal-action path on B must eventually enable action o to be illegal-state-free. Therefore, o is weakly enabled on $A \otimes B$.

Alternatively, suppose that $s \stackrel{i}{\Longrightarrow}$ only for internal actions i. In this case, because of condition ii) on illegal states, B must weakly enable at least one of them, so enabledness on $A \otimes B$ is also guaranteed. □

As an additional note, it is worth noting that composition, while preserving WpCTL properties, may not actually preserve the *exact* event probabilities for a given property. For example, assume environment E satisfies the property $P_{\leq 0.75}\psi$. Recalling the satisfiability definition, this means that E satisfies ψ with probability at most 0.75 under the control of *any* scheduler. There may, or may not, be an actual scheduler that, when controlling E actually witness probability 0.75 for formula ψ. The interesting issue is that even if there is such a scheduler, the existence of a scheduler for $E \otimes S$ witnessing probability 0.75 for ψ is not guaranteed; in fact every scheduler for $E \otimes S$ may witness an inferior probability.

This distinction, however, is only important from a more formal point of view. In practice, if the approach is being used in a software engineering context, this distinction is not as important. For example, an engineer may be interested in proving that a given component has at most a 0.05 chance of failing. That is, the engineer poses the formula $P_{\leq 0.05}failure$, where *failure* is a formula capturing the conditions under which the component actually fails. The engineer then validates this formula over the component and finds it to be true. Then, it is guaranteed that the probability of this same component failing over the whole composition is at most 0.05. Further, suppose that in fact the engineer observes that the probability of failure of the isolated component is *exactly* 0.05. However, it may very well be that, because of behaviour restriction imposed by the composition, the exact failing probability drops to, for example, 0.03 or even zero in the composition. In any case, the reliability objective posed by the engineer, although it does not preserve the exact probability, is only reinforced by the composition. The failing probability never *increases* because of the composition, it can only decrease (and in fact, can only decrease down to the *minimum* probability of failure of the isolated component, and no further).

CHAPTER 4

Preliminary evaluation

In this Chapter we outline, through the use of the model of a critical reactive system, the benefits of modelling systems with Probabilistic Interface Automata. We further argue that *i*) realising a model of an interacting environment that conforms to Probabilistic Interface Automata is not necessarily hard; *ii*) the resulting model is easily understandable and is not cluttered by the presence of unnecessary transitions that are foreign to the actual interaction. In addition, *iii*) we empirically show, by calculating the probability of some interesting properties, that the preservation results of Theorem 3.1 do hold.

In order to illustrate our approach, we will quantitatively analyse the behaviour of an existing software system. We provide a model of this system and analyse some properties of interest. We validate our approach by constructing a probabilistic behaviour model of the system's environment, and show that it is a Probabilistic Interface Automaton. Further, we show that this Probabilistic Interface Automaton is a legal environment for the system model.

Later, we analyse the impact of varying the expected probabilistic behaviour of this environment, as well as the probabilistic behvaiour of the system itself. By doing so we show that, in a way that is independent of the actual probabilities modelled in the behaviour, the result of Theorem 3.1 holds. In other words, the initially validated behaviour of the system (or the environment) is shown to still be valid when interacting with these different environment (or respectively, system) models.

In order to show that this behavioural preservation holds, the various environmental/system variations are composed with the original system/environment models, and we produce bounds on the probability of environment-specific and system-specific properties holding. In each case, we verify the composability of the system/environment ensemble, and analyse and validate the property preservation characteristics of Probabilistic Interface Automata.

4.1. The TeleAssistance System

The software system we analyse is an extension of the case study presented in [EGMT09], which was further refined in [PBU09]. The original model was realised under the stringent synchronisation conditions of Interface Automata. Although our approach is applicable to Interface Automata, in this Chapter we relax the restrictions that were previously present regarding the immediate need for synchronisation,

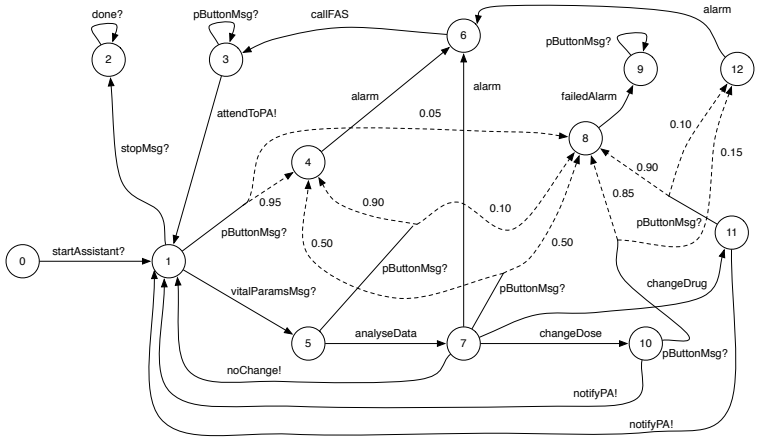


Figure 4.1: The TeleAssistance Software.

and allow for it to be delayed. This is in fact a more realistic approach to modelling the problem, which takes into account the internal processing of the TeleAssistance software. In turn, this will allow us to better illustrate the modelling benefits of Probabilistic Interface Automata.

The *TeleAssistance* (TA) software is envisioned as a web-based application providing remote assistance to patients that, for any reason, need to remain at their homes and need constant monitoring. In its most basic interaction, the patient commences operation via a `startAssistance` command. This results in the TA system entering an infinite loop, where it can accept any of the following requests:

- `stopMsg`, which signals that the user wishes to cancel TA service for now.
- `vitalParamsMsg`. This signal allows the user to send various body readings via a supplied device. The patient's health parameters are analysed by the application server which, if necessary, may then suggest a course of action. The system may decide that a change in the patient's medication is needed, and communicates this decision via either the `changeDrug` or `changeDose` commands. These messages result in an automatic adjustment of the medication that is delivered to the user. If a successful adjustment is made, the patient is notified via the `notifyPA` message, but no details regarding the kind of adjustment are communicated to the user. If any anomalies are detected during the analysis, a First-Aid Squad (FAS) is requested and sent. In the case of a FAS being sent, the patient is informed via the `attendToPA` message.
- `pButtonMsg` allows the patient to activate a panic signal. The patient may trigger such a signal if at any moment she begins to feel sick and cannot cope. The `pButtonMsg` signal triggers an alarm in the TA service. A successful processing of the alarm results in a FAS being sent to the patient's home. The system is expected to always dispatch a FAS in the case of a panic signal.

We have augmented the simplified model presented in [EGMT09] in two ways in order to introduce richer software-environment interactions. First, by specifying that for emergency reasons the panic button may be pushed at any operational state of the software, even if waiting for other results. Second, by refining the feedback provided by the software so that the patient is also told if no medication adjustment is needed. Note that these changes make the system model more complex, rather than ease our environment modelling task, since the model actually grows larger

and introduces new reactive actions. Similarly, the changes we introduced are quite general; they are in no way tailored to our modelling approach.

We depict an abstract model of the TA software system in Figure 4.1. Note that the model can be understood as an Interface Automaton, which is a particular case of the Probabilistic Interface Automata introduced in the previous Chapters. As is customary, output actions are appended with '`!`', and input actions are appended with '`?`', while internal actions are left with no annotations.

The TeleAssistance software as modelled exhibits a critical failure. This failure is reached by the triggering of the `failedAlarm` event. This happens if an alarm has been raised but it failed to be acknowledged or properly handled, thus not calling and sending the First-Aid Squad. In this iteration of the system model, such an error (state 9) can be reached at several times during execution. All of the interactions that reach the error state are the result of the user pressing the panic button. However, it is not always the case that this button press will trigger the failure. The reasons behind this erratic behaviour are unknown, but we have some quantitative, probabilistic information that we can analyse. We know that once the software has started analysing vital parameters' data the probability of failure when the panic button is pressed increases (see states 1, 7, 10 and 11). This is likely the result of event sequences not properly foreseen by the team documenting the system specification.

Relying on the software's model only, we can easily see that such a state is reachable. However, actual probability of reaching said failure state is highly dependent on factors external to the TA system as well as the depicted probabilities. First, it will depend on the environment's behaviour, which may be modelled probabilistically. For example, if the user never panics and does not press the button, the failure is evidently never realised. Another source of uncertainty is in the timing races that come up in the interaction between the environment and the system. Sometimes, the system may be fast enough that it does not allow the button to be pressed while analysing the data, and therefore it will avoid the failure. This speed can work against the software reliability as well. If the button press is processed after the drug or dose is changed, it will surely end in failure. Therefore the probability of reaching the failure state in a given execution depends on both the environmental interaction as well as the scheduling between the environment and the system. The probability of failure can range between 0 if the user never panics and 1 if the user panics repeatedly, since the failure will eventually happen in that case. It is also interesting to note that on any one interaction cycle the probability of failure ranges between 0 and 0.90 if it both panics and the drug has been changed. Table 4.1 shows a range of failure properties and their associated probabilities. Recall that, because of non-determinism, we will not obtain a single probability as the event measure, but rather an interval of where the probability lies. These intervals are determined by the schedulers for which the probability of occurrence of the event is lowest; and conversely, the scheduler for which this probability is largest. Minimum probabilities are sometimes uninteresting since they are zero in these cases.

Table 4.1 first states the properties being evaluated in a colloquial manner. The first two properties have been extracted from [EGMT09]. To perform the calculations we used the model checker PRISM [HKNP06], a well-known probabilistic verification tool. These properties were sometimes modelled with suitable pCTL formulae, and some others were modelled by an additional observer automaton, which was modelled in the form of a valid, composable PIA. For example, property SP1 is captured by the formula $true\,U\,(state = 9)$; and SP3 and SP4 are modelled respectively by the formulae $actionCount < 1\,U\,(state \in \{4, 8, 12\})$ and $actionCount < 5\,U\,(state \in$

System property	P_{min}	P_{max}
SP1: The button is pressed yet the First-Aid Squad is not sent to the patient location	0.0000	1.0000
SP2: A `changeDrug` or `changeDose` occurs, and the next message received by the TA generates an alarm which fails	0.0000	0.9000
SP3: The button is pressed during the first interaction	0.0000	1.0000
SP4: The button is pressed sometime before the fifth interaction	0.0000	1.0000

Table 4.1: Some example system properties

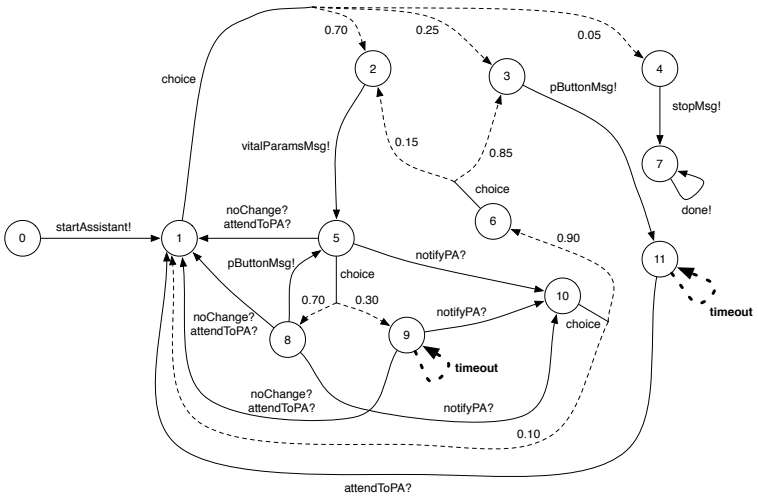


Figure 4.2: An initial environment for the TA system

$\{4, 8, 12\}$). In these cases, *actionCount* is an additional variable that tracks the number of interactions carried out in the TeleAssistance system. Alternatively, SP2 was modelled by an observer automaton that monitors the property.

We now show how to model the probabilistic behaviour of the environment using Probabilistic Interface Automata, and how such model and the theory presented in previous Chapters allow meaningful quantification of the probability of critical failures based on the modelled probabilistic assumptions of the environment.

4.2. Modelling the Environment

In Figure 4.2 we depict a first attempt at modelling the probabilistic behaviour of the environment of the TeleAssistance software. This environment, when waiting for a vital parameters analysis response, probabilistically chooses to wait patiently, or press the panic button. Also, it reflects a certain degree of anxiety in the patient's behaviour, since it behaves quite differently depending on whether the software determines to adjust her medication or not. If the medication is not adjusted, the environment reverts to its usual behaviour, however, if the medication is indeed adjusted, the patient becomes more prone to pressing the panic button.

Although seemingly a reasonable model of this environment, this is not the case. It is straightforward to see that Figure 4.2 is a Probabilistic Interface Automaton. Further, this PIA is composable with the TA system model in Figure 4.1 (see Definition 3.3). However, the PIA depicted in Figure 4.2 is not a valid environment for the TA system model, as it allows reachable illegal states (see Definition 3.5). For exam-

ple, the composite state consisting of state 9 in the TeleAssistance system model; and state 9 of the environment is an illegal pair that is reachable in the parallel composition of both models via the trace: $(0_s, 0_e)$ `startAssistant` $(1_s, 1_e)$ `choice` $(1_s, 2_e)$ `vitalParamsMsg` $(5_s, 5_e)$ `analyseData` $(7_s, 5_e)$ `choice` $(7_s, 8_e)$ `pButtonMsg` $(8_s, 5_e)$ `choice` $(8_s, 9_e)$ `failedAlarm` $(9_s, 9_e)$. We have suffixed each state with either e or s to make clear whether we refer to the environmental or system state respectively.

The fact that $(9_s, 9_e)$ is an illegal state highlights that the environment is making incorrect assumptions on the behaviour of the system and renders the probabilistic environment behaviour modelled meaningless. For instance, analysing the behaviour of the probabilistic environment it is easy to conclude that the probability of sending a `vitalParamsMsg` to the system as the next message if being at state 9_e is at most 0.7, and at least 0.205. Note that the upper bound is obtained if the `noChange/attendToPA` transition is followed, while the lower bound is the result of the sum of the possible outcomes of taking the `notifyPA` transition.

However, the same analysis on the (potential) product results in a probability inconsistent with the analysis on the environment alone. The inconsistency is that while on the environment the lower bound for the property was 0.205, the lower bound was decreased to zero (rather than increased) when composed with the system. The increase of the lower bound is due to the fact that the environment's behaviour specified in the environment's state 9_e is restricted when the system is in its own state 9_s, hence the environment probabilistic contribution that outgoing transitions from 9_e made to the lower bound of the property are no longer possible. However, this particular environment fails to makes a provision in modelling the possibility of such a restriction.

In summary, if the analyses performed to validate the probabilistic behaviour of the environment are not valid once the environment is composed with the software, then the model of the environment has a limited, if any, potential for sound analysis. The definition of legal environment, which the model in Figure 4.2 does not satisfy, is aimed to guarantee sound analysis.

We could, however, produce a legal environment for the TA system by slightly modifying the current one. For example, a possible solution is to add `timeout` transitions from states 9_e and 11_e (denoted with thick dotted lines), modelling that the environment can give up waiting for the software response, concluding that it has probably crashed in some way. That is to say, the previous model of the environment was establishing very strong assumptions on the system; the environment required the system to *always* generate an input at these states. This assumption, which turns out to be wrong, results in an illegal environment as it generates illegal states in the composition–see condition (II) in Definition 3.5.

The probability of sending a `vitalParamsMsg` starting from state 9_e now evaluates to the interval $[0, 0.7]$ in this legal environment. This is consistent with the evaluation of the property when composing the legal environment with the software. In fact, due to Theorem 3.1 we know that any property that has been used to validate the probabilistic behaviour in this legal environment will be preserved in its composition with the software. Asserting the validity of the conditions for legal environments essentially entails verification of several liveness properties, which all check out in this case.

In a similar way as we did for the system model, we also evaluated the probability of some properties over the environment in isolation, as depicted in Table 4.2. In the case of the environment, all of the properties were modelled by suitable observer automata.

Environment property	P_{min}	P_{max}
EP1: A `changeDrug` or `changeDose` is received, and the next user action is a button press	0.0000	0.7373
EP2: The button is pressed during the first interaction	0.2500	0.9281
EP3: The button is pressed before the fifth interaction	0.2500	0.9281

Table 4.2: Some example environment properties

4.3. Quantitative Analysis of the TeleAssistance System

Now that we have a legal environment for the TA software, we can quantitatively analyse the behaviour of the TA software system by checking the probability of system properties holding when the TA software is composed with the legal probabilistic environment.

We will now consider all the properties that we already analysed in Tables 4.1 and 4.2. In order to have a complete analysis, we first computed the product of the Interface Automaton for the TA software and the Probabilistic Interface Automaton modelling the environment. We later used PRISM to quantify the probabilities of the events described, and the results are comprised in Table 4.3.

Consider the failure property that states that the First-Aid Squad may not be sent to the patient location whenever the alarm has been raised (property SP1). Clearly, the TeleAssistance software may realise this failure (see the transition from state 8 to state 9 in the system model, which we already discussed). However, it is interesting to quantify the probability of such error under the assumption of a particular probabilistic behaviour of the environment. When we quantitatively analysed the TA system behaviour in isolation, we could only say that there exist execution traces that exhibit the failure with probability 0 (i.e., they always avoid it); and others that realise the failure with probability 1. Note that this analysis assumes schedulers that can non-deterministically choose to stop operation, or that can avoid pressing the panic button. However, looking at the environment it is clear that there is a non-zero chance that the button will be pressed, even in the first interaction with the system. This suggests that the minimum probability of failure of the system composed with its environment is actually greater than zero. In fact, recall that the probability of property SP1, when analysed over the system only, was found to lie in the $[0, 1]$ interval. Once we analyse the composition however, we find that it actually lies in the $[0.2, 0.9057]$ interval. This is consistent with the preservation theorem (Theorem 3.1). The minimum probability is being raised as a consequence of the fact that, for the environment, there is always a nonzero chance that it will press the button. For example, it may be the first action it takes, with probability 0.25. Then, the system can fail with probability 0.05. This seems to suggest that the minimum probability of failure is $0.25 \times 0.05 = 0.00125$, and not 0.2. However, this is only taking into account just one interaction; once the infinite possibilities are compounded we get to the obtained 0.2 value. Similarly, the maximum is diminished from 1 to 0.9057 since the system cannot force the environment to always press the button.

Similar results are obtained for the other properties, where the interaction between environment and system makes some choices unenforceable, thus restricting the possible probability values. The case of properties EP2 and EP3 is noteworthy, since the probabilities do not change at all from the isolated environment analysis to the composed one. However, this is a natural consequence, since the decision of pressing the panic button is completely governed by the environment. The system

Property	Composite probs.		Component probs.	
	P_{min}	P_{max}	P_{min}	P_{max}
SP1: The button is pressed yet the First-Aid Squad is not sent to the patient location	0.2000	0.9057	0.0000	1.0000
SP2: A `changeDrug` or `changeDose` occurs, and the next message received by the TA generates an alarm which fails	0.0000	0.7982	0.0000	0.9000
SP3: The button is pressed during the first interaction	0.2500	0.7400	0.0000	1.0000
SP4: The button is pressed sometime before the fifth interaction	0.2500	0.9489	0.0000	1.0000
EP1: A `changeDrug` or `changeDose` is received, and the next user action is a button press	0.0000	0.7252	0.0000	0.7373
EP2: The button is pressed during the first interaction	0.2500	0.9281	0.2500	0.9281
EP3: The button is pressed before the fifth interaction	0.2500	0.9281	0.2500	0.9281

Table 4.3: Properties' probabilities for the composite system

Source state	Original μ	Variant TA_1	Variant TA_2
1	$\{0.95 \mapsto 4, 0.05 \mapsto 8\}$	$\{0.99 \mapsto 4, 0.01 \mapsto 8\}$	$\{0.75 \mapsto 4, 0.25 \mapsto 8\}$
5	$\{0.90 \mapsto 4, 0.10 \mapsto 8\}$	$\{0.95 \mapsto 4, 0.05 \mapsto 8\}$	$\{0.60 \mapsto 4, 0.40 \mapsto 8\}$
7	$\{0.50 \mapsto 4, 0.50 \mapsto 8\}$	$\{0.75 \mapsto 4, 0.25 \mapsto 8\}$	$\{0.20 \mapsto 4, 0.80 \mapsto 8\}$
11	$\{0.10 \mapsto 12, 0.90 \mapsto 8\}$	$\{0.20 \mapsto 12, 0.80 \mapsto 8\}$	$\{0.01 \mapsto 12, 0.99 \mapsto 8\}$

Table 4.4: TeleAssistance distribution variants

cannot either block the environment from pressing the button, nor can it force the environment to press it.

The (rather high) value of the maximum probability of SP1 (alarm failure) is sensitive to the probabilistic behaviour of both the environment and the TeleAssistance system. The probabilistic distributions in states 1, 5, 7 and 11 on the TeleAssistance system; and states 1, 5, 6 and 10 all contribute to this probability. Varying the probabilities on these transitions has an impact on the probabilistic system behaviour.

To better understand this impact, we built some variants of both the TeleAssistance model as well as the patient model, by varying these distributions' probabilities. Table 4.4 and 4.5 summarise these variants. We calculated the probability of property SP1 over the composite system for each combination of these variants. These

Source state	Original μ	Variant E_1	Variant E_2
1	$\{0.70 \mapsto 2, 0.25 \mapsto 3, 0.05 \mapsto 4\}$	$\{0.60 \mapsto 2, 0.15 \mapsto 3, 0.25 \mapsto 4\}$	$\{0.50 \mapsto 2, 0.05 \mapsto 3, 0.45 \mapsto 4\}$
5	$\{0.70 \mapsto 8, 0.30 \mapsto 9\}$	$\{0.50 \mapsto 8, 0.50 \mapsto 9\}$	$\{0.30 \mapsto 8, 0.70 \mapsto 9\}$
6	$\{0.15 \mapsto 2, 0.85 \mapsto 3\}$	$\{0.50 \mapsto 2, 0.50 \mapsto 3\}$	$\{0.85 \mapsto 2, 0.15 \mapsto 3\}$
10	$\{0.90 \mapsto 6, 0.10 \mapsto 1\}$	$\{0.50 \mapsto 6, 0.50 \mapsto 1\}$	$\{0.10 \mapsto 6, 0.90 \mapsto 1\}$

Table 4.5: Patient distribution variants

	TA_1	TA_2
E_1	$P_{min} = 0.0059$ $P_{max} = 0.5543$	$P_{min} = 0.1304$ $P_{max} = 0.6144$
E_2	$P_{min} = 0.0011$ $P_{max} = 0.2425$	$P_{min} = 0.0270$ $P_{max} = 0.2763$

Table 4.6: Evolution of probabilities for SP1 with different distribution variations

results are depicted in Table 4.6. Not surprisingly, the major factor in decreasing the probability of failure is reducing the probability of the patient pressing the button, either by decreasing that probability itself, or increasing the probability of stopping the interaction with the system.

Summarising, in this Chapter we have shown how Probabilistic Interface Automata supports quantitative analysis of non-deterministic models. The notion of legal environment (and related theorems) is crucial, since it constrains the acceptable models of the probabilistic behaviour of the environment to those that ensure that analysis performed to validate the environment's probabilistic behaviour is sound and preserved when analysing the composite system.

CHAPTER 5

Discussion

In the last few decades, researchers have paid attention to the concept and consequences of operational profiles in system reliability specification and analysis [Che80, Mus93].

Regarding enriching models with probabilistic information, we can mention the work in [RM04, EGMT09]. This work, unlike our own that allows for composite-level modelling, yields a verification artefact that is a single model containing all the relevant probabilistic transition information, both pertaining to the environment and to the system. Our approach has the added benefit of allowing the engineer to isolate each component, and only add probabilistic information to the source where it has already been validated.

Additionally, the Markov models such as those obtained in [RM04, EGMT09] are purely probabilistic, which may not allow the engineer to fully model the non-deterministic behaviour of concurrent systems. This is an issue especially in the case where a system is known to behave in different ways at the same point, but the choice between these different behaviours cannot be properly quantified.

The problem of being able to model both probabilistic and non-deterministic behaviour through a single, consistent formalism is not a new issue. For example, although *generative* models [Chr90] do not directly allow non-determinism themselves, an asynchronous parallel composition (à la CSP [Hoa78]) induces such non-determinism and must be dealt with, while preserving the intended behaviour of the components. Works such as [DHK99] advance in this direction resorting to redistributing probabilities when finding synchronising actions with no matching counterpart. It is unclear if this approach is suitable when the probabilities reflect system-environment interaction. The environment (in the most usual case, a user) may not actually redistribute probabilities on allowed action when the desired one is not allowed. Regarding *reactive* models [vGSS95] we have already discussed the limitations they pose towards realising our modelling goals. This discussion can be found in Section 3.1.1.

It must be noted as well that an important precedent to this work is that of Probabilistic I/O automata [WSS97]. This model enriches classic I/O automata [LT87] with probabilities, establishing a hybrid between the generative and reactive models, since output actions are modelled in a generative way while input actions are modelled reactively. The approach in itself is interesting, but the probabilistic I/O automata model has some characteristics we consider problematic. In the first place,

it inherits from I/O automata the notion of input-enabledness. Under this paradigm, every component automaton, at every state, must allow every possible input as a transition. As we previously argued, this is not a realistic restriction in most cases, since systems are usually designed with some concept of the environment in mind, and thus it is reasonable that they restrict some inputs at certain points of execution. Another characteristic aspect of probabilistic I/O automata is that they introduce a real-valued parameter to each state in each component automaton. This parameter, an additional random variable as it happens, models a *delay* on each automaton state. The rationale for this delay is the need to somehow resolve conflicting races, since at some points of the asynchronous concurrent execution, it would be feasible for more than one component to synchronise its actions. This delay is intended to establish an order in which the automata advance, that is, the automata in which the state delay is the least will advance first. Since the delay variable is random, this allows this order to also be random.

The notion of resolving races between competing transitions is also present in our model, as in other proposed models [SdV04]. However, this choice is represented by an external entity, the scheduler. The scheduler, however, can be seen as a process that is completely independent of the system model itself; while the system behaves independently of the scheduler as well. Additionally, the notion of a scheduler models an unknown within the system under analysis. That is, it models a behaviour that cannot be explicitly quantified. The I/O automata notion of delay defeats this modelling objective. In this sense, we argue that the idea of a built-in scheduler as a composite aspect of the system model–be it probabilistic or not–is undesirable, as we aim for a separation of concerns.

Finally, a behaviour composability result is presented for probabilistic I/O automata, though it is different to the one we present in this thesis. Probabilistic I/O automata behaviour preservation stems from that of the original non-probabilistic I/O automata. This result states that every execution trace in the composite automata, when restricted to the actions of each component automaton, is an execution trace of said component automaton. However, this result leverages heavily on the embedded scheduler concept depicted above. Our result does not establish such a stringent relation, since we establish that system-environment composition does refine the specified behaviour, but observed probabilistic behaviour in the environment is still preserved, thus allowing for early elicitation of interesting properties.

Apart from modelling system behaviour by means of synchronising automata, there have also been advances in quantitative contract-based modelling or, in a similar fashion, quantitative assume-guarantee reasoning. The work by Delahaye et al. [DCL11] presents a contract-based approach that shares many similarities with the work we present in this thesis. In particular, both this work as well as ours aim at presenting a formalism that can reason about isolated components in the context of a composite systems.

There exist two key differences between the approach presented here and that of [DCL11], which allow both techniques to be used complementary. First, the work in [DCL11] analyses contracts in isolation, and results in a lower bound for the probability of satisfying the contract that results of the composition of these contracts. Our approach is also intended for the isolated analysis of components; however we introduce a notion of preservation of behaviour properties rather than bounds. Second, the object of study is very different in both cases. Our work deals with automata-like description of behaviour, while Delahaye et al. deals with contracts which are represented by sets of traces. This allows them to define composition and conjunction between systems (by composing or conjoining their contracts), while also allowing for

a notion of refinement between systems (that is, contracts that refine other contracts that otherwise allow less or require more). This marks another few differences with our work, as we do provide the notion of composition, but where conjunction does not have a direct analogue. However, our choice of automata as models allows for explicit representation of non-deterministic choices. Although this choice hampers the possibilities of properly defining a notion of refinement, it allows a larger degree of expressibility than that of the contracts of Delahaye et al. In that sense, our approach is closer to modelling formalisms such as Segala's Probabilistic Automata and Markov Decision Processes than those of contracts.

There is also work on assume-guarantee verification of safety properties, which have some similarity to our own. The work of Kwiatkowska et al. [KNPQ10] is noteworthy. In that work the authors model probabilistic systems through probabilistic automata much like those presented here, and aim at the verification of safety properties modelled via deterministic automata. [HKK13] also presents an assume-guarantee approach where the object of study are Interactive Markov Chains [HK09]. However, in all these cases there is no notion of preservation of behaviour through compositional construction.

The notion of refinement in automata-based formalisms is related to that of simulation (and bisimulation). Since our Probabilistic Interface Automata are a restricted case of Segala Simple Probabilistic Automata [Seg95], the notion of (bi)simulation is well-defined. However, bisimulation can be too strict, and not an effective notion, in the presence of components with internal computation that needs to be abstracted away. In regards to this question, the notion of *weak bisimulation* [Mil89] has been employed effectively in the context of non-probabilistic systems. Such a notion of weak bisimulation has been recognised, although it is problematic for probabilistic systems [HJ90, SJ90]. We do not go into detail in these aspects, however some interesting work includes [BH97] where the authors present a weak bisimulation notion along with a decision procedure, albeit focused on fully probabilistic systems alone. Also, [SL95] introduces a notion of weak bisimulation for systems exhibiting non-determinism, where the bisimulation proposed includes the potential generation of infinite probabilistic distributions representing all possible intermediate internal steps. Philippou et al. [PLS00] and Cattani [CS02] attack this problem by restricting distributions to a certain class. In order to prove the behavioural preservation properties of Probabilistic Interface Automata, we have based our efforts on the notion of weak probabilistic branching simulations [Seg95]. It remains to be seen, however, if other simulation notions are just as suitable.

An important improvement relative to Interface Automata is also presented in the previous Chapters. This result regards the synchronising conditions for Interface Automata, and is independent of probabilities. We found the synchronising conditions posed by Interface Automata to be too strict regarding the immediate necessity for synchronisation. However, software systems that need to perform several internal actions before allowing inputs from its environment are commonplace. Such systems cannot be easily modelled with Interface Automata without abstracting away such internal behaviour, eliminating the possibility to document this potentially interesting behaviour, and possibly analyse it at a component level. When developing the Probabilistic Interface Automata formalism, we have relaxed the need for immediate synchronisation in these cases, while requiring a notion of fairness on the schedulers allowed for the composite system.

This decision on fairness restrictions, however, calls for further analysis. Although the fairness conditions imposed are not esoteric or overly restrictive, it may be the case that they can be refined and further relaxed. Preliminary analysis has shown

that the fairness requirement over some states may be relaxed in some cases–for example, loops made up purely of internal actions, that can be ignored if not allowed to happen–but a generalisation and proper characterisation remain as future work.

5.1. Conclusions and Further Work

Quantitative model checking and analysis are promising techniques to complement Yes/No automatic analyses of behaviour. This first Part of the thesis has dealt with some of the software engineering challenges that need to be solved to enable such a technology, namely, the incorporation of probabilities into system models lacking probabilistic information. This naturally raises several formal and practical challenges. These challenges range through several aspects: first, it is important that these probabilities be introduced in a component-wise fashion, as it is often difficult to establish the quantitative behaviour of the system at large. Second, this probability introduction should not interfere with the behaviour that was described previously, that is, it should not preclude previously modelled behaviour, nor otherwise allow for emergent behaviour that was not modelled before. Finally, the introduction of probabilities should be in such a way that component-wise verified properties still make sense, and hold, once the whole model is built as a composition of these components. That is, the formal model and composition must preserve the meaning of annotations in both of the existing and composed artefacts.

The key to these challenges is a careful treatment of controllability of actions, non-determinism, and fairness assumptions over the behaviour of composite systems. We presented Probabilistic Interface Automata as a suitable formalism satisfying these requirements and showed that the language is compositional, that is, there is a notion of property preservation between the components and the composite system. Although we have preliminarily validated this approach, research on the generation of useful and sound environments is the focus of future and ongoing work.

Deeper understanding of fairness assumptions also merits further work. In the particular case of the work presented here, we have shown that a notion of strong fairness, relaxed for probabilistic behaviour, is sufficient to ensure compositionality of Probabilistic Interface Automata. However, it remains to be seen if such assumptions are completely necessary, or if they could be weakened. If so, further analysis is necessary for understanding under which conditions these assumptions may be weakened and what their impact is on modelling different environmental domains.

Part III

Partial exploration and evaluation of models

CHAPTER 6

Efficient partial verification

In this Part of the thesis we will focus on defining and solving the problem of obtaining feedback information from failed model checking efforts, as was described in the introductory part of the thesis. We will formalise the notion of partial state space, and will introduce our ideas for meaningful quantitative feedback.

After the introduction of these concepts, we will perform a preliminary validation of the approach. This experimentation, apart from resulting in a satisfactory sanity check of the technique, allowed us to identify further requirements towards an approach that can both scale in time and space, and also provide the practitioner with useful information.

As a result of the previous analysis, we motivate the approach in its present form. Further in this Chapter, we delineate the basis of a quantification and verification procedure that is suitable to our setting. The technique we present is tailored towards avoiding, or at least reducing, the problems that threaten the applicability of a straightforward quantification and analysis technique. The result of the work presented in this Chapter is a technique that comprises a combination of guided simulation, analysing the features of these simulated paths, property inference, and probabilistic model checking.

In addition, we extend the target of our work to reactive probabilistic systems in general, and no longer limit our work to the quantification of the partial state space of a non-deterministic system's behaviour. The workflow presented in this Chapter is applicable without modifications to any reactive probabilistic system model, either monolithic or compositionally built. Of course, the scenario that kicked off our research remains as a particular case of this, more general, analysis setting.

In the remainder of these chapters, we will specialise on Segala's Simple Probabilistic Automata [SL95, Seg95] as the reactive formalism of study (recall Definition 2.19). Note that since the Probabilistic Interface Automata presented in Definition 3.1 are Segala's Simple Probabilistic Automata themselves, this approach is applicable to our original research setting.

6.1. The problems with state-of-the-art techniques

As we have already discussed in the introduction of this thesis, applicability of model checking techniques for verification of properties of complex model is threatened by the sheer size of these models. Probabilistic model checking is no exception

to this, as this topological problem is still present in its resolution method. Even worse, techniques such as on-the-fly model building are not applicable in a probabilistic setting, since the numerical resolution part of the analysis requires the whole model to be built. Any attempt at reliability assessment of complex models through probabilistic model checking will suffer from these drawbacks.

Although state space reduction techniques exist [LLPY97, CGMP99], they may still fail to prevent state explosion to a manageable extent on sufficiently complex models. As if this was not discouraging enough, even in the event that the entire state space can be explored in its totality, its size typically impedes exact numerical calculation of reliability metrics through methods such as Gaussian elimination or the Grassmann, more stable, algorithm. To overcome this limitation, iterative methods (such as Jacobi or Gauss-Seidel) that approximate metrics need to be used. However, these methods do not always have convergence guarantees. In fact, even in the cases where they do converge, they may do so slowly; as much as to become intractable. The latter problem is heightened in the case of metrics related to rare events (e.g. reliability estimation for models where the probability of failure in a fixed period lies below 10^{-5}). In this case, since the execution budget time for the iterative methods is not infinite, exhausting this budget can lead to iterations being cut short far from the actual value of the metric being estimated. This becomes a problem for safety critical systems since, as the model is further refined and corrected, it is expected that the remaining errors will become rarer with every iteration.

In summary, although probabilistic model checking may seem to promise exact calculation of quantitative reliability properties, state space explosion and application of numerical methods can be computationally prohibitive or result in poor approximations. Despite these limitations, probabilistic model checking can provide bounds with 100% confidence for reliability metrics even though the distance of these bounds to the real value cannot be known in general.

Numerical analysis and, to some extent, state explosion can be avoided using statistical methods over many samples of the system. Variations of these approaches are usually referred to with the umbrella term of Monte Carlo estimations. When using these techniques to estimate quantitative metrics, the actual population mean X is approximated through an estimator such as the sample mean $\overline{X}$ [Lyu96]. Of course, such estimation is subject to statistical error and thus it is crucial to understand how far and with what likelihood the estimator deviates from the actual mean. This contrasts with probabilistic model checking, which does not suffer from such statistical imprecision.

The deviations from the actual value that result from the specific samples used while performing Monte Carlo based estimations is usually conveyed in terms of statistical errors and confidence intervals. Bounds for statistical error and confidence intervals can be computed, based partly on the number of samples being analysed and prior knowledge of the distribution of the events of interest (in particular its variance). Although significant progress for fast generation of random walks over models has been made [Nim10, RP09], sample generation can be very costly time-wise even for analyses with modest guarantee requirements, simply due to the sheer number of samples required [Saw03].

The number of samples required is not the only limiting factor for these approaches Sample-based reliability estimations must also take into account the length of samples. Sample length can be particularly problematic, since sampled executions must reach a state satisfying a (usually unlikely) property (e.g. a failure) in order to allow the computation of an estimator. This fact, compounded with the need for many samples, may turn sample generation for high-reliability systems intractable.

In summary, statistical techniques can provide approximations with measurable confidence intervals and error bounds. However, in the presence of models with rare events, the required number and length of samples may make such techniques intractable, and attempts to reduce either sample size or length might result in weakened (or downright lost) statistical guarantees over results.

In this Chapter we present an alternative to exhaustive model exploration—as in probabilistic model checking—and partial random exploration—as in statistical model checking—which may counter some of the limitations of existing model-based reliability verification techniques. Our hypothesis, inspired on the Pareto principle, is that a (carefully crafted) partial systematic exploration of system models can be effectively analysed to provide good bounds on quantitative metrics with lower computation cost. More specifically, probabilistic model checking of a submodel of the system can bound the value of these metrics for the complete model, and do so in a cost effective manner. Furthermore, it can produce better approximations, given equal time and memory budgets, than those that both probabilistic and statistical model checking can achieve.

We hypothesise that there is a gain to be had by identifying a small, but probabilistically significant, portion of the state space, considering all other states as failures and performing probabilistic model checking on the resulting submodel. The intuition is that, in contrast to full-model probabilistic model checking, performing a probabilistic check on only a portion of the full model allows for faster iterations of the numerical analysis methods. Consequently, more iterations can be performed within the same time budget and, for slowly converging models, a better approximation may be achieved.

More specifically, in this Part of the thesis we present a novel automated technique for quantitative metric estimation that combines simulation, invariant inference and probabilistic model checking. We use model simulation to produce a set of traces that represent likely behaviour of the full model. These traces are used to infer an invariant that describes the state space explored during the simulation. A submodel, which restricts the states by not allowing those that do not satisfy this invariant, is constructed and the value of the desired metric is computed over this partial model using a probabilistic model checker.

The technique we propose obtains lower bounds to the actual values of the desired metrics with 100% confidence (as full-model probabilistic model checking and in contrast to statistical model checking). In a more technical note, our technique provides a lower bound on the expectation of a random variable. This random variable is modelled as a reward structure over suitable probabilistic models. Our technique also provides bounds on the probability of a reachability property being satisfied.

In a subsequent Chapter, we will put the proposed approach to the test. As the results will show, the experimental evidence suggests that the lower bounds achieved (for a fixed budget of time and memory) are higher than those obtained by full model probabilistic and stochastic model checking, especially for models where the probability of reaching the interesting property is low given a fixed time. High bounds are of special interest in reliability, as they allow to argue a reliability case even in the absence of the exact values. Furthermore, automated invariant generation seems to perform reasonably well against domain-expert provided invariants, and have the added advantage of being useful when such expert-provided invariants are unavailable.

6.2. Approach

This section formally defines an approach to computing bounds to reachability probabilities and reward values of probabilistic system models. The approach is based on calculating this values for only a partial systematic exploration of the model's state space. We first define what is meant by a partial exploration and show that the mean reward computed over these partial explorations is indeed a lower bound to the mean reward computed over the entire system model. We also show that reachability probabilities computed over partial models are an upper bound to those that would be computed over the whole model.

We then show how some partial explorations can be specified declaratively through invariant properties that drive the exploration, discussing at length the details of the procedure. Finally, we show how these invariant-driven partial explorations can be obtained automatically from any given model, without need for human intervention. In the next section we will show, via some case studies, that given a fixed budget of time and memory, analyses performed over automatically inferred invariant-driven partial explorations perform at least as well as, and sometimes outperforms, partial explorations driven by manual specification.

6.2.1. Partial Explorations

We refer to a partial exploration of a system model as a submodel. Intuitively, a submodel of a probabilistic process M is a model that retains a subset of the states and transitions of M and in which all other states in M have been abstracted away into a new λ trap state. Moreover, the retained states include the initial state, and all other retained states are reachable from this initial state. Formally, the notion of a submodel of a probabilistic model is captured by the following definition.

Definition 6.1 (Submodels)**.** *Given a probabilistic model $M = \langle S, s_0, A, R\rangle$, a* submodel *of M is another probabilistic model $M' = \langle S' \cup \{\lambda\}, s_0, A, R'\rangle$ such that $S' \subseteq S$, $s_0 \in S'$, and $R' \subseteq (S' \cup \{\lambda\}) \times (A \cup \{\tau\}) \times \mathcal{D}(S' \cup \{\lambda\})$ is such that for all $a \in A$*

1. *for each $(\lambda, a, \mu_{R'}) \in R'$, it must be the case that $supp(\mu_{R'}) = \{\lambda\}$ and $a = \tau$;*
2. *for all $s \in S'$ and $a \in A \cup \{\tau\}$*
   a) *for all $\mu_{R'}$ such that $(s, a, \mu_{R'}) \in R'$, there exists μ_R such that* i) *$(s, a, \mu_R) \in R$,* ii) *for all $s' \in S'$ $\mu_{R'}(s') = \mu_R(s')$, and* iii) *$\mu_{R'}(\lambda) = 1 - \sum_{s' \in S'} \mu_R(s')$.*
   b) *for all μ_R such that $(s, a, \mu_R) \in R$, there exists $\mu_{R'}$ such that* i) *$(s, a, \mu_{R'}) \in R'$,* ii) *for all $s' \in S'$ $\mu_{R'}(s') = \mu_R(s')$, and* iii) *$\mu_{R'}(\lambda) = 1 - \sum_{s' \in S'} \mu_R(s')$.*

Clause 1 states that transitions originating on the λ state all lead back to the same λ state, and that they do so through the model's internal action τ. Clause 2 states that action transitions on the submodel are drawn from the original model ones, that is, if an action transition is possible at a given state in the submodel, that action must have been possible from the same state in the whole model. Further, it also states that the probabilities on those transitions are also preserved from the original model, except for the case of those that were rerouted to the λ state, which accumulates the probabilities of those rerouted transitions. Finally, Clause 2 states that every transition on the original model is preserved on the submodel for each of the states present in the submodel, while the λ states accumulates the remaining probability.

There is a close relationship between the schedulers that can be defined for a given model M and those that can be defined on its submodels M'. Intuitively, any

scheduler σ for M is still a valid scheduler for M', although with some changes. In particular, transitions that over the original model traverse to states that do not exist in the submodel are instead rerouted to the λ state. The following definition captures these changes.

Definition 6.2 (Restricted schedulers)**.** *Let $M = \langle S, s_0, A, R\rangle$ be a probabilistic model, and $M' = \langle S', s_0, A, R'\rangle$ one of its submodels. Let σ be a scheduler for M. Also, let $\alpha \in execs^*(M')$ which implies that either $\alpha \in execs^*(M)$ or $last(\alpha) = \lambda$. The* restriction *of scheduler σ to M' is another scheduler σ' for M' such that*

- *if $last(\alpha) = \lambda$ then $\sigma'(\alpha) = (\tau, \mu)$ where μ is such that $supp(\mu) = \{\lambda\}$.*
- *if $last(\alpha) \neq \lambda$ and $\sigma(\alpha) = (a, \mu)$ and $(a, \mu) \in R'(last(\alpha))$, then $\sigma'(\alpha) = (a, \mu)$.*
- *if $last(\alpha) \neq \lambda$ and $\sigma(\alpha) = (a, \mu)$ and $(a, \mu) \notin R'(last(\alpha))$ then it must be the case that, because of Definition 6.1, there must exist $(a, \mu') \in R'(last(\alpha))$ such that*
  - *$(supp(\mu') \setminus \{\lambda\}) \subseteq supp(\mu)$;*
  - *for each s' in $supp(\mu) \cap supp(\mu')$ it holds that $\mu(s') = \mu'(s')$;*
  - *$\lambda \in supp(\mu')$ and is such that $\mu'(\lambda)$ captures the remaining probability.*

  In such cases, $\sigma'(\alpha) = (a, \mu')$.

We also say that σ' is the scheduler σ restricted to *M'.*

It is also easy to see that any scheduler for a submodel can be extended to a scheduler that is valid for the complete model—in fact, it can be extended to possibly many schedulers. In other words, every valid scheduler for a submodel is a restriction of one or more schedulers of the complete model.

Submodels are key to our approach since they conservatively approximate the value of both probabilities and reward structures for reachability properties. Even though we restrict ourselves to reachability properties, this more than suffices for our intended verification setting. For example, consider the mean time to failure metric. In order to be able to calculate this metric, we first need to be able to describe what a failure means in our system. In other words, we need to identify which system states model a failure, or an irrecoverable situation. In the setting of this work, these states would comprise the interesting S_{reach} set. Calculating the mean reachability reward value to this S_{reach} set effectively calculates the mean time to failure of the system.

Expressing this bounding property more formally, given a reward structure ρ for a model M and a scheduler σ, the mean reward value of ρ under σ for M until reaching some state in a distinguished set $S_{reach} \subseteq S$ is always greater or equal to the mean reward value of any of its submodels M', under the same scheduler restricted to M', until reaching a state in the set $S'_{reach} = (S_{reach} \cap S') \cup \{\lambda\}$.

On a similar note, submodels also bound reachability probabilities, both for time unbounded reachability (i.e., formulae of the form $\phi U \psi$) as well as bounded reachability (i.e. $\phi U^{\leq t} \psi$). However, as we will see later on when performing experimental validation, this is not as useful as bounding rewards. The following two theorems express this in a formal way. We provide the proof for the case of rewards, but the proof follows the exact same argument for the case of probabilities.

Theorem 6.1 (Submodels bound reward values)**.** *Let M, M', S, S', S_{reach}, S'_{reach}, σ and σ' be defined as in Definition 6.2. Then $\overline{X_{reach}(S'_{reach}, M', \sigma')} \leq \overline{X_{reach}(S_{reach}, M, \sigma)}$.*

Proof. Note that, for every trace in the complete model, it either exists completely in the submodel, or the submodel contains only a prefix that is extended by the λ state. Since reward structures are based on transitions, every trace in the full model accumulates *at least* as much reward to each of the interesting states (possibly ∞) as the corresponding trace (or prefix) in the submodel. Hence these prefixes contribute to $\overline{X_{reach}(S'_{reach}, M', \sigma')}$ at most what their extensions in M contribute to $\overline{X_{reach}(S_{reach}, M, \sigma)}$.

Alternatively, if the submodel allows a trace that never reaches either λ or one of the target states in $S' \cap S_{reach}$, then this trace also exists in the complete model. In such a case, both $\overline{X_{reach}(S_{reach}, M, \sigma)} = \overline{X_{reach}(S'_{reach}, M', \sigma')} = \infty$. □

Theorem 6.2 (Submodels bound reachability probabilities)**.** *$M = \langle S, s_0, A, R\rangle$ and $M' = \langle S', s_0, A, R'\rangle$ be two probabilistic models with state spaces S and S' and such that M' is a submodel of M. Let $S_{reach} \subseteq S$ be a set of states representing the interesting events and σ a scheduler for M. Also, let σ' be the restriction of σ to M. Then, the following holds for every $p, q \in [0, 1]$ such that $q \leq p$*

$$M', s_0, \sigma' \models P_{\leq q}(true\, U \mathbb{1}_{S'_{reach}}) \implies M, s_0, \sigma \models P_{\leq p}(true\, U \mathbb{1}_{S_{reach}})$$

where $\mathbb{1}_B : B \to true, false$ denotes the indicator function *of set B, that is, the function that returns true if and only if its argument is in set B.*

Proof. The proof for the theorem bounding probabilities is analogous to that of reward bounding, although it must be noted that probabilities, as opposed to rewards, decrease the longer the execution fragment is extended by the scheduler. □

The above results entail that if computing the value of either a reachability probability or a reward structure for a system model is intractable, it can be conservatively approximated on any of its submodels. In the case of Segala's Simple Probabilistic Automata, because of the presence of non-determinism, it is interesting to examine the case for the extrema schedulers. The following corollaries captures the bounding relation for these extreme values.

Corollary 6.1 (Extreme rewards bounding)**.** *Let the probabilistic model M as defined in the previous theorem, and its submodel M', be SPAs. Let σ_{min} and σ_{max} be two schedulers for M such that, for any other scheduler σ for M*

- $\overline{X_{reach}(S_{reach}, M, \sigma_{min})} \leq \overline{X_{reach}(S_{reach}, M, \sigma)}$*; and*
- $\overline{X_{reach}(S_{reach}, M, \sigma_{max})} \geq \overline{X_{reach}(S_{reach}, M, \sigma)}$*.*

In turn, let σ'_{min} and σ'_{max} be schedulers for M' such that for other schedulers σ' for M' it holds that

- $\overline{X_{reach}(S'_{reach}, M', \sigma'_{min})} \leq \overline{X_{reach}(S'_{reach}, M', \sigma')}$*; and*
- $\overline{X_{reach}(S'_{reach}, M', \sigma'_{max})} \geq \overline{X_{reach}(S'_{reach}, M', \sigma)}$*.*

Under these conditions, it holds that $\overline{X_{reach}(S'_{reach}, M', \sigma'_{min})} \leq \overline{X_{reach}(S_{reach}, M, \sigma_{min})}$ and also that $\overline{X_{reach}(S'_{reach}, M', \sigma'_{max})} \leq \overline{X_{reach}(S_{reach}, M, \sigma_{max})}$.

Corollary 6.2 (Extreme probabilities bounding)**.** *Let the probabilistic model M be as defined in the previous theorem, and its submodel M', be SPAs. Let p_{min}, p'_{min}, p_{max}, p'_{max} all lie in the interval $[0, 1]$; let σ_{min} and σ_{max} be two schedulers for M such that, for any other scheduler σ for M*

- $M, \sigma_{min}, s_0 \models P_{\leq p_{min}}(true\, U \mathbb{1}_{S_{reach}})$;
- $M, \sigma_{min}, s_0 \not\models P_{\leq q}(true\, U \mathbb{1}_{S_{reach}})$ *for any other* $q < p_{min}$;
- $M, \sigma, s_0 \not\models P_{\leq q}(true\, U \mathbb{1}_{S_{reach}})$ *for any other* $q < p_{min}$;

for the minimum probability case, and also for the maximum probability:

- $M, \sigma_{max}, s_0 \models P_{\geq p_{max}}(true\, U \mathbb{1}_{S_{reach}})$;
- $M, \sigma_{max}, s_0 \not\models P_{\geq q}(true\, U \mathbb{1}_{S_{reach}})$ *for any other* $q > p_{max}$;
- $M, \sigma, s_0 \not\models P_{\geq q}(true\, U \mathbb{1}_{S_{reach}})$ *for any other* $q > p_{max}$;

In turn, let σ'_{min} *and* σ'_{max} *be schedulers for* M' *such that for other schedulers* σ' *for* M' *it holds that*

- $M', \sigma'_{min}, s_0 \models P_{\leq p'_{min}}(true\, U \mathbb{1}_{S'_{reach}})$;
- $M', \sigma'_{min}, s_0 \not\models P_{\leq q}(true\, U \mathbb{1}_{S'_{reach}})$ *for any other* $q < p'_{min}$;
- $M', \sigma', s_0 \not\models P_{\leq q}(true\, U \mathbb{1}_{S'_{reach}})$ *for any other* $q < p'_{min}$;

for the minimum probability case, and also for the maximum probability:

- $M', \sigma'_{max}, s_0 \models P_{\geq p'_{max}}(true\, U \mathbb{1}_{S'_{reach}})$;
- $M', \sigma'_{max}, s_0 \not\models P_{\geq q}(true\, U \mathbb{1}_{S'_{reach}})$ *for any other* $q > p'_{max}$;
- $M', \sigma', s_0 \not\models P_{\geq q}(true\, U \mathbb{1}_{S'_{reach}})$ *for any other* $q > p'_{max}$;

Under these conditions, it must hold that $p_{min} \leq p'_{min}$ *and analogously* $p_{max} \leq p'_{max}$.

Proof. Again we prove the case only for the bounding of rewards, and note that the proof for probabilities is analogous.

The proof stems directly from the proof of Theorem 6.1. The case for σ_{max} is straightforward. Suppose that $\overline{X_{reach}(S'_{reach}, M', \sigma'_{max})} > \overline{X_{reach}(S_{reach}, M, \sigma_{max})}$. Recall that, because of the definition of restricted schedulers, it must be the case that every trace generated by σ' in M' either exists as it is in M, or else it diverts to λ at the end. In any case, traces in M' cannot accumulate more reward in M' than they would accumulate in M, therefore such a situation is not possible.

By the same argument, let σ' be the scheduler obtained by restricting σ_{min} to M'. By the previous theorem, it must happen that $\overline{X_{reach}(S'_{reach}, M', \sigma')} \leq \overline{X_{reach}(S_{reach}, M, \sigma_{min})}$. Since σ'_{min}, by definition, yields a lower reward, it must be that $\overline{X_{reach}(S'_{reach}, M', \sigma'_{min})} \leq \overline{X_{reach}(S_{reach}, M, \sigma_{min})}$. □

In a similar manner as Theorems 6.2 and 6.1, these results indicate that *i)* estimations for the minimum and maximum reachability probabilities over a submodel yield an upper bound to the actual minimum and maximum probabilities; and in the case of rewards, that *ii)* estimations for the minimum and maximum rewards over a submodel yield lower bounds for the actual minimum and maximum rewards, respectively, for the whole model.

Key questions are which submodels are cost-effective (i.e. provide good approximations at reasonable computation cost) and how to find them. Another important question to address is whether effective submodels provide reasonable approximations in general. With this objective in mind, we first validate our submodel ideas over simple partial explorations of the full model.

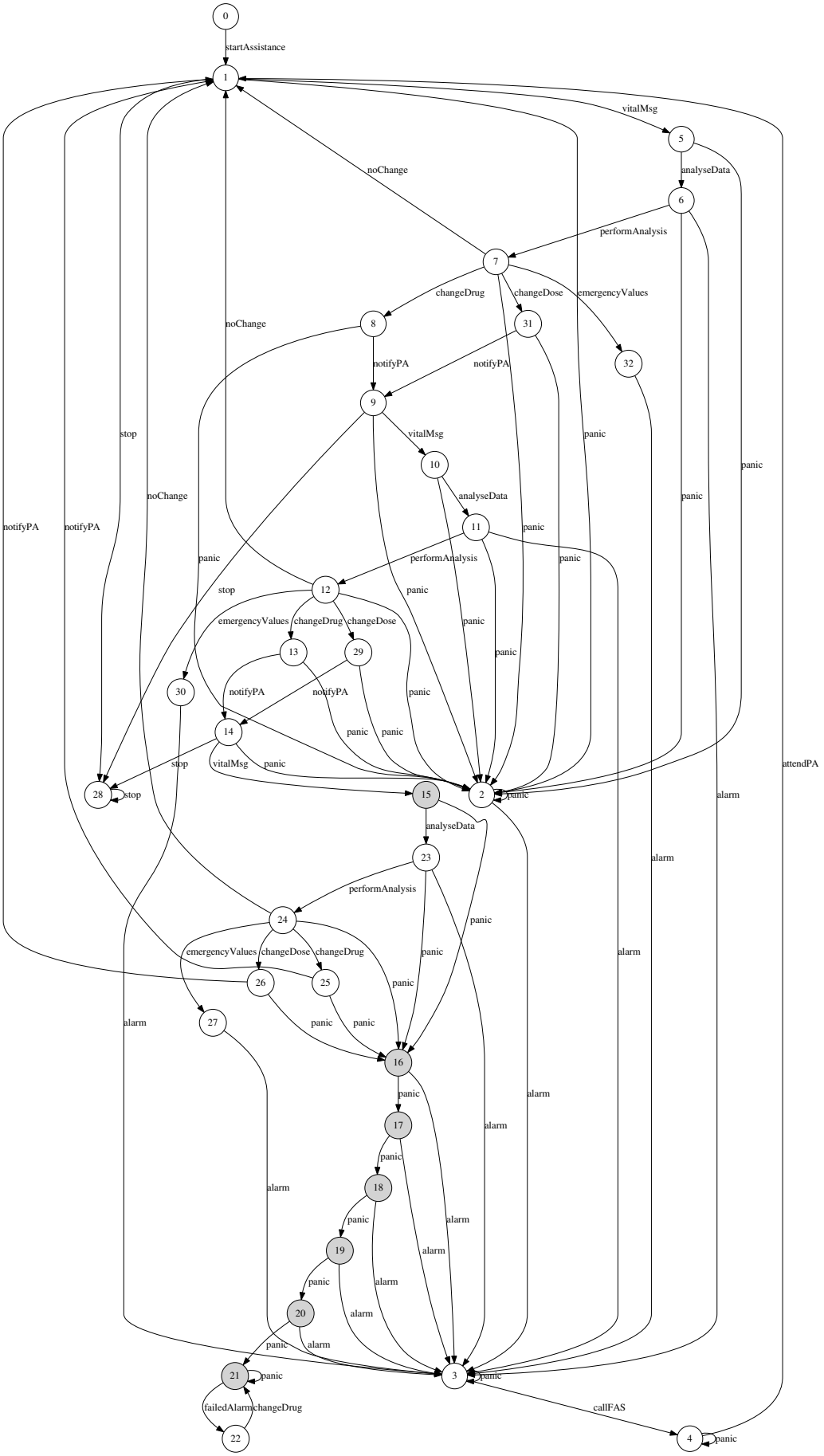


Figure 6.1: The degraded TeleAssistance software model

6.2.2. Preliminary submodel evaluation

In this section we set out to validate our approach against very simple submodels. We will perform this preliminary experimentation over a variation of the TeleAssistance software [EGMT09] presented in Chapter 4. An abstraction of this version of the software is depicted in Figure 6.1.

In this variation, the system does not inform the user whether it changed either the administered drug or its dosage. Another important difference is that, when the patient presses the panic button while the data is being analysed (see state 15) the system, rather than fail outright, enters a degraded mode. In this mode, the system is bound to fail, but it is somewhat more resilient than in the previous case. Once the system has entered this degraded mode, it might safely raise the alarm. However, if the patient persists in notifying panic before the alarm is raised, the system will eventually fail. More specifically, it will fail if it does not raise the alarm while the patient triggers five additional panic signals (see states 16 through 21). The actual model is much larger than what we can show here, since it has several other degradation modes built in. The triggering mechanism is similar in all of them, although it is raised at different moments in the execution.

We first compose this software model with a model of its environment, that is, a model of the patient's behaviour. This model can be seen in Figure 6.3. Again, this model is an abstraction of the patient's actual behaviour.

Note the behaviour highlighted in states 5 and 11. At this point, the patient has sent her vital parameters to the system, and is now waiting for the results. However, there is a probability that the patient will become uneasy and press the panic button. This behaviour may be repeated indefinitely while the patient is waiting for the system's response. Recall, however, that if the patient persists and presses the panic button five or more times before the system sends its response, the failure described above may be triggered.

We first calculated, using the PRISM model checker, the probability that the failure is eventually triggered. The minimum probability of failure is actually zero, since there are schedulers that can consistently avoid the degradation mode. Some of these schedulers represent, for example, a not very anxious patient. The maximum probability of failure was established to be 0.00005089.

For the sake of argument, let us assume now that a model checker fails to verify neither the minimum or maximum probabilities of the failure state being reached. So, we set out to validate our approach by generating some submodels from the full, composed model. We performed this generation by setting a bound to the number of states explored by the model checker. We further modified the model checker's exploration algorithm so that it would explore either in a breadth-first search (BFS) or depth-first search (DFS) order. The complete model spans 6717 states, and we generated submodels by setting the state space size bound to 600, 1800, 3100 and 4400 states.

Submodels constructed through a DFS exploration turn out to provide very bad bounds. In fact, in every case the probability of reaching either a failure state or the special λ state in these DFS-generated submodels turned out to be 1. Although this is a correct bound, it doesn't convey any information. The rationale behind the failure of these DFS-driven submodels is that, since DFS explorations prioritise exploring deeper in the model, they avoid traversing transitions very early in the model. As a result, schedulers that choose these transitions early on are very likely to reach the λ state quickly.

BFS explorations, on the contrary, do not suffer from this problem and perform better. Table 6.1 shows the results obtained for the BFS-driven submodels for the

Submodel size	Probability bound	Difference to actual probability
600	0.806334	0.80628311
1800	0.729199	0.72914811
3100	0.611195	0.61114411
4400	0.051490	0.05143911

Table 6.1: Estimated probability bounds for different submodel sizes (BFS explorations)

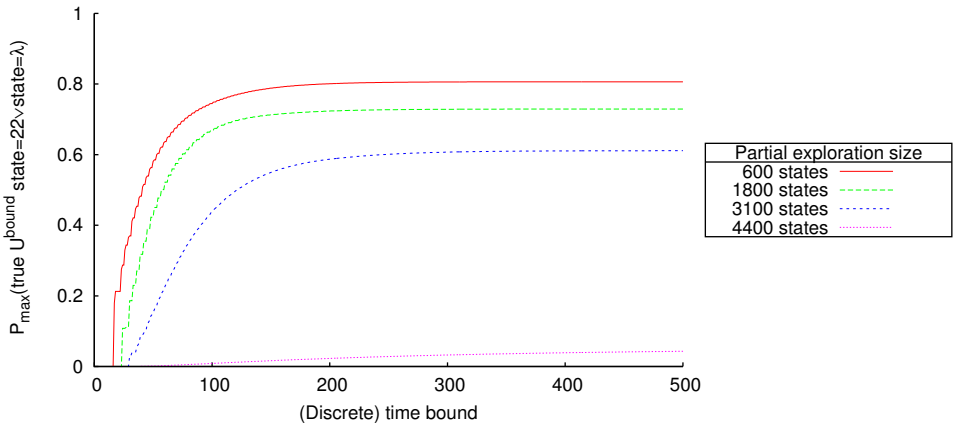


Figure 6.2: Preliminary evaluation of BFS-driven submodels

unbounded property $P_{max}(true\, U(state = 22 \vee state = \lambda))$, where 22 is the error state as seen in Figure 6.1. In turn, Figure 6.2 shows the progression of estimations of probability for the time-bounded property $P_{max}(true\, U^{\leq bound}(state = 22 \vee state = \lambda))$, which would eventually converge to the values shown in the Table. Different coloured lines represent the values obtained with differently sized submodels. The horizontal axis shows the progression on the *bound* variable used to bound the property, while the vertical axis shows the probability bounds obtained in each case.

It is clear that in the case of BFS-driven submodels, we can obtain some meaningful bounds, as we have bounded the failure probability to at most ~ 0.05. However, in order to get probability bounds closer to the actual probability it is necessary to have a BFS driven model of more than 4400 states. This comprises roughly 65% of the complete state space. In case of models that fail to be verified because of memory exhaustion, 65% of the total might still be unmanageable.

Two preliminary conclusions arise from this analysis. First, that not every submodel is created equal, and submodel size is not the only factor that comes into play. Some submodels may be able to provide useful bounds, whereas others of the same size will not be as effective. A second conclusion is that there is a need for an effective procedure to generate submodels that *i*) provide good bounds to the values of interest, and *ii*) provide a cost-effective way to approximate these bounds.

The preliminary evaluation seems to suggest that simplistic ways to drive the submmodel generation, such as standard BFS or DFS explorations, may not suffice

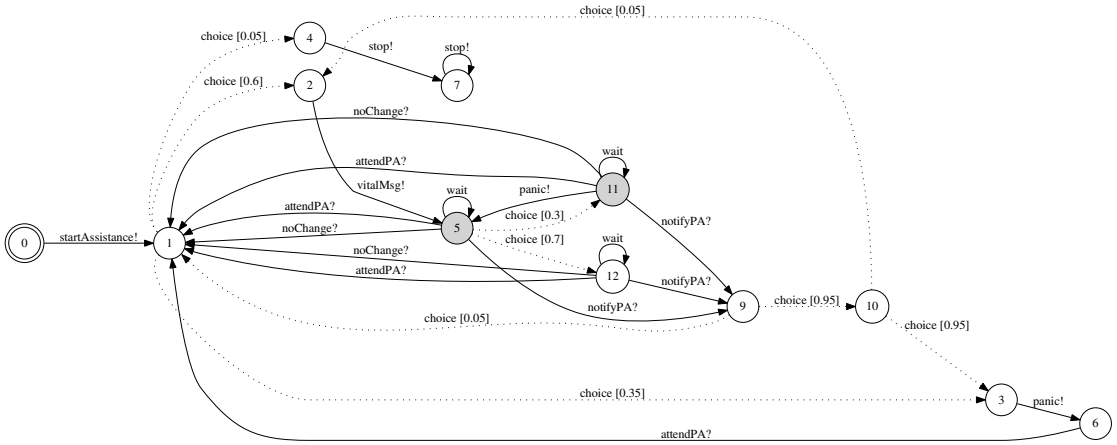


Figure 6.3: Patient behaviour model for the degraded TeleAssistance software

for these objectives. In the next subsection we discuss one particular way of driving the generation of submodels that results in cost-effective bounds computation. The key insight to this approach is that the semantics of the model under analysis must be taken into account in the submodel generation procedure. Later, in Chapter 7 we will argue that the submodels obtained through our approach are effective at estimating these bounds.

6.2.3. Automatic submodel generation

From the previous analysis we conclude that, although any submodel will provide a lower bound for the value of a given probability or reward structure, the key to a tractable estimation technique is to identify a submodel for which its values of interest can be computed within a reasonable time budget, and for which the resulting bound is a useful approximation to the actual value sought after in the full model. In the section above we have already shown results that hint that submodels obtained as the result of a depth-first search exploration are generally very bad at providing either good reward or good probabilities estimates. Conversely, submodels obtained through breadth-first search explorations seem to outperform those obtained through DFS, most likely due to the fact that they do not escape the explored space as quickly. Nevertheless, they still do not provide good estimates in general either. In other words, not all submodels are created equal; two submodels similar in size can obtain wildly different estimates.

Regrettably, and independently of the fact that the values of interest for the full model is unknown, the problem of computing an exact solution (i.e. obtaining the "best" submodel for the computation of an estimate) is intractable [JD07]. In this section we discuss a heuristic for automatically constructing submodels that can provide better bounds for reliability at lower computation cost than both full model checking and Monte Carlo approaches.

Our approach adopts a heuristic based on the reasoning that the submodel construction strategy should aim to identify a portion of the model that is probabilistically dense, that is, a submodel for which the probability of reaching the λ trap state in a given fixed time is low. More formally, a submodel M_1 of M is more probabilistically dense than another submodel M_2 if, for every $n \in \mathbb{N}$, the maximum probability of reaching the trap state λ in at most n steps in M_1 is at most as much as that probability in M_2. That is, $P_{max}(trueU^{\leq n}(state = \lambda))$ is lower in M_1 than it is in M_2.

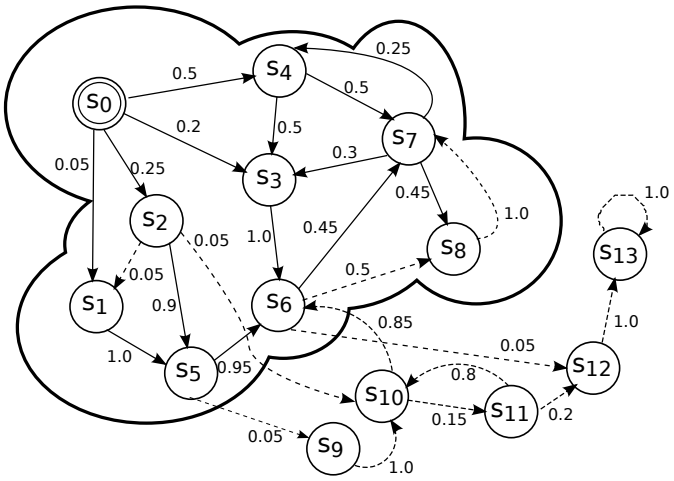


Figure 6.4: Example partial exploration of a state space

These probabilistically dense models will contain loops that are more probabilistically likely. These loops delay the traces from reaching the submodel boundary. Since reward structures are always positive and can never decrease, these loops contribute to a larger bound for the reward being estimated (or conversely, a smaller bound in the case of probability estimation).

The problem of finding the most probabilistically dense submodel is known to be NP-hard [JD07]. Our approach attempts to approximate such a submodel through bounded simulation. Hence, the basis of our approach involves the simulation of several traces over the full model. The resulting set of finite traces, if sufficiently large and consisting of sufficiently long traces, is likely to cover a good part of a probabilistically dense submodel. These traces form the basis for building our submodels. The smallest submodel that includes the set of states and transitions covered by the simulated traces can be constructed easily by simply adding any non-visited transitions between any two visited states, abstracting all non-visited states into the λ trap state, and adding transitions to the λ state for whichever state has transitions that were neither explored nor added in the first step. Figure 6.4 shows such a construction, where solid lines represent transitions that were covered by the simulated traces, while dotted lines are transitions in the model that were not covered. States outside the boundary have not been covered, and would be abstracted away into the λ state of the submodel.

However, submodels built through such a procedure are likely to have relatively short traces that escape the submodel (see path $s_0, s_2, s_{10}, \ldots$ in the figure). These short traces contribute a relatively high probability of escaping the submodel (in general, the shorter the prefix, the larger the probability of the set of traces that extend from it), reducing the bound estimated by the submodel. Note that, in our example, s_{10} falls back within the boundary to s_6 with high probability. If we were to include this state into our submodel, and according to the submodel completion procedure outlined before, the result would be that the bound estimated by the submodel would be raised. This is consistent with our experimentation in [PBU10]. In that work, we observed that submodels generated with a breadth-first search strategy tend to approximate reliability measures better, as they delay the chance of escaping traces until the lowermost levels of the breadth-first exploration.

In the approach that we detail in this present work, rather than adopting a syntactic notion of breadth first traversal for extending the submodel determined by a simulation of the full model, we take a more semantic approach based on

the attributes of states visited during the simulation. We compute state invariants based on the states visited during the simulation and then add to the submodel any states that satisfy the invariant, as well as the transitions between them. In this way, we expect to add behaviour that, although not exactly equivalent to what was simulated, represents variations in terms of symmetries, race conditions, and independent events [BK08], and contributes significantly to the probabilistic weight of the submodel.

We now formally define our submodel construction method. We start with the notion of invariant of a set of traces.

Definition 6.3 (Invariant)**.** *Given a probabilistic process $M = \langle S, s_0, A, R\rangle$, and a set of finite execution traces T obtained from said model, an* invariant *of M through T is a state predicate ψ on the variables of M such that for every execution trace $t = s_0 \xrightarrow{p_0} s_1 \xrightarrow{p_1} s_2 \ldots s_n \in T$, it holds that $\forall 0 \leq i \leq n, s_i \models \psi$.*

An invariant then induces a unique submodel as follows:

Definition 6.4 (Invariant-driven submodels)**.** *Let $M = \langle S, s_0, A, R\rangle$ be a probabilistic model and ψ a state invariant; an* invariant-driven submodel *induced by ψ is a submodel $M' = \langle S' \cup \{\lambda\}, s_0, A', R'\rangle$ of M such that*

a) each state $s' \in S'$ is such that $s' \models \psi$;

b) for each $s'_1 \in S'$ such that $s'_1 \neq s_0$ it holds that $s_0 \xrightarrow{\alpha} s'_1$; and finally

c) for all states $s'_2 \in S \setminus S'$ such that there exist $s'_1 \in S', (s'_1, a, \mu_R) \in R$ with $\mu_R(s'_2) > 0$, it is the case that $M, s'_2 \not\models \psi$.

In other words, if a state s'_2 not in the submodel is directly reachable from a state s'_1 in the submodel, it must be the case that s'_2 violates ψ. The submodel is thus maximally connected from the initial state through the invariant ψ.

Our approach places a focus on maximising the automation of the estimation process. Therefore, we aim at automatically obtaining invariants. To this end, we produce probabilistically driven walks over the full system model, bounded in length, while we record the states (i.e. variable valuations) traversed. We use the tool Daikon [EPG+07], an invariant inference engine, to obtain predicates that hold over all traversed states. These invariant predicates, in turn, are used to synthesise an observer automaton that can drive the generation of a submodel via its parallel composition with the system model.

It is important to note that for working with Segala's Simple Probabilistic Automata it is necessary to resolve non-deterministic transitions during the probabilistically driven walk generation. In this thesis, we have chosen to replace non-deterministic transitions with an equiprobable distribution that chooses between the possible target distributions. The correctness of our approach is not hampered by this choice, as in fact any method of resolving non-determinism would serve our needs – any non-determinism resolution approach yields a valid submodel. However, it is left to be studied if this is the best way to resolve non-determinism. That is, whether a different determinisation scheme exists that produces a DTMC that, when analysed for determining reliability bounds, obtains better bounds or does so with less computational effort. We discuss on this decision and possible alternatives in Chapter 8.

The first step of our approach is then to perform simulation over an *equiprobably determinised* version of the original SPA.

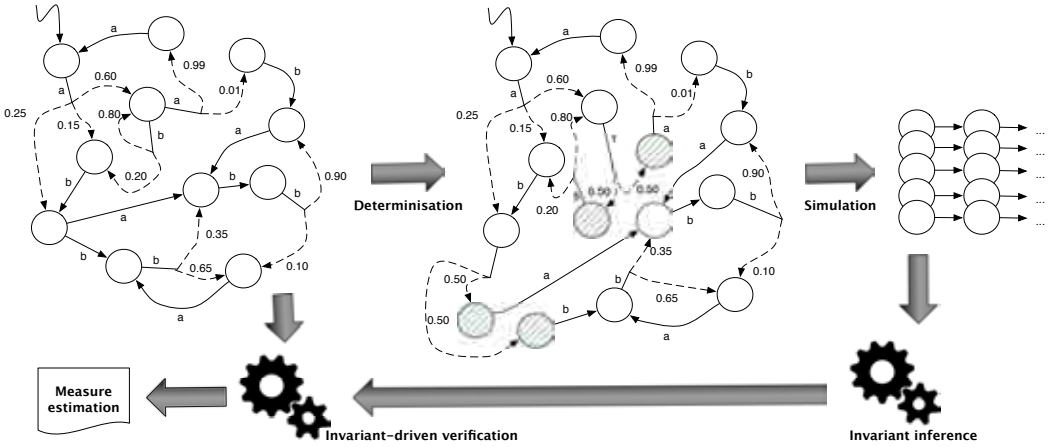


Figure 6.5: Workflow for partial exploration analysis

Definition 6.5 (Equiprobably Determinised Segala Simple Probabilistic Automaton)**.** *Let $M = \langle S, s_0, A, R\rangle$ be a Segala Simple Probabilistic Automaton. The* equiprobably determinised *Segala Simple Probabilistic Automaton of M is a DTMC $M_{det} = \langle S_{det}, s_0, A, R_{det}\rangle$ constructed in such a way that $S \subseteq S_{det}$, and for every $(s, a, \mu) \in R$:*

- *If (s, a, μ) is the only transition for s in M, add the transition to R_{det};*
- *otherwise, take all (s, a_i, μ_i). Add i states $t^s_1, \ldots, t^s_i$ to S_{det}. Add a transition (s, τ, μ) to R_{det} where $\mu(t^s_j) = 1/i$ for each of those added states, and 0 everywhere else. Finally, add transitions (t^s_i, a_i, μ_i) to R_{det} for each of the added states.*

Once the invariant is inferred through the simulations, it is used to generate the partial submodel of the *original* SPA. Figure 6.5 depicts the workflow of this approach.

In the following Chapter we put our approach to the test. We identify some exemplars from the literature that we believe are representative of several different system models and evaluate some of their properties. We state our research questions and present the results and conclusions we obtained by the application of our technique.

CHAPTER 7

Empirical Evaluation

In this section we set out to answer three questions in order to validate our approach.

> *Q1*: can our approach, when compared to model checking over full explorations, produce better bounds, in less time, for reward values and reachability probabilities of system models?

Probabilistic model checking approaches rely heavily on numerical solving of linear equations to calculate both reward values and probabilities. These numerical methods can suffer from convergence problem, which causes this calculation to grow steadily but very slowly. Since time budget is not unlimited, there must exist a stopping criteria for this convergence; either an absolute one such as stopping after a certain number of iterations or execution time, or else stopping whenever the increases in calculation is smaller than a given tolerance. In any case, results yielded by model checkers need to be considered as bounds because of this reason. This first research question aims at establishing whether the bounds obtained for our approach are more useful than those obtained by full-model checking efforts.

We will subdivide this research questions into questions *Q1a* for evaluation of reward bounds and *Q1b* for the case of probabilistic reachability bounds. Here we also answer related questions: first, whether submodels obtained through our approach perform better than similarly-sized submodels obtained through other approaches such as predetermined exploration criteria (e.g., BFS or DFS); and second, how good the obtained bounds are, especially in the cases where we can actually obtain the real reward value, and therefore we can contrast our estimated bounds to the actual value. Whenever we cannot obtain the actual reward value, we compare the bounds obtained through our approach to those obtained through the established model checking approach.

> *Q2*: can our approach, when compared to Monte Carlo approaches, produce better bounds, in less time, for the reward values and reachability probabilities of system models? Can Monte Carlo approaches benefit from our partial exploration techniques, that is, do Monte Carlo approaches perform better over partial explorations?

This question aims to compare our approach to Monte Carlo techniques, which are suited especially for the cases where the complete state space cannot be computed. We will study the assumptions needed to apply Monte Carlo techniques as well as our own, and will discuss these assumptions and their impact on the case studies that we analyse.

> *Q3*: how do the reward value and reachability probability estimations for submodels compare when these submodels are generated from automatically inferred invariants as in our approach against manually generated ones?

Q3 aims at assessing the added value of automatic techniques for obtaining submodels, against the cost of gaining a deep understanding of the model to be verified and developing a good submodel manually.

For each of the three research questions, the cases where the interesting states to be reached are rare events are of special interest, and we will discuss these at length.

7.1. Methodology

We analysed three different systems from the literature, and properties that can be expressed in terms of reward values or probabilistic bounds. These systems are especially amenable to be specified in either LTS, DTMC or SPA form, depending on their reliance on non-determinism, and whether the systems are probabilistic in nature. In the following sections we provide a description of each of these systems.

For each case study, we analysed the system models whenever they were available, or built them if they were not. Some of these models are probabilistic in nature, while others are non-deterministic. If appropriate operational environments were not available, we drew up environment model for them. The probabilities exhibited in our environment models are not meant to be reflective on real use, but rather as examples based on educated guesses. For some case studies, we built more than one environment for experimental reasons, such as varying probabilities or introducing non-determinism. In each case, we exhaustively checked that the resulting environment-system models conformed to Probabilistic Interface Automata restrictions. That is to say, in every case we modelled a valid PIA environment for each system model.

We modelled the properties of interest as state reachability formulae, and defined appropriate reachability reward structures for the properties needing such information.

When possible, we first computed the desired probabilities and rewards over the complete composite model either analytically when this was feasible, or using the PRISM model checker [HKNP06] if that was not the case. The model checker performs a numerical approximation to calculate probabilities and rewards. As this approximation may not converge, we made a note of convergence in each case and therefore treated convergent results as certain results, and non-convergent ones as bounds on the actual result.

Then we put our approach to the test for all case studies. We tested the approach for several automatically generated invariants varying the number and length of traces used for invariant inference. We used Daikon v4.6.4 [EPG+07] configured to produce invariants that are conjunctions of terms of the form $x \sim y$, where x and y are either variables in the model, or integer constants, and $\sim \in \{<, \leq, =, \geq >\}$. States in the models we analysed are described as different valuations of these variables.

The invariants we obtained were used to automatically build an *observer* automaton O, that monitors the validity of the invariant. This observer, when composed with the system model M, synchronises with all actions and forces transitioning into the λ trap state whenever the destination state of the intended transition would result in an invariant violation. Because of this manner of construction, the resulting subsystem is guaranteed to be a submodel of the original system model.

7.1.1. Experimental setting for Q1

For *Q1* we used a modified version of PRISM v4.0.3 to perform probabilistic model checking to estimate the reward values for both the full state space and for its invariant-driven submodels. Modifications allow for batch trace generation on a format understandable by Daikon (used for invariant inference) and time and memory-use tracking (used for generating intermediate reward results and for timing out when time budget is up). Intermediate reward and reachability probability results were generated for visualising convergence rates. PRISM was deployed on an 8x Core Intel Xeon CPU @1.60 GHz with 8 GB RAM.

PRISM provides different numerical methods for reward calculation. We performed a preliminary comparison of computation of the desired values over the full and partial explorations of smaller models for the Jacobi, Gauss-Seidel and Power methods. In every case the Backwards Gauss-Seidel numerical method outperformed, although not dramatically, the other methods. Because of this reason we opted to use this same numerical method for all our experimentation.

PRISM runs were considered complete when any of the following criteria held: first, we cut the iterative computation if the *absolute* difference between results of successive iterations of the numerical method was less than 10^{-2} in the case of rewards; and for the case of probabilities the difference was set to 10^{-7}. Relative differences are not an adequate stopping criteria because of slow convergence, which causes iterative methods to cut too early. This is especially true in the case of convergence of probabilities, where the magnitude of the expected values is extremely small compared to reward values, and thus requires a much smaller difference as stopping criteria.

Alternatively, we also interrupt the computation if the running time reached 24 hours; or if the available memory, which was limited to 1 GB for each run as they were deployed concurrently, was exhausted. Note that the time measured includes only the execution of the numerical methods. This allows for convergence analysis and favours full-model exploration as the time spent on construction of the model state space is not considered (we comment on execution time for submodel generation later in the Experimental Results subsection).

As we discussed above, in the case of reward estimation this choice of cutting iterations short (for whichever reason triggers the cut) results in that the obtained result is a lower bound on the actual value. Additionally, it cannot be known exactly how far this bound is from the actual value. Even though we will show that the obtained results are useful for arguing about the reliability of the systems under analysis, we performed additional checks. Taking advantage of Theorem 6.2, we calculate the probability of an arbitrary execution exceeding the bound obtained. We perform this probability calculation over the complete model (in the cases where this is possible), and over the obtained submodels. Recall that the probability obtained by performing the calculation on the submodels is an upper bound on the actual probability. This combination of lower bounds (on rewards) and upper bounds (on probability) further strengthens our reliability claims.

7.1.2. Experimental setting for Q2

For *Q2* Monte Carlo simulations were generated using the same version of PRISM and the same hardware as $Q1$. However, note that while our approach produces lower bounds to actual reward values with 100% confidence but for which precision (percentual difference between the estimation and the actual value) is unbounded, Monte Carlo produces estimations with varying degrees of confidence but for which precision can be bounded. Consequently, we aimed at performing Monte Carlo-based estimations for a range of confidence and precision values.

A critical precondition for applying Monte Carlo approaches is that all randomly generated traces must eventually reach the target states, and enough traces must be generated in order to guarantee estimations with a fixed precision and confidence. Setting a trace length horizon for the simulator to ensure all traces reach their target is typically done based on a rough estimation of the actual reward value, or an estimate of the underlying probability distribution [SVA05a]. This seemingly circular procedure can, however, work in practice. In our particular setting, we used the estimations obtained in $Q1$ as the basis for setting this horizon for each case study. The reason for choosing such an estimate are twofold: first, the actual rewards are guaranteed to be at least as much; and second, we will already have a measure of how much effort is needed to arrive at such an estimation. We will see that even under this setting, Monte Carlo approaches may require excessive effort to arrive to similar results in some of the case studies.

In those cases where Monte Carlo techniques turned out to be infeasible, we performed additional validation. In addition to comparing probabilistic model checking of submodels against Monte Carlo simulations of the complete model, we compared probabilistic model checking against Monte Carlo simulations over the same submodels. In other words, starting from the hypothesis that submodel generation does provide an added value, we wanted to further establish which approach was best for the second phase of the analysis; that is, whether probabilistic model checking or Monte Carlo evaluations should be employed over the obtained submodels.

7.1.3. Experimental setting for Q3

Finally, *Q3* uses the same setup and reward estimation approach based on inferred invariants as in $Q1$. The key difference is in the method for submodel generation. Manually produced invariants for submodel generation were put forth before any of the experiments were performed. Therefore, the manually proposed invariants were not tainted by knowledge gained from the automatic approach. The main heuristic for coming up with the invariants was analysing the model and identifying necessary (and more likely) conditions for reaching the target states.

The cost of manually generating an invariant is not simple to estimate. However, coming up with invariants that are useful for a partial exploration does demand from the user a deep understanding of the model under analysis. This is in general not trivial. In the context of this work, the cost of manually generating invariants, although non-trivial, was mitigated by the fact that the authors are familiar with the models under analysis. Eliminating this author bias would require further validation, possibly involving a well-designed user study. Such a study falls outside the scope of this thesis and remains future work.

Case study	System model	Environment determinism	Properties
Tandem queue	Non-deterministic LTS	Deterministic	Mean time to Failure (Reward) Bounded failure reachability (Probability)
Bounded Retransmission Protocol	DTMC	Deterministic	Mean time to Failure (Reward) Bounded failure reachability (Probability)
Bounded Retransmission Protocol	DTMC	Non-deterministic	Mean time to Failure (Reward) Bounded failure reachability (Probability)
IEEE 802.3 CSMA/CD	SPA	Non-deterministic	Mean turnaround time (Reward)
Network virus	SPA	Non-deterministic	Mean time to total infection (Reward) Bounded total infection reachability (Probability) Bounded node infection reachability (Probability)

Table 7.1: Summary of case studies analysed.

7.2. Case Studies

In the following paragraphs we will describe in detail each of the case studies employed. However, in an attempt to introduce all of the case studies and their analysed properties as early as possible, we quickly summarise this information in Table 7.1.

7.2.1. Tandem Queueing Network

The first case study is a tandem queueing network, based on [HMKS99]. Queueing systems have been extensively studied in queueing theory, and analytical solutions for some variants exist. However, due to the complexity of this particular model and its different queueing modes, general analytical queueing models are not easily applicable. Generating an ad-hoc analytical formulation would require extensive expertise and time, and it would not be easily adaptable to modifications in the design of the queueing system; even if these modifications are smaller ones.

The system consists of two process queues C and M of given (and in this particular case equal) capacities. Clients queue processes for execution in the first queue while it is not full. This first queue may either route a process to the second queue after a probabilistically chosen time elapses, or it might choose to deal with the request itself. The behaviour of this first queue is governed by two different *phases*. The difference between the phases is given by the probability with which it will choose to route its requests to the second queue or deal with them directly. The second queue has no other queue on which to unload its processes. Therefore, all it can do is service its requests, and it does so after a probabilistically chosen time elapses. A failure is observed when both queues are full, as at this time, clients cannot do anything but wait until some requests have been serviced and there is room in the

first queue for another process.

In our specific scenario devised for experimentation in this thesis, the capacity of the queues is fixed at 1200 each. The system environment is represented by the behaviour of the clients. Clients are less inclined (i.e., they take more time in average) to enqueue processes as the free capacity of the queues decreases. The clients were modelled accordingly using PIAs.

The reliability metric that we wish to estimate is the the *mean time to failure* (MTTF) of the system. Mean time to first failure is a widely accepted metric for reliability. This metric represents for how much time a client can expect to operate a system until it experiences its first failure. In this case, the failure is represented by the moment where a client cannot push any more tasks in the queues, and the first queue cannot offload any more work to the second. That is, a failure is met when both queues are full.

Consequently, the reward structure ρ we choose to model assigns the value 1 to every timing transition. It is generally accepted to employ *execution time* rather than *calendar time* for MTTF estimations [Lyu96]. While calendar time measures real time in terms of hours, weeks, etc., execution time is the time actually spent in system execution. This distinction is important for reactive systems which may have long idle times.

In our model, the state predicate that captures failure is $cliC = 1200 \wedge cliM = 1200$, and computing the mean time to failure amounts to calculating the expectation of the accumulated reward before reaching a state satisfying this predicate. Once we have a satisfactory value for this mean expected time to failure, we also aim at calculating the probability of experiencing a failure before this mean time.

7.2.2. Bounded Retransmission Protocol

The second case study [DJJL01] models a robust communication protocol that attempts to ensure coherent and complete delivery of data, the bounded retransmission protocol (BRP) [HSV94].

BRP is a variant of the alternating bit protocol, which allows for a bounded number of retransmissions of a given chunk (i.e., a part of a file). The protocol consists of a sender, a receiver, and two lossy channels, used for data and acknowledgements respectively. The sender transmits a file composed of a number of chunks, by way of *frames*. Each frame contains the chunk itself and three bits. The first bit indicates whether the chunk is the first one; the second one if it is the last chunk; and the third bit is the alternating one, used for avoiding data duplication.

The sender waits for acknowledgement of each frame sent. The sender may timeout if either the frame or the corresponding acknowledgement are dropped which could be caused, for example, by either the frame or the corresponding acknowledgement being dropped. When this happens, the sender resends the frame and does so repeatedly up to a specified retry limit. If the limit is reached and the transmission is terminated, the sender may be able to establish that the file was not sent (if some chunks were left unsent) or it may not know the outcome (if the last frame was sent but no acknowledgement was received). In any case, the sender may send a new file, resetting the retry count. A maximum of 256 retransmissions are attempted per file before the sender gives up and aborts transmission of the file, regardless of the size of the file being sent, Once a file is sent successfully or its transmission fails, the system waits for another file to be sent.

Protocol clients send files one at a time. Each of these files is of a different size (in number of chunks). This size may be different for each file, varying between just a few and 1500 chunks. We developed two probabilistic models for this problem,

and analysed them separately. First, we assumed complete knowledge about the distribution of the sizes of the file being sent. Therefore, the choice of file size was modelled probabilistically, yielding a deterministic PIA as environment model, where exceedingly large or small files are modelled to be less likely to be sent than those of average size. In the second model we developed, we introduced uncertainty regarding this knowledge, and kept the size choice non-deterministic, representing this absence of information. Under this modelling choice, the second case yielded a non-deterministic PIA.

In this case, we also wish to estimate the mean time to the first failure, where failure is defined as the sender failing to send a complete file (*incomplete*) or not being able to establish if a file was sent successfully (*unknown*). Consequently, the state predicate describing failures is $incomplete \vee unknown$. The definition of time for this case study aims at establishing how many data packets can be expected to be sent successfully before failure. For the DTMC model we obtained the mean number of packets being sent before experiencing failure, while for the SPA model we obtained both the minimum and maximum mean number of packets, which represent the worst case and best case scenarios respectively.

Again, once we calculated the mean expected time to failure, we also calculated the probability of experiencing this failure before the obtained mean time.

7.2.3. IEEE 802.11 Wireless LAN

The third case study depicts the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism of the IEEE 802.11 protocol [Ins97]. The protocol uses a randomised exponential backoff rule to minimise the likelihood of transmission collision. That is, whenever a collision was averted by a component sensing the busy carrier when trying to send data over busy media, the component is backed off (it needs to wait until trying to resend) for a time. This time is chosen randomly from a specified range of possible delays, and successive failures cause this range to increase exponentially. The goal of the protocol is to divide, as equally as possible, the access to the channel between all participants that may collide.

The model used depicts a two-way handshake mechanism of the IEEE 802.11 medium access control scheme, operating in a fixed network topology. The probabilistic model itself was extracted verbatim from [HMZ+12]. This model exhibits both stochastic behaviour (for example, in the randomised backoff procedure, that allows up to seven exponential backoff levels) and non-deterministic behaviour (for example, in modelling the interleaving of actions between the two independent emitter stations). Therefore, the model is an SPA.

In this case, the protocol is probabilistically guaranteed to never fail, that is, both stations will eventually be able to send their packets. However, it is interesting to know for how long they will have to wait, in average, to achieve this objective. *Turnaround time* is a measure for both reliability of systems, as it may include time necessary for error correction or recovery, as well as a measure for performance. In general, the turnaround time for a process refers to the time that elapses between it starting its task until it finishes or provides some result. The starting and finishing times may be arbitrarily defined (for example, start time may be either the moment the process takes control of execution, or rather the moment it is sent a request). In general, we may refer to turnaround as the time it takes a process to produce the required results after it is started.

In this case, we are interested in estimating the turnaround time for two stations to be able to successfully send their packets and advance to their *done* state, while avoiding potential collisions. As such, the state predicate that describes this final

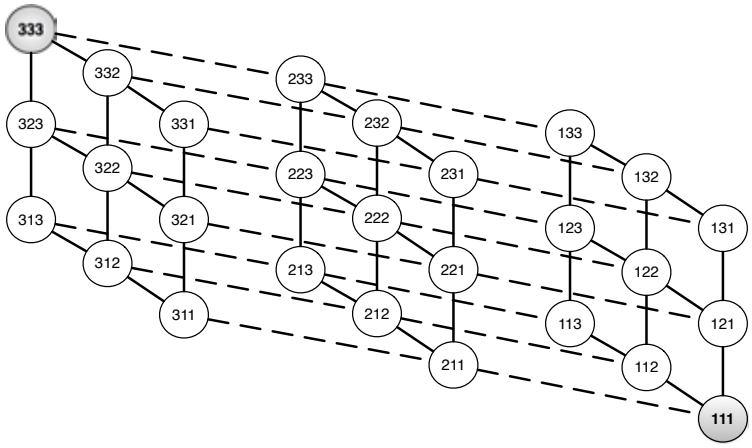


Figure 7.1: A $3 \times 3 \times 3$ network cube. On the lower right the infected node 111, the target node is 333 in the upper left.

state is $station1 = done \wedge station2 = done$. Note that, unlike the previous case study, both stations managing to send their messages is not a rare event at all if the protocol works correctly. However, the sheer size of the model does hamper direct estimation.

7.2.4. Network virus infection

In this case, we analyse the behaviour of a virus infection on a computer network. This case study is based on [KNPV09, DNKLM06] but is heavily expanded as we will detail further on.

The network is a cubic grid of nodes, as opposed to the original case study in [KNPV09] which was based on a plane grid; a cubic grid allows more virus paths as well as customising the model to sizes that quickly grow to be intractable. The size of the network is given by N, the number of nodes in any given edge of the cube. Each node is connected to the nodes at its left, right, up and down, as well as to those behind and in front of it. Nodes in the outer faces may have less connections. Figure 7.1 depicts a $3 \times 3 \times 3$ cubic grid.

We model the behaviour of a virus infection on a firewalled, self-healing network. In this setting, once a node is infected, it tries to propagate to its neighbouring nodes. In order to succeed, it needs to first defeat the node's firewall, and then attempt infection once the firewall is down. The network is self-healing, as healthy nodes will try to repair its infected neighbours.

The scheduling between these actions is completely non-deterministic. On the other hand, we built a Probabilistic Interface Automaton of the environment that describes the probabilities of success when trying to break a firewall, infect a vulnerable node, or repair an infected node.

In each case we start with a healthy network, save for one of the corner nodes, which starts infected. The properties of interest we analyse in this case are the following.

- the minimum expected time to total infection of the network;
- the minimum expected time to infection of the node at the opposite corner of the initially infected one;

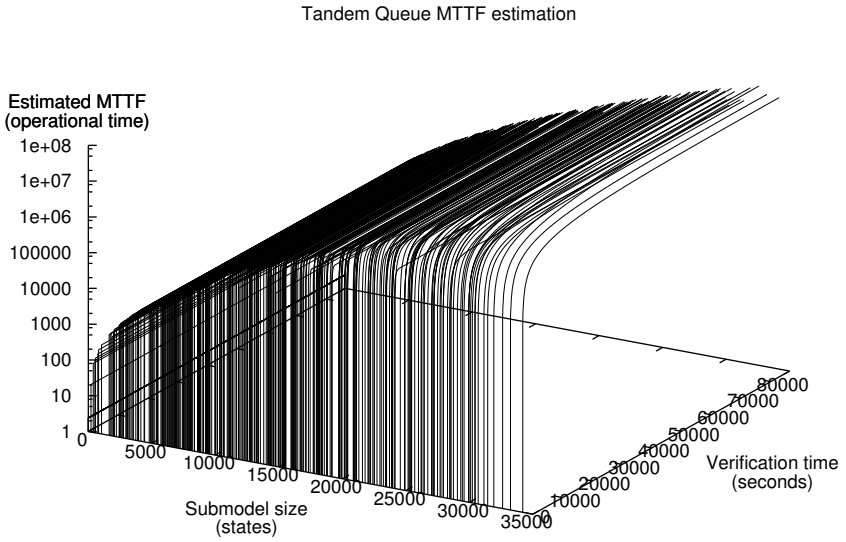


Figure 7.2: Results of analysis of Tandem Queue for different sized submodels, Backwards Gauss-Seidel method.

- the probability that the network is completely infected after a given number of operations; and
- the probability that the farthest node from the initial infection is infected after a given number of operations.

7.3. Experimental Results

We now present the experimental results obtained for the three research questions presented above.

7.3.1. Question 1

When comparing probabilistic model checking of both full and partial models we are interested in considering the relationship between the inferred invariant, the size of the resulting submodel, and the value of the reward estimation obtained from it. We are also interested in gaining insight on combinations of trace length and number of traces that are likely to yield the best overall result.

Tandem Queue analyses

For the Tandem Queue case study the estimated mean time to failure, calculated using probabilistic model checking, in 24 hours over the full model was 4.20×10^5. This full model comprises $\sim 1.50 \times 10^7$ states. Regarding computations over submodels, we report on MTTF estimation (Figure 7.2), submodel sizes (Figure 7.3) and a representative selection of invariants obtained (Table 7.2) for various settings of sample size and individual trace length. The complete set of obtained invariants can be seen in Tables A.1 and A.2 in Appendix A.

Note that our best MTTF estimation is about 7×10^7, a full two orders of magnitude larger than what could be estimated through full model checking. Even if

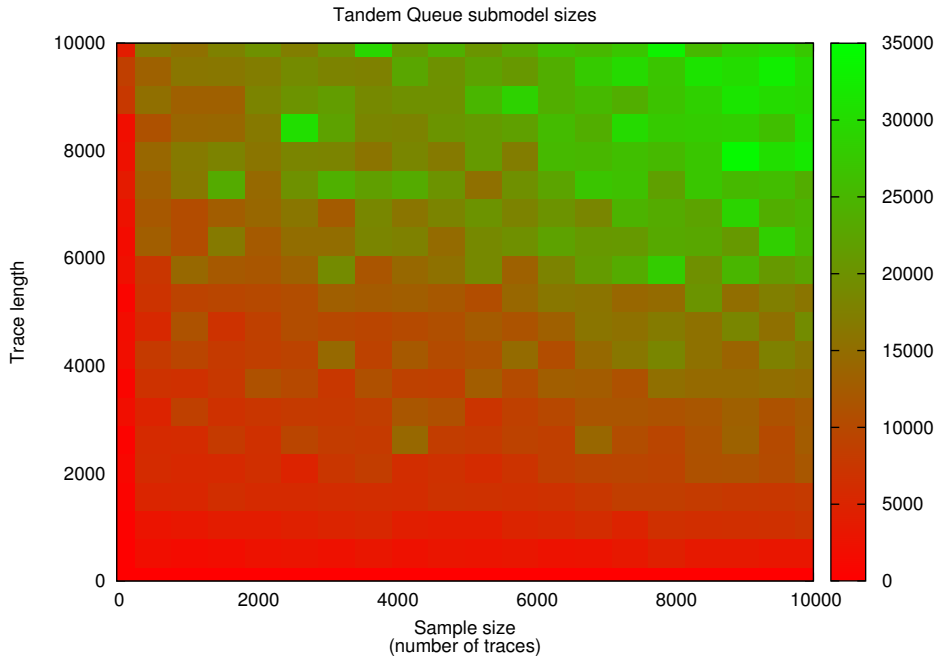


Figure 7.3: Tandem Queue submodels sizes for different sample size and trace length parameters.

Traces	Length	States	Invariant
5000	1000	14134	$cliC \leq 69 \wedge cliM \leq 18 \wedge state \leq 9$
10000	1000	16086	$cliC \leq 83 \wedge cliM \leq 17 \wedge state \leq 9$
5000	2000	23388	$cliC \leq 100 \wedge cliM \leq 21 \wedge state \leq 9$
10000	2000	22486	$cliC \leq 92 \wedge cliM \leq 22 \wedge state \leq 9$
5000	3000	20932	$cliC \leq 98 \wedge cliM \leq 19 \wedge state \leq 9$
10000	3000	25228	$cliC \leq 108 \wedge cliM \leq 21 \wedge state \leq 9$
5000	4000	24538	$cliC \leq 105 \wedge cliM \leq 21 \wedge state \leq 9$
10000	4000	24882	$cliC \leq 94 \wedge cliM \leq 24 \wedge state \leq 9$
5000	5000	26424	$cliC \leq 104 \wedge cliM \leq 23 \wedge state \leq 9$
10000	5000	23686	$cliC \leq 97 \wedge cliM \leq 22 \wedge state \leq 9$
5000	6000	26182	$cliC \leq 99 \wedge cliM \leq 24 \wedge state \leq 9$
10000	6000	31902	$cliC \leq 121 \wedge cliM \leq 24 \wedge state \leq 9$
5000	7000	29926	$cliC \leq 123 \wedge cliM \leq 22 \wedge state \leq 9$
10000	7000	30674	$cliC \leq 121 \wedge cliM \leq 23 \wedge state \leq 9$
5000	8000	23910	$cliC \leq 107 \wedge cliM \leq 20 \wedge state \leq 9$
10000	8000	29424	$cliC \leq 116 \wedge cliM \leq 23 \wedge state \leq 9$
5000	9000	29924	$cliC \leq 118 \wedge cliM \leq 23 \wedge state \leq 9$
10000	9000	29926	$cliC \leq 123 \wedge cliM \leq 22 \wedge state \leq 9$
5000	10000	27174	$cliC \leq 107 \wedge cliM \leq 23 \wedge state \leq 9$
10000	10000	27460	$cliC \leq 100 \wedge cliM \leq 25 \wedge state \leq 9$

Table 7.2: Tandem Queue model - Selection of submodel sizes and invariants for different parameter configurations.

this is not the actual MTTF, this jump in estimation quality could make a difference in establishing a case for reliability assurance of the system.

The first figure shows, for different automatically generated sized submodels, the estimated MTTF (shown over a logarithmic scale for convenience) along with how much time it took for the calculation to finish. Executions that finished before the 24 hour timeout are flattened on the MTTF axis at the time the result was reached. It is noteworthy that none of the automatically obtained submodels is larger than 35000 states, comprising roughly 0.25% of the states of the complete model. Despite having explored only such a small percentage of the full model, the obtained lower bound for MTTF is quite large in some cases, possibly sufficient to argue for high system

reliability – MTTF is *at least* in the order of 10^7. Although very small submodels do not provide good bounds, larger submodel MTTF estimations increase dramatically, quickly rising to the 7×10^7 maximum MTTF witnessed, which is a full two orders of magnitude beyond the estimation for the full model.

An important question is whether good submodels can be obtained in a consistent fashion by parameterising trace quantity and length parameters of the simulation phase. Figure 7.3 shows that such submodels can be obtained automatically in a consistent way for this example. Focusing on the upper-right corner of the figure, it can be seen that choosing values for trace length and sample size in that region consistently results in appropriate submodels.

It can be observed that experiments with trace length below 3000 do not consistently produce rich enough models that yield good MTTF estimates. Unsurprisingly, small sample sets are also inconsistent in their results. However, once the sample set size parameter is set to at least 6000 samples, the submodels produced consistently yield large MTTF estimates. In summary, for this case study a minimum of 6000 samples of traces at least 4000 steps long are necessary for consistent results. Furthermore, increasing these parameters does not yield clear advantage in terms of the final MTTF estimation. Both figures also show that results become more stable as these parameters are increased.

State space size alone is not the only important factor when evaluating the effectiveness of the approach. For a given size expressed in number of states, many submodels of that size exist, and not all of them may be effective. In [PBU10] we have already shown that submodels obtained through depth first search (DFS) explorations yield very poor results, as they allow short traces to escape the submodel to the λ state. Although breadth first search (BFS) obtains higher MTTF lower bounds than DFS when used as a submodel generator, it performs poorly against our approach, as the state space that it explores is not as relevant. For example, our approach using 10000 traces 10000 states long (one of the best performers) obtains a 27460 state sized submodel, which is characterised by the invariant $cliC \leq 100 \wedge cliM \leq 25 \wedge state \leq 9$. Consider a similarly sized BFS generated submodel of 28000 states. The Tandem Queue model allows four different actions (push, fwd, svc_1,svc_2). Conservatively assuming at most two actions enabled at each state, an equal sized BFS submodel would explore at most $\lceil \log_2(27460) \rceil = 15$ levels deep. Such a submodel would only allow for very limited behaviour. If each transition level generated a new state, queues of no more than 15 elements could be generated by such a submodel. Of course, it is not always the case that a new state is generated. In fact, a BFS exploration that allows for 50 elements per queue results in a 32000 state submodel. The MTTF obtained through such a submodel is ~ 70000, very far from the results we obtain.

Regarding potential overhead of trace generation and invariant inference, memory consumption is negligible with respect to representing the state space of the full model, as only one relatively short trace needs to be kept in memory at a time. Timewise, analysis of 10000 traces of length 10000 took less than an hour. Accounting for this hour in the verification time budget, the submodel that yielded the highest MTTF lower bound would have achieved a result of $\sim 6 \times 10^7$ in 23 hours, still a large increase against the estimation obtained via full model verification.

Although not intended to be shown to developers, we report on some of the automatically inferred invariants in Table 7.2. The discovered invariants deal with bounding the size of both queues, while the variable *state* encodes whether the queues are full or not, and the phase the system is in at the time. It is noteworthy that although it is intuitive that an invariant should bound the queue sizes, it is unlikely

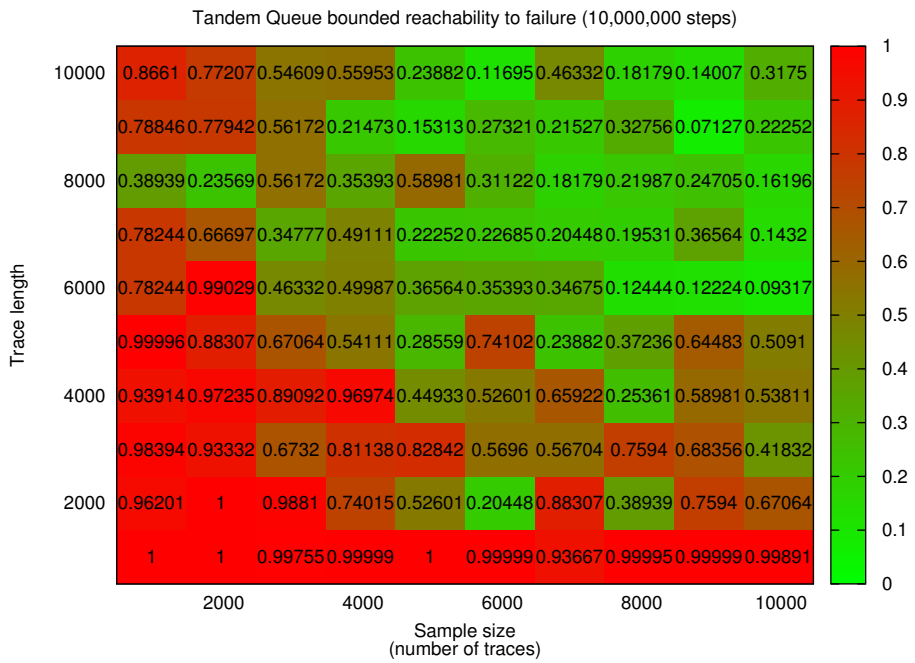


Figure 7.4: Tandem Queue failure bounded reachability probabilities for state spaces obtained from different sample size and trace length parameters.

that a human would come up with the particular bounding values used.

Bounded reachability properties A potential criticism to the previous analyses is that, just as it happens when performing model checking of complete models, the reward estimation over the partial submodels does not necessarily converge. Therefore, even though we know for certain that the mean number of operations before a failure is larger than $\mathcal{O}(10^7)$, we still don't know how far this may be from the actual mean.

In order to provide a more convincing answer to this question, we set out to validate whether an arbitrary execution is likely (or not) to exceed this obtained result. To this end, we performed a second verification over the obtained partial submodels. In this case, the property of interest is quantifying the probability of an arbitrary execution reaching the failure state *before* 10^7 operations have taken place.

We first attempted to verify this property over the complete model. Unfortunately, the probability calculation did not converge after 24 hours of execution, and at this time it had calculated a probability of 0. This is clearly wrong, as the failure state is reachable; and so is the trap λ state.

After this (failed) initial attempt at a complete verification, we proceeded to verify the same property over each of our previously constructed partial state spaces. We adjusted the convergence criteria to an absolute difference of 10^6 to account for slow convergence. The results obtained are depicted in Figure 7.4. Recall that, because of Theorem 6.2, the probabilities we get from these partial state spaces are *upper bounds* to the actual reachability probability. Therefore, in this case smaller probabilities are better. As a consequence, Figure 7.4 depicts smaller probabilities with greener colours. The number in each square is the actual probability obtained.

It must be noted that, in every case, the probability calculation converged well before the 24-hour timeout. If this convergence was not attained, the results would be difficult to interpret. This is not an issue for reward estimation, where the results yielded are lower bounds. However, in the case of probabilities, the results are *upper*

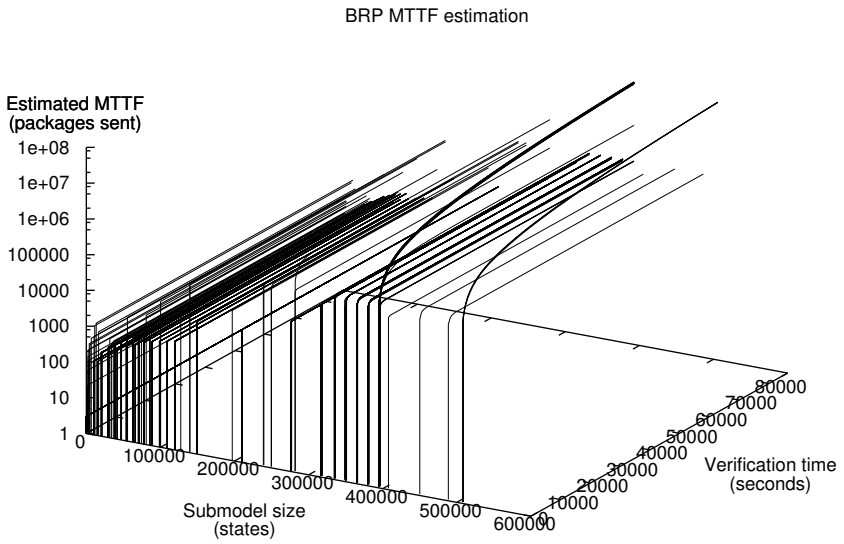


Figure 7.5: Results of analysis of BRP (probabilistic file size choice) for different sized submodels, Backwards Gauss-Seidel method.

bounds. If convergence is not attained, there is no telling whether the probability bound wouldn't keep rising.

We can see that our experiments concluded that the reachability probability is guaranteed to be at most 0.07127, a result we obtained from the partial state space constructed from 9000 traces that were 9000 steps long. The combination of both the lower bound on the mean time to failure with the maximum bound on the probability to exceed a large number of steps allows us to argue strongly for the reliability of this system.

Bounded Retransmission Protocol - probabilistic/deterministic environment

For the BRP case study in its fully probabilistic variation, similar results were obtained and are shown in Figures 7.5 and 7.6. Table 7.3 shows some selected invariants, while the complete list can be found in Tables A.3 through A.6 in Appendix A.

In contrast to the prior case study, we were unable to obtain the MTTF for the full model due to state explosion that exhausted available memory. However, observations prior to running out of memory showed that the full model contains at least 30 million states. Referring to the results figures and tables, this means that the submodels we analysed represent up to 2% of the size of the full model, still a very low percentage. Furthermore, the highest MTTF bounds were obtained for submodels with a size starting from 400000 states (less than 1.33% of the full model), which turned out to yield an MTTF in the order of 2.5×10^7. This result is most significant, because of the impossibility of estimating MTTF for the full model.

Note that for submodels whose size is around the 400000 and 500000 states mark, there are both estimations that provide very good bounds and those that yield not so useful ones. Interestingly enough, those that do not perform well arise from submodels obtained through invariants inferred from sample sets where generated traces were shorter than 7000 states long, while sets of longer traces perform very

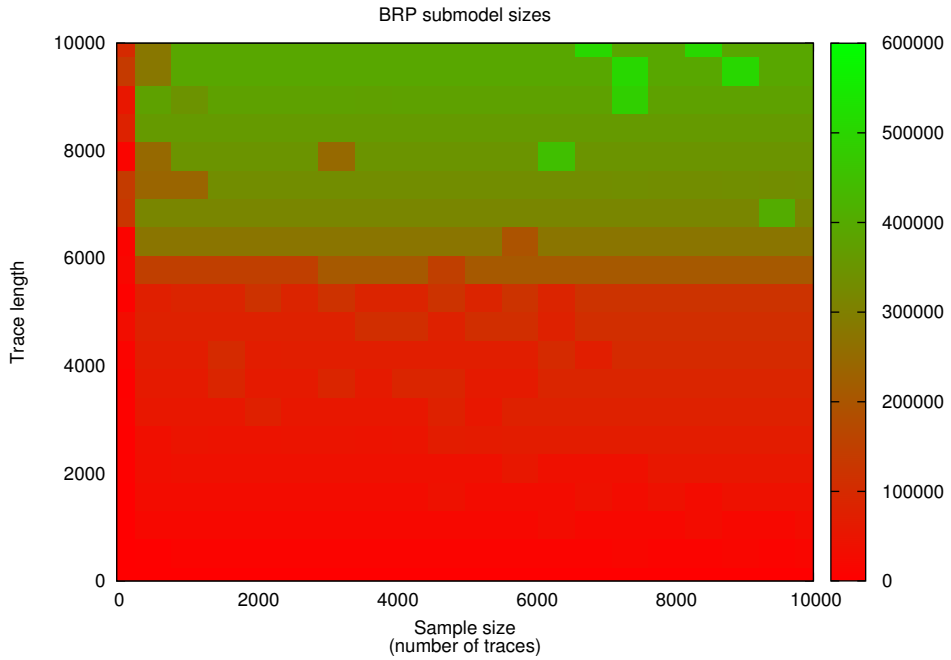


Figure 7.6: BRP submodels (probabilistic file size choice) sizes for different sample size and trace length parameters.

well. This shows that appropriate trace length, as well as sample size, is critical to the final MTTF estimation.

Performing a similar analysis to that performed for the Tandem Queue model, we discover that similarly sized submodels obtained through BFS exploration do not provide such higher MTTF lower bounds. One of our best performers, at 10000 traces 10000 states long, produces a submodel 392786 states in size which (with eight BRP actions and conservatively assuming three enabled at any time) results in a BFS submodel of depth $\lceil \log_3(392786) \rceil = 12$, which models very few frames being sent. In fact, a BFS-like submodel that allows only for 5 frames to be sent per file comprises ~ 400000 states and yields an MTTF of only 40.

Figure 7.6 depicts information related to the possibility of obtaining useful submodels. It can be seen that it is quite easy to obtain such submodels, without many restrictions on experiment configuration. In fact, the configurations for this case study behave much more steadily than with that of the Tandem Queue. Sets of 4000 traces of at least 7000 states seem to be enough for obtaining good estimates. Further increases of these parameters yield larger and slightly better-performing models, and this increase is much smoother (hence predictable) than is the case for the Tandem Queue submodels.

As in the previous case study, trace generation and invariant inference incurs an overhead. In this case, since the model is more complex, this analysis can take up to 2 additional hours. Reducing the verification time by these 2 hours, the estimated MTTF would have been still large, about 2×10^7. Recall that this overhead was not included in measured time to allow graphs to show convergence speed of numerical analysis.

Regarding the invariants in Table 7.3, it turns out they can be quite cryptic. The variables $fileSize$, i and $nrtr$ describe the size of the file being sent, how many frames have been sent for that file, and the number of retries attempted, respectively. Other variables such as s_{ab}, r_{ab}, bs and fs encode the bit alternation in the protocol. The invariants obtained establish relationships between variables that at first glance

Traces	Length	States	Invariant
5000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	4000	315191	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	5000	333099	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	5000	333099	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	6000	348129	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	8000	486334	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 3 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	9000	505704	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 3 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$

Table 7.3: BRP (fully probabilistic) model - Selection of submodel sizes and invariants for different parameter configurations.

seem unrelated, making them quite unintuitive even for a domain expert.

Bounded reachability properties Just as we analysed the probability of reaching an error within a reasonable bound for the Tandem Queue system, we performed the same analysis for the case of the BRP system. The outcome of this experimentation is depicted in Figure 7.7. The results are similar to those obtained for the Tandem Queue case: for the BRP system model, the best performing submodels when it came to MTTF estimation are also the best estimators for the reachability probability. Additionally, the obtained probability bounds are low enough for us to guarantee reliability.

However, as opposed to the case of the Tandem Queue analyses, these results must be taken with a grain of salt. In these cases, the results were obtained after the 24 hour timeout, and we have no certainty to what point these bounds might have increased.

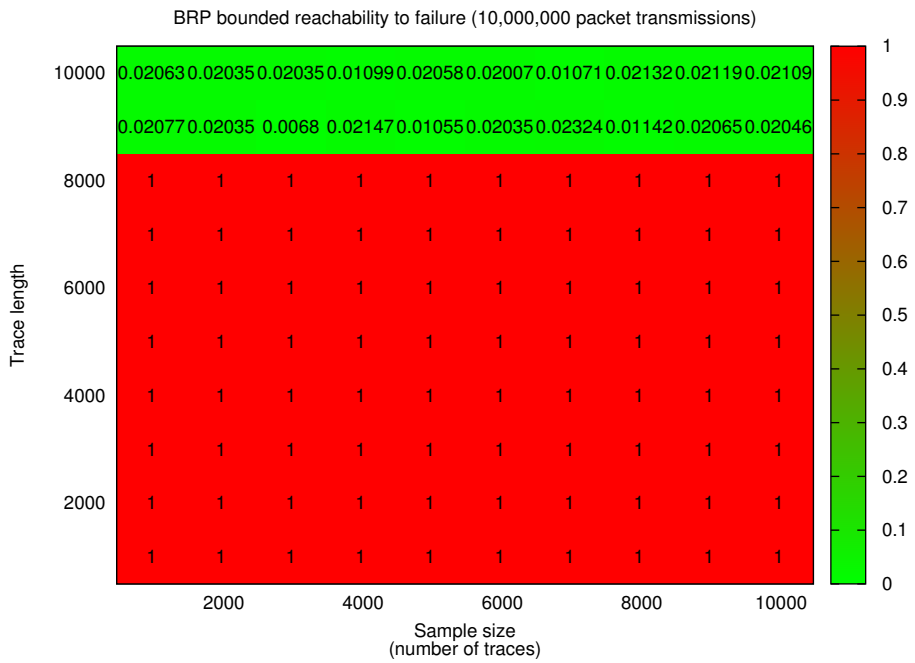


Figure 7.7: BRP failure bounded reachability probabilities for state spaces obtained from different sample size and trace length parameters.

Bounded Retransmission Protocol - non-deterministic environment

As we explained before, we also developed a version of the BRP model that leaves the file size choice to a non-deterministic process. Recall that introducing non-determinism into a model requires a scheduler function to solve this non-determinism, and that we focus on those that yield the minimum and maximum probabilities or reward values. Therefore we turned our attention to finding out the minimum and maximum possible mean times to failure. We performed the same verifications we did for the deterministic model, but effectively twice, as we require both extreme values. However, the invariant inference phase is performed over only one set of simulation traces, regardless of whether we will ultimately estimate minimum or maximum values. The same submodel will be used for both extreme estimations. Figure 7.8 shows the sizes of the submodels obtained. Note that they are slightly smaller than in the fully probabilistic case. Also, larger submodels are obtained more consistently in this non-deterministic case.

As was the case for the fully probabilistic case, we were unable to obtain an estimation for the MTTF for the full model via probabilistic model checking, because of memory being exhausted due to state explosion. After the 24 hours of allotted time elapsed for each extreme value estimation, the results yielded a model comprising nearly 29 million states, while the reward estimation set a minimum MTTF value of 60297 and, surprisingly, a maximum MTTF of 50819. This discrepancy of the maximum estimation being actually less than the minimum one can be explained as an unintended consequence of the numerical verification procedure. The verification algorithm for extreme probabilities involves solving an optimisation problem for each extreme value. In the case of the minimum time to failure, the optimisation resolution converges much faster. Indeed, the minimisation procedure actually performed about 20% more iterations than its maximisation counterpart, a factor that can explain this discrepancy.

After failing to obtain an exact value for the MTTF extreme values, we turned

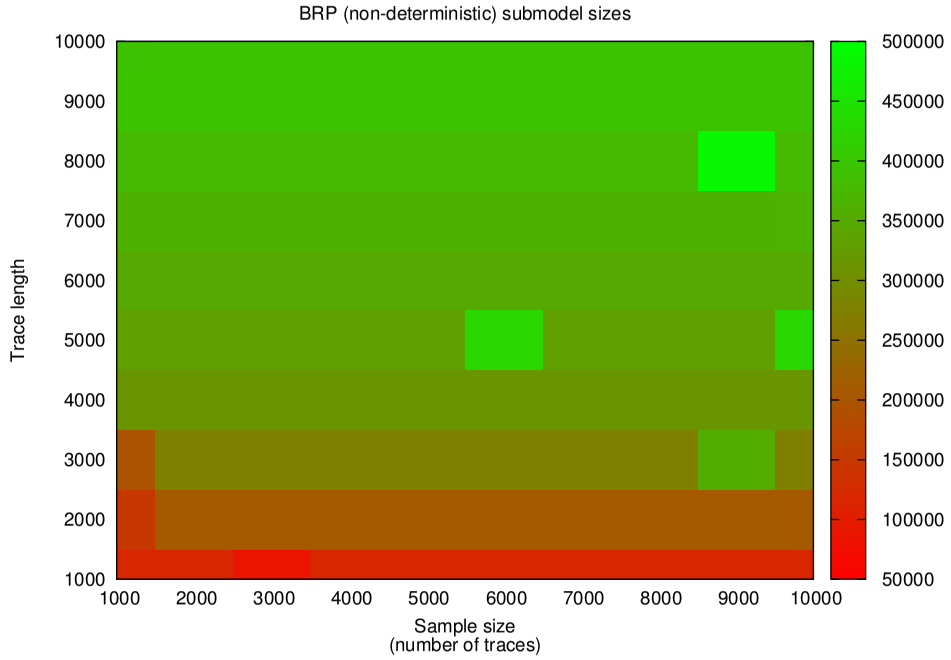
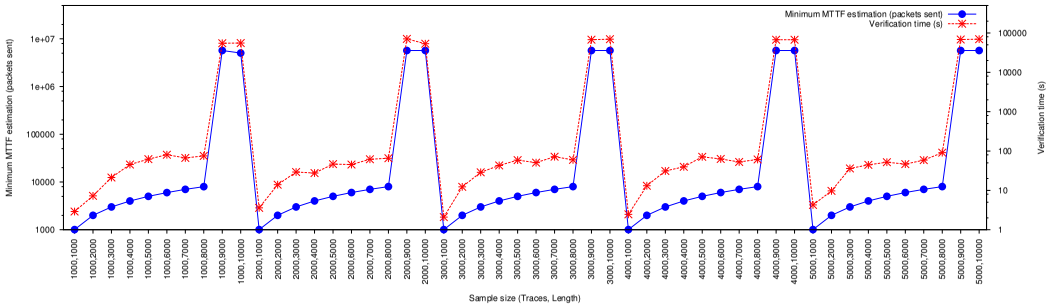
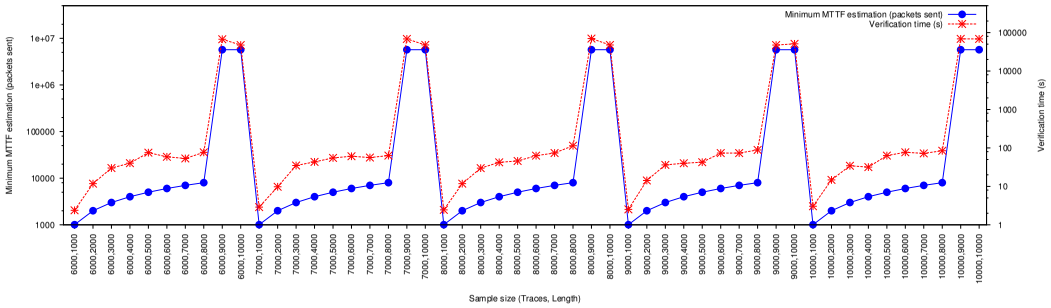


Figure 7.8: BRP submodels (non-deterministic file size choice) sizes for different sample size and trace length parameters.

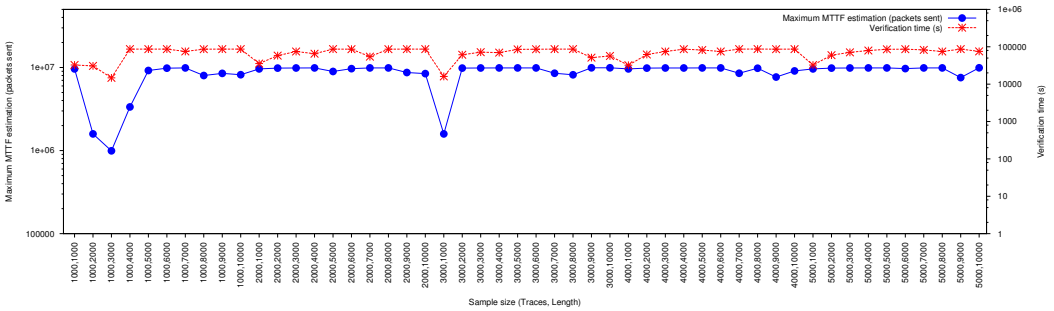


(a) (1000,1000) - (5000,10000)

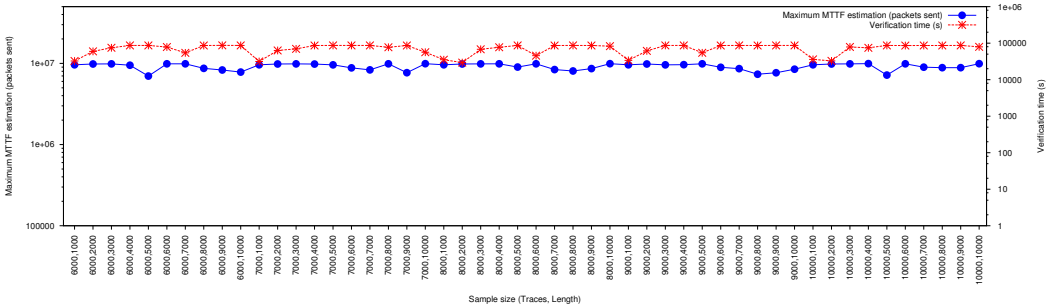


(b) (6000,1000) - (10000,10000)

Figure 7.9: Verification times and submodel sizes for minimum MTTF estimation



(a) (1000,1000) - (5000,10000)



(b) (6000,1000) - (10000,10000)

Figure 7.10: Verification times and submodel sizes for maximum MTTF estimation

our attention to the estimation over partial explorations. We report on these experiments in Figures 7.9 through 7.10. The first two summarise the results obtained for minimum MTTF estimation, while the other two do the same for maximum MTTF estimation.

It is interesting to note several things about these results. First, the submodels analysed represent, similarly to the fully probabilistic case, about 2% of the size of the full model, a very low percentage. It also quickly becomes evident that there is a strange phenomenon taking place with the estimation of the minimum rewards. Almost all results are polarised either towards the 5.6×10^6 value; or towards the much less impressive $[1000, 8000]$ range. Further, the length of traces simulated is critical, particularly in the case of estimating the minimum MTTF. Note that simulating traces less than 9000 actions long, results in the smaller estimations for minimum MTTF. This seems to have its correlation with the invariants that were inferred in each case, for which we have a selection in Table 7.4 (the complete list of invariants can be found in Tables A.7 through A.10 in Appendix A).

The invariants explain the results obtained. Note that, in the invariants obtained with traces less than 9000 steps long, the variable i is restricted to no more than 1333. Recall that i indicates the number of packets of the file that have already been sent. These invariants show that, for the traces analysed, sometimes the maximum file size (1500) was chosen, but never completely sent. For our approach, such situations would lie in the *unknown* set of the state space, and thus conservatively evaluated as failing states. However, invariants obtained for longer traces do allow i to reach its maximum of 1500, which explains the dramatic increase of the estimations. Even more, increasing the simulation length to 10000 actions does pay off in some cases, although the increase is not nearly as dramatic.

In the case of the maximum MTTF estimation, all submodels behave more or less uniformly, except for a couple of runs that estimate a lower MTTF. Note however that these submodels are obtained as the result of the analysis of few, and short,

Traces	Length	States	Invariant
5000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	4000	315191	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	5000	428334	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 3 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	7000	363159	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$

Table 7.4: BRP (non-deterministic) model - Selection of submodel sizes and invariants for different parameter configurations.

simulation traces. As a result, it is not surprising that these simulations failed to capture a significant portion of the system behaviour.

When compared with the result obtained for full model estimation, it can clearly be seen that estimation over submodels pays off – the maximum MTTF estimated for submodels is, in all cases, at least 50 times larger than those obtained for the full model.

There is a final point that needs to be noted. As we discussed earlier, the submodels obtained by analysing shorter simulations are not very good for minimum MTTF estimation. However, they are the best performers for estimating *maximum* MTTF. This is a consequence of the state space being smaller, as this allows for more numerical iterations in the same time budget. Another important factor is that choosing a smaller file size allows for a larger Mean Time to Failure. This is because when transmitting a smaller file, the chance that the protocol will deplete its allowed retries is smaller than with a bigger file, simply because it has less chances to fail. This contrasts with the minimum MTTF calculation, which becomes larger just as bigger files are allowed in the model.

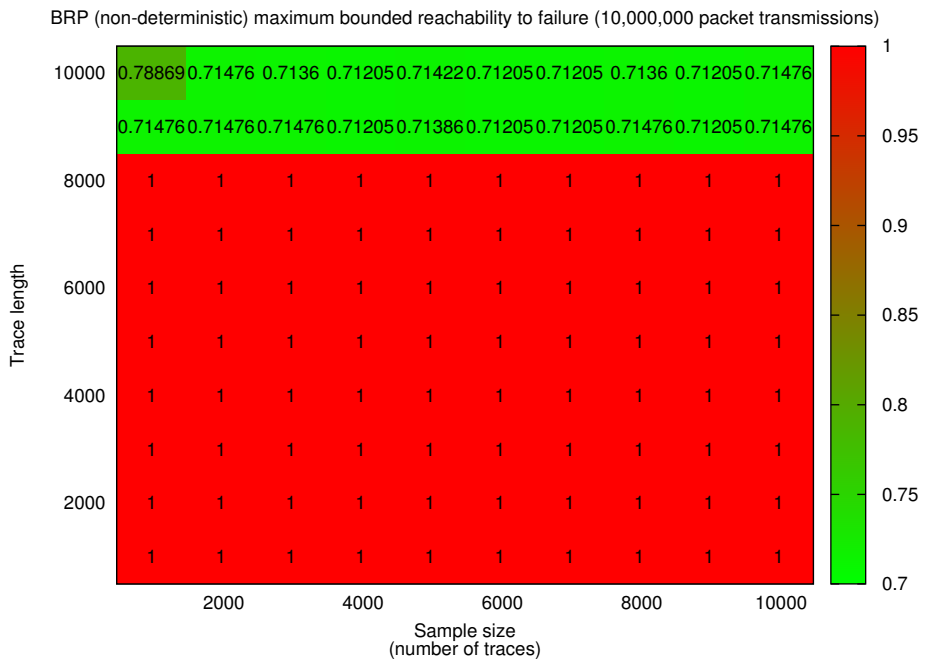


Figure 7.11: BRP (non-deterministic) failure maximum bounded reachability probabilities for state spaces obtained from different sample size and trace length parameters.

Bounded reachability properties Similar to the case of the fully probabilistic environment model, we also analysed the bounded error reachability probabilities for the non-deterministic environment model. The outcome of this experimentation is depicted in Figures 7.11 and 7.12.

The maximum reachability probability results follow a pattern similar to the fully probabilistic case. The larger probabilities, though, suggest that the actual mean time to failure is closer to 1×10^7 than in the fully-probabilistic case. These experimental runs converged in every case, so there is little question to their correctness.

The minimum reachability properties calculated are roughly the same for all submodels, differing in at most ~ 0.05 in most cases. It also happened that all calculations converged before the 24 hour timeout, reinforcing their validity.

There are some exceptions where the probability estimations take values closer to one. These cases coincide, unsurprisingly, with those for which the estimation of maximum mean time to failure performs poorly. Again, this is likely a result of the simulation traces not being descriptive enough to produce a significant submodel.

WLAN collision avoidance protocol

We now turn our attention to the analysis of the WLAN collision protocol model. In this case study, we are interested in estimating the turnaround time (TAT) for both emitting stations to complete sending their intended data. That is, we wish to know the mean time from the moment the first station intends to send data until both of them have successfully sent their data, including all necessary backoff time.

For this case study we also attempted to produce an estimate for the full model. Contrasting with the previous case studies, the event under analysis is not a rare event at all. On the contrary, it is desirable that in every instance both stations are able to send their data in a reasonable time. During this analysis, we obtained a

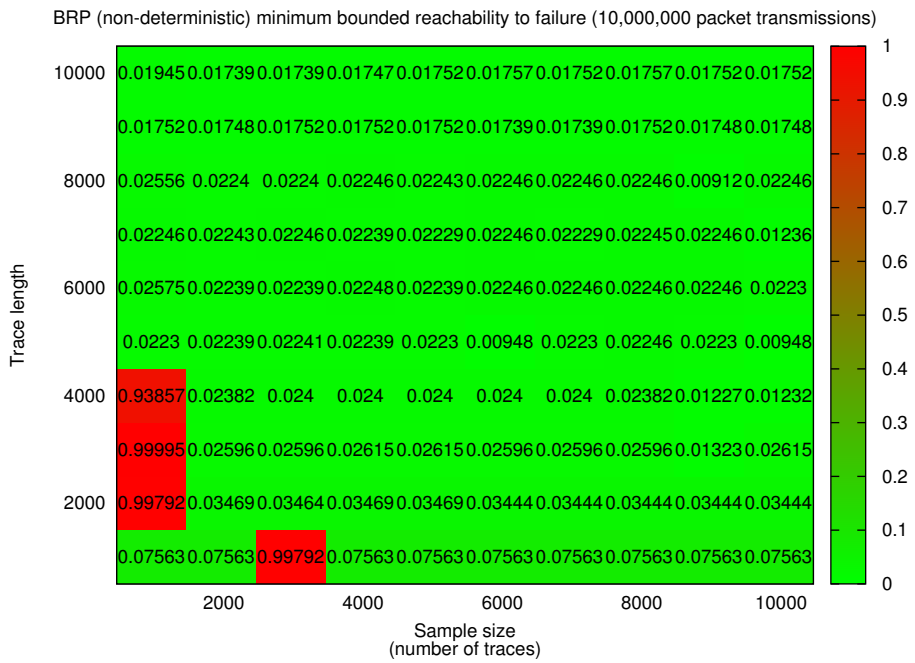


Figure 7.12: BRP (non-deterministic) failure minimum bounded reachability probabilities for state spaces obtained from different sample size and trace length parameters.

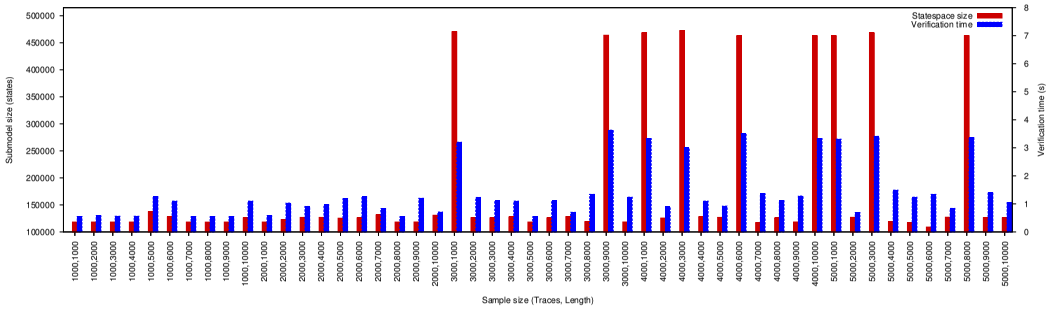
full model comprising about 75 million states. The minimum TAT was estimated at 1725 after executing for just 10 minutes, while the maximum one was calculated to be 4301.65, after 15 hours into the verification process execution. Turnaround time is measured in microseconds (μs).

Again, we compared this performance with our approach. We depict the results obtained for the minimum turnaround estimation in Figure 7.13 and those for maximum turnaround estimation in Figures 7.14 and 7.15. These Figures have been split to ease readability. We also show some of the obtained invariants in Table 7.5. The complete invariant list can be found in Tables A.11 through A.15 in Appendix A.

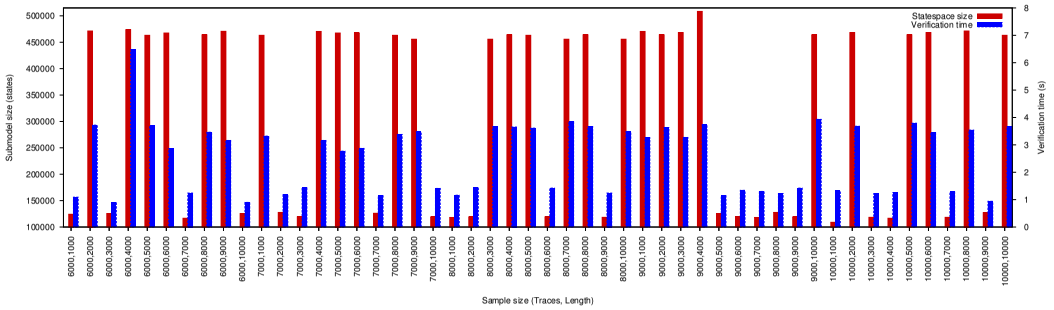
In this case, the results are much easier to interpret. We analyse first the results for minimum turnaround time estimation. These graphs show, for each combination of sample size and trace length, the size of the obtained submodel (in red), and the verification time. In every case the minimum turnaround time estimated was 1725.00, which coincides with the actual minimum.

Estimation of this minimum reward was also very efficient, requiring no more than 7 seconds for every case, while several of the estimations were completed in much less time, about 1 second. From the Figures it is clear that there is a direct correlation between the verification time and the submodel sizes. We can group the submodels in roughly two groups: those that comprise about 120000 states, and those that grow to about 4500000 states. The former required only 1 second of verification while the latter were closer to 7 seconds. We will explain these size differences when we take a look at the inferred invariants.

In the case of the maximum turnaround estimation, the submodels do not estimate the exact value. However, all estimations differ in no more than 1.25% from the actual value estimated through full model evaluation, which was 4301.65. Moreover, most estimations are only 0.53% away from the actual value, with only one estimation straying farther away. Figure 7.15 shows the different estimated values for the sample size and trace length combinations.

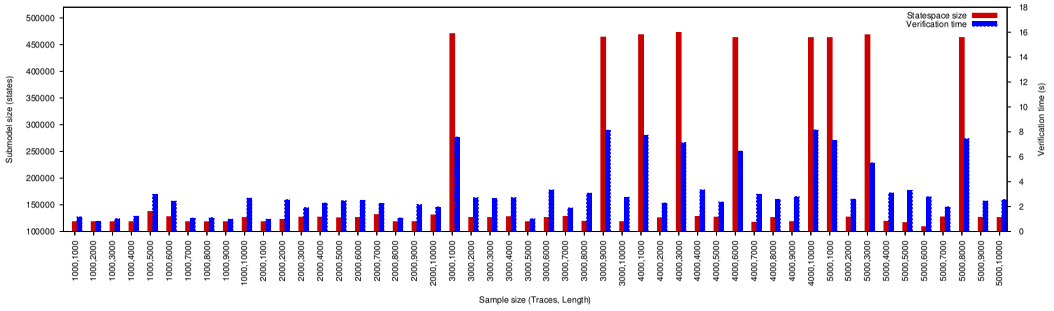


(a) (1000,1000) - (5000,10000)

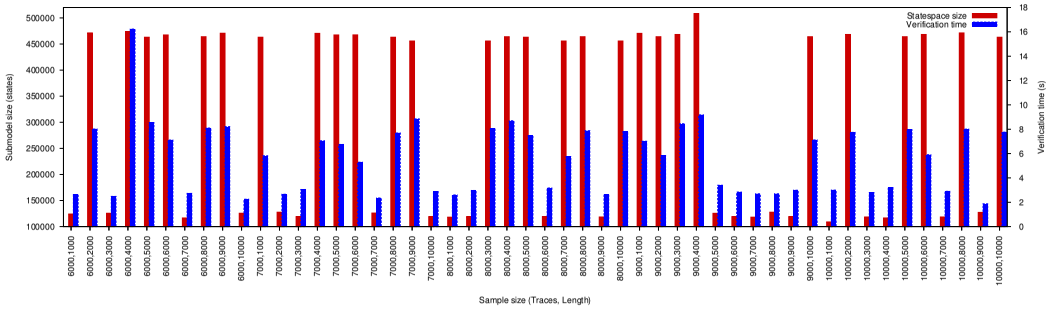


(b) (6000,1000) - (10000,10000)

Figure 7.13: Verification times and submodel sizes for WLAN minimum turnaround estimation

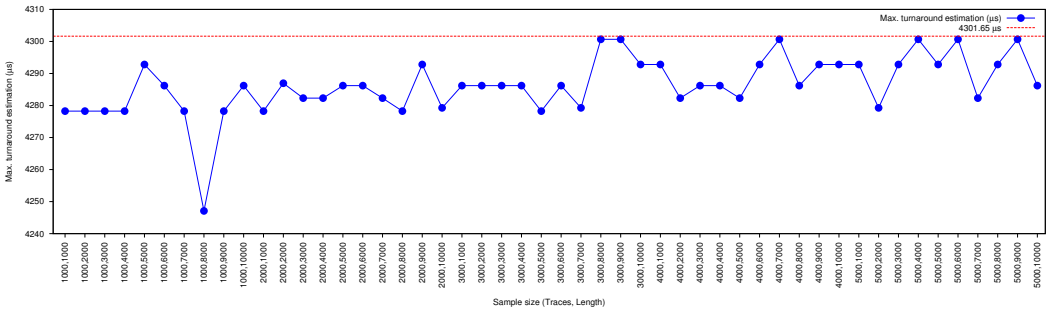


(a) (1000,1000) - (5000,10000)

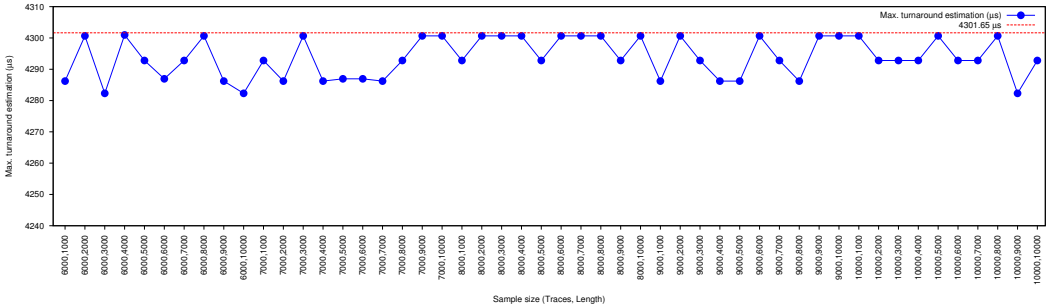


(b) (6000,1000) - (10000,10000)

Figure 7.14: Verification times and submodel sizes for WLAN maximum turnaround estimation



(a) (1000,1000) - (5000,10000)



(b) (6000,1000) - (10000,10000)

Figure 7.15: WLAN maximum turnaround estimation values

The verification times that were necessary for estimating these results are what are most significant. All reward estimations finished in less than 17 seconds, with most of those estimations taking much less time. Again, larger submodel sizes correspond with longer verification times as can be seen in Figure 7.14.

These results mark a stark contrast with the time needed for the full model verification. Recall that minimum TAT calculation over the full model required 10 minutes, while maximum TAT calculation was finished only after 15 hours. Although the partial verification requires an initial simulation and inference step, this time is offset in the case of full model verification by the time required to build the model. Simulation an inference was finished after 2 hours, which is roughly the same time required to build the complete model.

The size of the submodels evaluated is also striking. In all cases, this size is about 0.15% to 0.50% of the size of the whole model. This seems to suggest that the full model has a very large portion of behaviour that is largely irrelevant with regards to their actual contribution to the system's TAT. In fact, it is easy to see from Table 7.5 that although the waiting slots ($slot1$ and $slot2$) can be increased to as much as 128 different slots, the simulations only observed waiting times up to 4 of these slots. Since the slot is chosen equiprobably within the same backoff level, this seems to suggest that only the first two backoff levels were taken on all of the simulated executions. In other words, it was never necessary to increase the backoff to more than this second level.

As in the previous case study, the choice of parameters for the number of traces to simulate and the length of the simulated paths also plays a role. However, this is not as clear-cut as in the previous case. Note that the size of the submodels evaluated seems to lie either near the 120000 state mark except for a few that lie near the 460000 state mark, yielding a partial state space that is roughly 4 times as large as the others. This also explains the discrepancy on the estimation times. When the larger submodels were analysed, the calculations took nearly 7 times as much time

Traces	Length	States	Invariant
5000	1000	463087	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
10000	1000	108964	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
5000	2000	127016	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
10000	2000	468306	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
5000	3000	468306	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
10000	3000	118393	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
5000	4000	119096	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
10000	4000	116549	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
5000	5000	116549	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
10000	5000	464017	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
5000	6000	108964	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
10000	6000	468306	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
5000	7000	127195	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
10000	7000	118393	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
5000	8000	463087	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
10000	8000	471284	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
5000	9000	126363	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
10000	9000	127195	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
5000	10000	126010	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
10000	10000	463087	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$

Table 7.5: WLAN collision avoidance model - Selection of submodel sizes and invariants for different parameter configurations.

as the other estimations. The estimated values, however, were not much better than the ones estimated over smaller partial state spaces. They all yielded an estimate equal to the actual value in the case of the minimum turnaround time. In the case of the maximum turnaround time, estimations over the larger partial state space did

produce a value that is closer to the actual value than the other estimations, but this difference is only marginal.

We may, however, find an explanation for such a disparity in the invariants inferred–see Table 7.5. In the cases where a bigger submodel was generated, it turns out that the second sender station was allowed to take the slot number 3 in some of the executions, while in the smaller ones it never did. Since the choice of slot is uniform, and whenever the slot 2 is available the slot 3 also is, we can only conclude that these differences are only a coincidental artefact of the stochasticity of the sampling procedure.

Network virus infection

Finally, we study the network virus infection scenario. As we described earlier, this network has a cubic grid topology. For these experiments we chose to set the number N of nodes per edge to be 3; that is, the network is comprised of a total of 27 nodes. This is more than enough to quickly deplete all available memory before reaching a full state space. The total potential state space is $3^{27} \sim 7 \times 10^{12}$ states. The actual reachable states are less. For example, a state where every node has its firewall down is unreachable (there should be at least one infected node responsible for having broken the firewall of the last node). However, the reachable states are still enough to make a complete analysis infeasible.

This is a similar situation to that of the BRP case study. Therefore we focus on partial explorations only. We will show, however, that in this case we have a way of computing the values of interest in an analytical manner.

We start out with a non-deterministic model of the network, since we do not know which distribution (if any) governs the races between the different nodes. At any given point any of the nodes can choose to perform its action. However, we modelled probabilistically the behaviour of each node through a Probabilistic Interface Automaton.

According to the behaviour we modelled, the nodes are quite resistant to attack. An infected node has a 0.01 chance to break a neighbour's firewall. Once this firewall is down, it has a further 0.01 chance to infect it. A healthy node is much more efficient and has a 0.98 chance of repair success. However, all nodes are agnostic respect the status of their neighbours. This means that an infected node may attempt to reinfect an already infected node, and a healthy node may attempt to repair a non-infected one.

Properties of interest The first property of interest is the expected time to total infection of the network. Since the system model is non-deterministic, we will need both a minimum expected time as well as a maximum expected time. However, the maximum expected time is infinite. A fair scheduler may choose to alternatively infect a node, and once it is infected, have a neighbour repair it, and do so indefinitely. Therefore, there exist valid schedulers that avoid attaining total infection. In fact, there are valid schedulers that make infection of any one given node infeasible (apart from the initially infected one). Additionally, given that we are analysing the possibility of failure, it is more interesting to study the worst case (i.e., the fastest possible time to total infection).

In a similar way, we aimed at calculating the probability of achieving total infection of the network before a given time bound. In the evaluation of this property, we regard a time step as a communication operation (firewall break, infection, or reparation attempt) between any two nodes, regardless whether they are successful or

not. Again, the minimum probability of such total infection is 0, given the scheduler described above. Therefore, we are interested in the maximum probability.

Following the same reasoning, we analysed a second pair of probabilistic properties. In this case we wish to calculate the mean time and the (bounded in time) probability of propagating infection from one corner of the cubic grid to the opposite corner. In contrast with the previous case, we do not require full infection.

Analytical solutions Even though we cannot perform a complete model check over the whole system, we can calculate the values of the interesting properties in an analytical manner.

For the first property, the fastest way to achieve total infection is to infect each of the remaining 26 nodes, without allowing for any recovery from the healthy nodes. Recall that infection of a node implies first lowering its firewall. Since the probability of breaking the firewall and infecting a vulnerable node is the same (0.01), the previous analysis amounts to studying a Negative Binomial distribution with parameter 0.99. In order to witness total infection, we need to see 52 (26 firewall breaks + 26 infections) failure events. Therefore the expected time to total infection is $52/0.01 = 5200$.

In the case that we give a time bound N for total infection, we can also calculate, for the worst scheduler case described in the previous example, the probability of failure before time N. This is given by the cumulative distribution function (CDF) of the Negative Binomial distribution (CDF_{NB}), which is given by

$$\begin{aligned} CDF_{NB}(N, 52) &= \sum_{k=1}^{N-52} P(52 \text{ successes and } k \text{ failures}) \\ &= \sum_{k=1}^{N-52} \binom{k+52-1}{k} 0.01^{52} 0.99^{k} \end{aligned}$$

For example, the probability of total infection at time at most 5200 (the mean expected time) is ~ 0.51872.

The case for corner infection is similar. We can calculate the mean time to corner infection, since the worst scheduler is the one that takes the fastest vector of infection from one corner to another. This involves infecting just 6 nodes to reach the opposite corner. The expected time to corner infection and the probability of corner infection before a certain time bound follow the same distributions as before. Following these known distributions, it turns out that the expected time to infection of the opposite corner is 1200, and the probability of infecting it before this mean time is 0.53898.

Partial exploration approach results As we did with the other case studies, we put our approach to the test. Although we managed to obtain correct results, in this case the values obtained turned out to lie far from the actual values. Using our standard simulation parameters of simulating 1000-10000 traces of 1000-10000 steps each, we always obtained submodels for which *i)* the bound to mean time to total infection was ~ 200; *ii)* the bound to probability of total infection before 5000 steps was very close to 1; *iii)* the bound to mean time to corner infection was again ~ 200; and *iv)* the bound to probability of total infection is again close to 1. These results are a consequence of the simulated traces not capturing enough of the system's behaviour. This is caused, in turn, by the strongly non-deterministic nature of the model. It happens that, at any given point in simulation, there exist several possible actions to take. Namely, since each node is unaware of its neighbours status, each node can try to break or infect its neighbours (if itself is infected), or repair

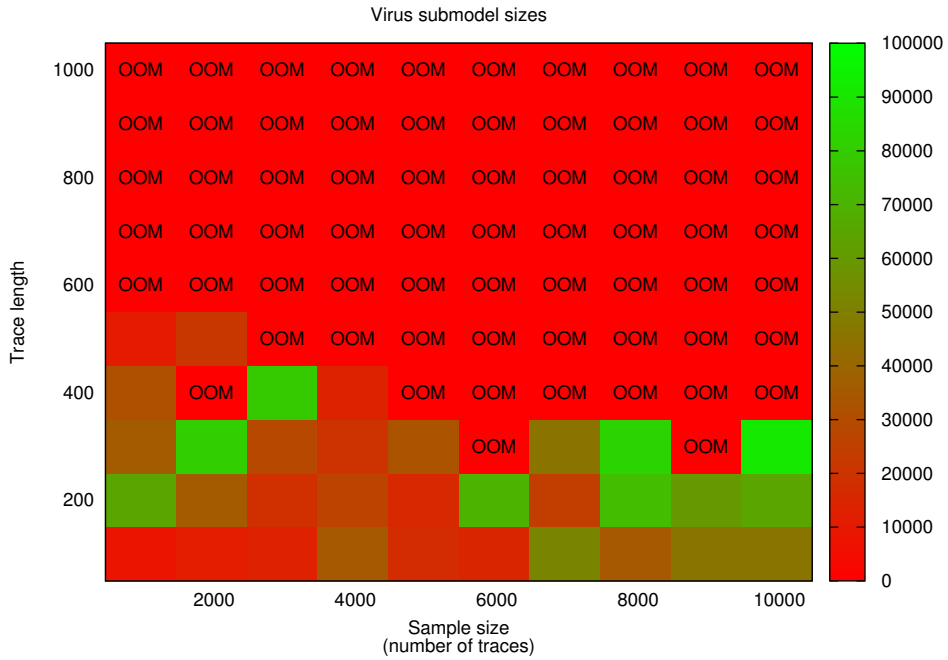


Figure 7.16: Sizes of submodels of the Virus infection model for different simulation parameters. OOM denotes submodels that exceeded available memory.

it (if it is not infected). At each point, there are in excess of 27 choices possible, each with a simulation probability of $1/27 = 0.03737$. This makes it extremely unlikely that a simulation will even infect 2 nodes. In fact, the probability of a simulation immediately infecting two nodes is $(0.03737 \times 0.01)^4 = 1.95 \times 10^{-14}$. Even taking into account that a simulation can take up to 10000 steps, the probability still remains extremely small. This results in submodels that describe very little behaviour. However, the results are still correct, although arguably not as useful as in the other cases.

In order to be able to perform a more meaningful analysis, we modelled a second version of the virus infection where we restricted some behaviour. This second model introduces two changes. First, the nodes do not perform repair operations. Therefore, once a node is infected, it stays infected. Second, nodes are aware of their neighbours status. As a result, infected nodes do not try to break broken neighbours, and do not try to infect infected neighbours. These two changes significantly constrain the model, and reduce both the number of reachable states as well as available transitions. Interestingly enough, the analytical results for the extreme case still hold the same values, as the analysis is still valid under this constrained model.

From initial experimentation it was clear that running simulations as long as those we performed for the previous case studies yielded submodels that were still large enough to be infeasible to analyse. Therefore, we reduced the length of simulations for this case study. The results we present in this section were obtained by performing simulations where the number of traces varied between 1000 and 10000 (stepping size by 1000), and the traces were between 100 and 1000 steps long (stepping size 100). Even with this model simplification and simulation parameters adjustment, we also ran into cases where memory was not enough to hold the submodel. Figure 7.16 shows these results.

As a result we only report results on those submodels that we could analyse. Figures 7.17 and 7.18 show the bounds on minimum and maximum expected time to total and corner infection, respectively, along with the time taken to arrive to those

Traces	Length	States	Invariant
1000	100	7728	$s123 = s132 \wedge s123 = s133 \wedge s123 = s213 \wedge s123 = s222 \wedge s123 = s223 \wedge s123 = s231 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s312 \wedge s123 = s313 \wedge s123 = s321 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 1 \wedge s121 \leq 2 \wedge s122 \leq 1 \wedge s123 \leq 0 \wedge s131 \leq 1 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s221 \leq 1 \wedge s311 \leq 1 \wedge s111 \geq s112 \wedge s111 > s113 \wedge s111 \geq s121 \wedge s111 > s122 \wedge s111 > s123 \wedge true \wedge s111 > s131 \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s221 \wedge s111 > s311 \wedge s112 \geq s113 \wedge s112 \geq s122 \wedge s112 \geq s123 \wedge s113 \geq s123 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s122 \geq s123 \wedge s123 \leq s131 \wedge s123 \leq s211 \wedge s123 \leq s212 \wedge s123 \leq s221 \wedge s123 \leq s311 \wedge s211 \geq s311$
5000	100	17378	$s123 = s132 \wedge s123 = s133 \wedge s123 = s213 \wedge s123 = s222 \wedge s123 = s223 \wedge s123 = s231 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s312 \wedge s123 = s313 \wedge s123 = s321 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 1 \wedge s123 \leq 0 \wedge s131 \leq 1 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s221 \leq 2 \wedge s311 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 > s122 \wedge s111 > s123 \wedge true \wedge s111 > s131 \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s221 \wedge s111 > s311 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s113 \geq s123 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s122 \geq s123 \wedge s123 \leq s131 \wedge s123 \leq s211 \wedge s123 \leq s212 \wedge s123 \leq s221 \wedge s123 \leq s311 \wedge s211 \geq s311$
1000	400	3128661	$s133 = s223 \wedge s133 = s233 \wedge s133 = s313 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 1 \wedge s131 \leq 2 \wedge s132 \leq 2 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 1 \wedge s221 \leq 2 \wedge s222 \leq 2 \wedge s231 \leq 1 \wedge s232 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 1 \wedge s321 \leq 1 \wedge s322 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge s111 \geq s131 \wedge s111 \geq s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 > s231 \wedge s111 > s232 \wedge s111 \geq s311 \wedge s111 > s312 \wedge s111 > s321 \wedge s111 > s322 \wedge s112 \geq s113 \wedge s112 \geq s133 \wedge s112 \geq s232 \wedge s112 \geq s322 \wedge s113 \geq s133 \wedge s121 \geq s131 \wedge s121 \geq s133 \wedge s121 \geq s232 \wedge s121 \geq s322 \wedge s122 \geq s133 \wedge s122 \geq s232 \wedge s122 \geq s322 \wedge s123 \geq s133 \wedge s123 \geq s322 \wedge s131 \geq s133 \wedge s131 \geq s322 \wedge s132 \geq s133 \wedge s132 \geq s232 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s231 \wedge s133 \leq s232 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s321 \wedge s133 \leq s322 \wedge s211 \geq s231 \wedge s211 \geq s311 \wedge s211 \geq s321 \wedge s211 \geq s322 \wedge s212 \geq s213 \wedge s212 \geq s312 \wedge s221 \geq s232 \wedge s222 \geq s232 \wedge s222 \geq s322$
4000	400	13385277	$s223 = s232 \wedge s223 = s233 \wedge s223 = s313 \wedge s223 = s322 \wedge s223 = s323 \wedge s223 = s331 \wedge s223 = s332 \wedge s223 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 2 \wedge s131 \leq 2 \wedge s132 \leq 2 \wedge s133 \leq 1 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 2 \wedge s221 \leq 2 \wedge s222 \leq 2 \wedge s223 \leq 0 \wedge s231 \leq 2 \wedge s311 \leq 2 \wedge s312 \leq 2 \wedge s321 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 \geq s123 \wedge s111 \geq s131 \wedge s111 \geq s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s213 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 > s223 \wedge true \wedge s111 \geq s231 \wedge s111 \geq s311 \wedge s111 \geq s312 \wedge s111 > s321 \wedge s112 \geq s113 \wedge s112 \geq s133 \wedge s112 \geq s223 \wedge s113 \geq s223 \wedge s121 \geq s131 \wedge s121 \geq s133 \wedge s121 \geq s223 \wedge s122 \geq s133 \wedge s122 \geq s223 \wedge s123 \geq s133 \wedge s123 \geq s223 \wedge s131 \geq s133 \wedge s131 \geq s223 \wedge s132 \geq s223 \wedge s133 \leq s211 \wedge s133 \geq s223 \wedge s211 \geq s223 \wedge s211 \geq s311 \wedge s212 \geq s223 \wedge s213 \geq s223 \wedge s221 \geq s223 \wedge s222 \geq s223 \wedge s223 \leq s231 \wedge s223 \leq s311 \wedge s223 \leq s312 \wedge s223 \leq s321$
1000	500	10495696	$s133 = s232 \wedge s133 = s233 \wedge s133 = s313 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 2 \wedge s131 \leq 2 \wedge s132 \leq 2 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 2 \wedge s221 \leq 2 \wedge s222 \leq 2 \wedge s223 \leq 1 \wedge s231 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 2 \wedge s321 \leq 1 \wedge s322 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 \geq s123 \wedge s111 \geq s131 \wedge s111 \geq s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s213 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 > s223 \wedge s111 > s231 \wedge s111 \geq s311 \wedge s111 \geq s312 \wedge s111 > s321 \wedge s112 \geq s113 \wedge s112 \geq s133 \wedge s112 \geq s213 \wedge s112 \geq s223 \wedge s113 \geq s133 \wedge s113 \geq s223 \wedge s121 \geq s131 \wedge s121 \geq s133 \wedge s121 \geq s223 \wedge s121 \geq s322 \wedge s122 \geq s133 \wedge s122 \geq s223 \wedge s123 \geq s133 \wedge s123 \geq s223 \wedge s131 \geq s133 \wedge s131 \geq s223 \wedge s132 \geq s223 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s223 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s321 \wedge s133 \leq s322 \wedge s211 \geq s311 \wedge s211 \geq s321 \wedge s211 \geq s322 \wedge s212 \geq s223 \wedge s212 \geq s322 \wedge s213 \geq s223 \wedge s312 \geq s322$
2000	500	21603820	$s133 = s232 \wedge s133 = s233 \wedge s133 = s322 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 2 \wedge s131 \leq 2 \wedge s132 \leq 2 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 2 \wedge s221 \leq 2 \wedge s222 \leq 2 \wedge s223 \leq 1 \wedge s231 \leq 2 \wedge s311 \leq 2 \wedge s312 \leq 2 \wedge s313 \leq 1 \wedge s321 \leq 2 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 \geq s123 \wedge s111 \geq s131 \wedge s111 \geq s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s213 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 > s223 \wedge s111 \geq s231 \wedge s111 \geq s311 \wedge s111 \geq s312 \wedge s111 > s313 \wedge s111 \geq s321 \wedge s112 \geq s113 \wedge s112 \geq s133 \wedge s112 \geq s223 \wedge s112 \geq s313 \wedge s113 \geq s133 \wedge s113 \geq s313 \wedge s121 \geq s131 \wedge s121 \geq s133 \wedge s121 \geq s223 \wedge s121 \geq s313 \wedge s122 \geq s133 \wedge s122 \geq s223 \wedge s123 \geq s133 \wedge s131 \geq s133 \wedge s132 \geq s133 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s223 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s313 \wedge s133 \leq s321 \wedge s211 \geq s223 \wedge s211 \geq s311 \wedge s211 \geq s312 \wedge s211 \geq s313 \wedge s213 \geq s313 \wedge s221 \geq s223 \wedge s222 \geq s223$

Table 7.6: Selection of virus infection submodel sizes and invariants for different parameter configurations.

results (note that the scale for minimum and maximum estimations changes). In the case of the minimum estimation, the results are still not close to the actual values, but are much more informative than in the more relaxed case.

Perhaps a yet more interesting result from these graphs is that the obtained values are exactly the same both for the total infection property as well as the corner infection, although verification times are higher for the total infection case due to the added complexity of the formula that describes this total infection. This suggests that the bound is being calculated to the point of reaching the trap state rather than the actual infection states. This is confirmed by the invariants obtained, that effectively prune the infection states out of the partial state space. Table 7.6 shows a subset of the invariants for the submodels where partial verification was feasible.

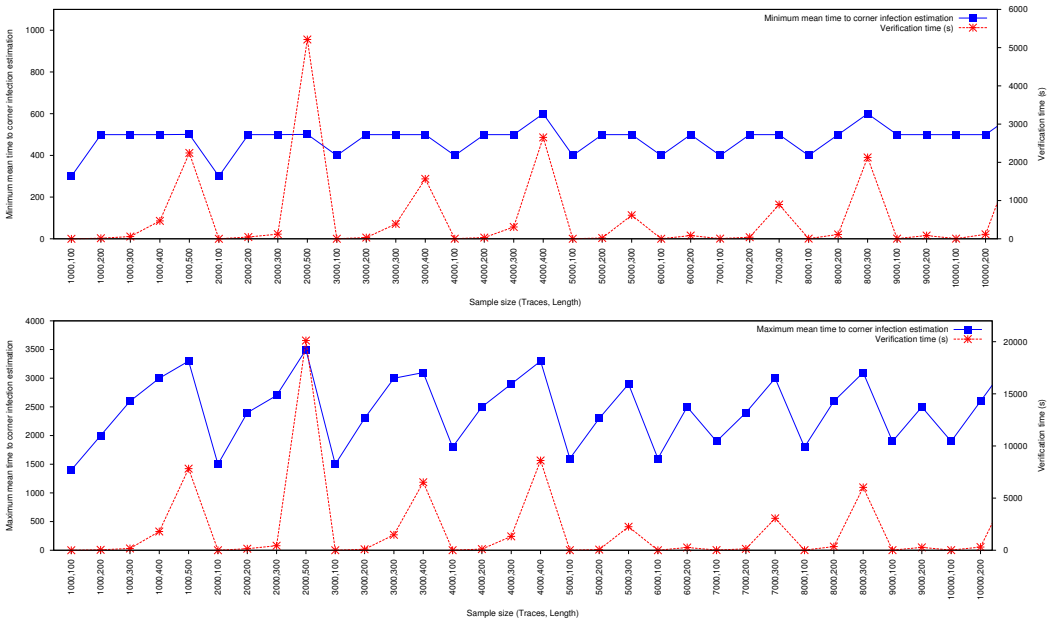


Figure 7.17: Minimum and maximum mean time to total infection. Bounds calculated on submodels obtained through combinations of traces and trace lengths.

The complete set of feasible invariants can be found in Tables A.16 through A.19 in Appendix A.

We also attempted to calculate bounds to maximum bounded probabilities of total and corner infection. In this case, however, the results do not improve much and are not very informative, as all the obtained values are very close to 1. This result showcases that the simulation step was not able to capture a partial statespace that is representative of usual behaviour or, alternatively, that the usual behaviour is not concentrated around a minority of the complete state space.

Further in this chapter we will compare the performance of our approach to that of the Monte Carlo approaches and submodels obtained through manually provided invariants.

Summary of analyses

What all case studies and experiments indicate is that, through careful partial exploration of the model, we can obtain useful bounds for reward estimation and reachability probabilities with very low percentages ($< 1.5\%$) of the actual state space explored. Further, submodels that yield these results also converge very quickly (much before the 24 hour timeout) to good estimation results.

In the case of estimations that did not converge, it turned out that while they do constantly improve during the rest of the 24 hours, they do so at a much slower pace than at the beginning. This was the case for some of the reward estimations. Far from being a problem, this turns to be good news, as even with the trace analysis, good results can still be attained under the same time budget. From these results it follows that, for these case studies, effort into estimating reward values through automatically obtained submodels through model invariants of the full model pays off.

It must be noted that it is possible that the *actual* value of the reward being estimated is much larger than any of those obtained. Of course, we are always limited by the fact that the actual reward value cannot be calculated, neither with

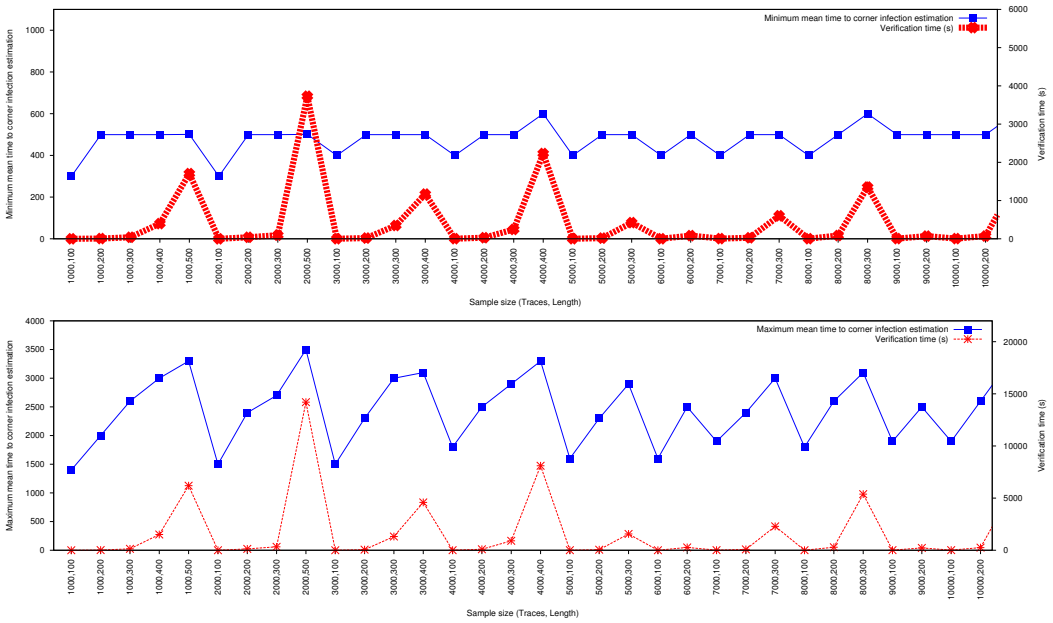


Figure 7.18: Minimum and maximum mean time to corner infection. Bounds calculated on submodels obtained through combinations of traces and trace lengths.

partial nor full models. It can be argued, though, that it is often the case that the exact value is not needed as such; rather, satisfying a minimum threshold value is a sufficient guarantee for the reliability measure being analysed. Hence, methods which provide higher lower bounds faster are potentially useful.

It is also interesting to note that the efficiency of our proposed approach does not seem to depend on whether the states tested for reachability are actually reachable in the submodels or not. For example, in all of the Tandem Queue, BRP and Virus infection cases, the inferred invariants preclude the failure states from appearing in the submodels. However, in the case of the WLAN protocol, the interesting states which describe the property of interest are not completely cut out from the submodels by the invariant.

7.3.2. Question 2

Monte Carlo estimation of system properties

Contrasting to the previous experimentation that aimed to compare our approach with probabilistic model checking, Q2 aims to establish a comparison with Monte Carlo techniques. Experimentation to answer this question is not straightforward due to the problem of generating sufficient failing simulations to ensure given precision and confidence parameters. We first aimed at performing a straightforward statistical analysis of the model. A first experiment was designed requiring a result precision of 99%. As is standard for statistical analyses, we also required a 95% confidence.

A straightforward calculation of the necessary sample size based on the Chernoff bound [Che52] determines that a total of $\sim$ 60000 samples are necessary, which does not seem excessive. However recall that each sample must eventually reach a state where the property can be determined to be true or false. For systems where witnessing this behaviour is rare, this means that samples may be extremely long. Through trial and error, and based on the bounds obtained in *Q1*, we tried to determine the minimum length for samples to consistently reach failure states. For the Tandem Queue full model—for which its MTTF was already estimated to be at

least 7×10^7—even samples as long as 4×10^8 do not consistently reach the failure state where the queues are both full. Considering that generating a sample of such length takes 15 minutes, generation of the full 60000 traces required leads to a 2 year period for sample generation. A similar situation is found upon analysis of the BRP model.

Relaxing the precision requirement to 95% reduces the sample generation cost to 1 month. Further relaxation to 90% still requires a week of execution. In fact, if we were to set a 24 hour budget for sample generation, the precision obtained would be of just 70%. That is, the MTTF estimate would be up to $\pm 30\%$ away from the true MTTF value with a 95% guarantee. Note that this is a very conservative estimate as it is unlikely that all traces of length 4×10^8 generated in the 24 hour period will consistency reach failure states, and possibly much lengthier traces will be needed.

To overcome this limitation of standard Monte Carlo verification, we tried carrying out a variation of Wald's sequential testing [Nim10]. This procedure generates samples while at the same time it determines whether more samples are necessary or not. As a result of this online estimation, it might require less samples than those mandated by the Chernoff bound, although it cannot be stated beforehand how many samples will be needed exactly. This optimization does not eliminate the need for samples to reach property-determining states, so sample length remains a problem. We attempted to perform this analysis truncating generated samples at length 4×10^8 and treating them as *failing* samples once they reached this threshold. This is a similar strategy as the one used in our approach (anything beyond the submodel is a failure). However, this procedure yielded no results after 24 hours of execution, indicating that the sequential testing still needed more evidence in order to produce a reasonable estimate.

In the case of the bounded probability properties, the main difference is that the horizon for trace simulation length is already set by this bound. This represents an advantage with respect to the previous properties. However, recall that we aim at calculating these probabilities for meaningful bounds, that is, bounds that bear some resemblance to those already obtained by our partial verification approach. For both the Tandem Queue and BRP case studies, we set this bound to be 10^7. Such a length makes generation of 60000 samples also prohibitive. Setting a confidence parameter of 95%, sampling would require at least a month of computer time.

As a final approach to this strategy of over-approximation of failures in Monte Carlo verification, we generated samples over the submodels with highest MTTF obtained in *Q1* rather than over the full model. However, the problem of producing samples that consistently fail persisted, failing to provide an estimate for MTTF in the budgeted time. These results suggest that Monte Carlo approaches may be unsuitable to answer reliability questions in systems with high MTTF (i.e., rare failures). Monte Carlo approaches are not suitable either for probability estimation in these cases.

Monte Carlo evaluation of non-rare events

As we have seen in the previous discussion, Monte Carlo approaches are not amenable to scenarios where the properties of interest entail rare events. However, in both the WLAN collision avoidance protocol and Virus infection case studies, the event under analysis is not a rare event at all. This makes Monte Carlo analyses for these cases presumably feasible.

We first set out to estimate the minimum and maximum turnaround times for the WLAN collision avoidance protocol. Recall that we already analysed this model completely and found these times to be 1725 and 4301.65 for the minimum and

maximum cases respectively. In the previous section, we already established that 60000 samples would be necessary for a robust estimation. Since we know that the maximum expected turnaround time is ~ 4300, we set the trace horizon to 10000 in order to have a reasonable confidence that every trace would hit the success state (i.e., one where both stations have sent their data successfully).

The obtained results are disconcerting, however. In both cases, the estimation procedure was efficient, as it only required 80 seconds of execution in both cases. The reason for this fast sample generation is that not only is the bound low, but the required property is reached on an average of 25 steps as well. This is because, unlike the other case studies, the reward structure for the WLAN case assigns a reward of at least 50 to transitions. Because of these reasons, most samples are very short and are generated very quickly.

The estimations themselves are the problem in this case. For the minimum turnaround estimation, we obtained a time of 2729.45 ± 0.1929 with 95% confidence. Surprisingly enough, the estimation for the *maximum* turnaround is extremely similar: 2731.06 ± 0.1915 with 95% confidence. Not only are both results the same, they are equally incorrect.

The estimation analysis for the Virus infection case does not fare better. We have already noted that we could calculate the minimum time to complete infection and the minimum time to infection of the opposite corner network node in an analytical way. We already calculated these expected times to be 5200 and 1200 respectively. However, we know that the maximum expected time is actually infinite. This makes the setting of a trace horizon as difficult as in the Tandem Queue and BRP cases. In fact, experimentation showed that traces as long as 10^7 steps long do not consistently reach the target state. This situation renders the estimation analysis as infeasible as in the BRP and Tandem Queue cases.

On the other hand, since the minimum bounds are low enough, we set out to use them as bounds for a bounded probability analysis. We performed Monte Carlo estimations of the probability of reaching total infection before the expected 5200 steps, and the probability of infecting the opposite node before the 1200 steps expected in that case. The results for these analyses are included in Table 7.7. Again, it can easily be seen that these results cannot be correct.

These (incorrect) results can be easily explained, however. Both the WLAN collision avoidance protocol and the Virus infection system share the trait of being non-deterministic. Unfortunately, Monte Carlo approaches are not very good at dealing with non-determinism [HMZ+12]. The reason is that simulated executions, when faced with a non-deterministic choice, are at a loss regarding which transition to choose next. The simple approach taken in these cases is to choose one of the available transitions uniformly.

This uniform choice explains why both minimum and maximum estimations resulted in the same values. Since the neither the *best* nor the *worst* schedulers are uniform in their choice, these extreme behaviours are not witnessed, and therefore cannot be estimated. The second problem is that turning a non-deterministic choice into a probabilistic one introduces a bias that cannot be estimated itself. As a result, estimation results when non-determinism is present are meaningless.

Surprisingly, this uniformity also explains why the Monte Carlo approach yielded a result close to the actual one in the case of total virus infection, but not in the case of corner infection. In the case of total infection, since every node needs to be infected, every non-deterministic choice needs to be taken. Since the Monte Carlo simulations are more or less uniform in resolving non-determinism, they turn out to actually be selected, and therefore provide a result close to the true one. However, in the

Property	Known value	Time to estimation	Estimation
WLAN minimum turnaround	1725.00	81.67 sec.	2729.45 ± 0.1929
WLAN maximum turnaround	4301.65	80.22 sec.	2731.06 ± 0.1915
Max. prob. of total network infection before 5200 steps	0.51872	20 *hours*	0.00 ± 0.00
Max. prob. of corner infection before 1200 steps	0.53898	4 *hours*	0.00 ± 0.00
Max. prob. of total network infection before 5200 steps (constrained model)	0.51872	693.34 sec.	$0.54200 \pm 9.8 \times 10^{-4}$
Max. prob. of corner infection before 1200 steps (constrained model)	0.53898	166.23 sec.	0.00 ± 0.00

Table 7.7: Monte Carlo estimations for the WLAN collision avoidance protocol and Virus infection systems.

case of corner infection, only non-deterministic options that lead to advance towards the corner have to be selected. This is not the case for uniform non-determinism resolution, and therefore the (wrongly) estimated probability is 0.

In the next chapter we will discuss some recent research that has attempted to provide some alternatives to attack this problem.

Summary of Monte Carlo analyses

The conclusion of the previous analyses is that applicability of Monte Carlo techniques is limited to those cases where *i*) the property under analysis is both known to be realised, and not a rare event; and *ii*) the system under analysis does not exhibit non-determinism. These restrictions rule out a large class of interesting system behaviour. In fact, the case studies presented in this section are representative of somewhat common behaviour, but they are not amenable to Monte Carlo based analyses.

Our partial evaluation technique, however, obtains meaningful results for each of these cases, even though in the case of the Virus infection these are not as useful as in the other cases.

7.3.3. Question 3

In this section, we compare the results obtained while answering *Q1* with the results a practitioner might obtain by specifying invariants herself, based on her knowledge of the model. Prior to experimenting on automatically generated invariants, we analysed the models and came up with at least one invariant for each one. These invariants were selected based on our understanding that their negation is a necessary condition for reaching failure states. In particular, we manually inspected each model looking for variables that we believed, a priori, would increase as the

c	Size	MTTF		Bounded reach. prob.	
		Value	Time	Value	Time
20	2398 st 6560 tr	$0.83{\cdot}10^{3}$	68.75 s	1.00000	544.47 s
40	8778 st 24280 tr	$1.12{\cdot}10^{4}$	82.72 s	1.00000	1984.27 s
60	19158 st 53200 tr	$1.25{\cdot}10^{5}$	276.69 s	1.00000	4351.62 s
80	33538 st 93320 tr	$1.36{\cdot}10^{6}$	64.06 m	0.97873	7768.94 s
100	51918 st 144640 tr	$1.49{\cdot}10^{7}$	17.93 h	0.29723	12734.36 s
120	74298 st 207160 tr	$5.50{\cdot}10^{7}$	TO	0.03181	21985.64 s
140	100678 st 280880 tr	$4.63{\cdot}10^{7}$	TO	0.00296	26322.19 s
160	131058 st 365800 tr	$3.17{\cdot}10^{7}$	TO	2.71×10^{-4}	49119.85 s
180	165438 st 461920 tr	$2.31{\cdot}10^{7}$	TO	2.48×10^{-5}	44882.64 s
200	203818 st 569240 tr	$1.66{\cdot}10^{7}$	TO	2.28×10^{-6}	68516.20 s
900	4067118 st 11381440 tr	$8.41{\cdot}10^{5}$	TO	0.0000	TO
1600	11219198 st 31407194 tr	$4.20{\cdot}10^{5}$	TO	0.0000	TO
2400	14362898 st 40213194 tr	$4.20{\cdot}10^{5}$	TO	0.0000	TO

Table 7.8: Experimental results for tandem queue (2×1200 processes) mean times to failure and bounded reachability probabilities.

execution grew closer to the failure state. Once these variables were identified, we wrote invariants stating upper bounds for their possible values.

Manual invariant analysis of Tandem Queue

For the Tandem Queue case study, we established the invariant to be that the total number of enqueued processes globally in both queues is less than c, and ran experiments for different values of c ranging up to the total capacity of the queueing system ($2 \times C$). A failure entails that the invariant does not hold for $c < 2 \times C$, and that for $c = 2 \times C$ the resulting invariant-driven submodel is exactly the whole model. In our experiments we found that there exist multiple c values for which the invariant resulted in a significantly higher MTTF than the MTTF estimated for the full model.

Table 7.8 summarises the results obtained for various submodels derived from different values for parameter c of this manual invariant. From the table it follows that the best MTTF is obtained for the submodel which considers up to 120 processes queued (MTTF $> 5.5{*}10^{7}$), and the best bound on the bounded reachability property was 2.28×10^{-6}. Here we only take into account results for which convergence was attained. These results are summarised in Table 7.8.

retries	Size	MTTF		Bounded reach. prob.	
		Value	Time	Value	Time
1	366915 st 489574 tr	$1.50{\cdot}10^6$	21.06 h	0.99870	5.70 h
2	480460 st 646758 tr	$1.69{\cdot}10^7$	TO	0.01319	7.87 h
5	821095 st 1118310 tr	$1.08{\cdot}10^7$	TO	0.0000	TO
10	1388820 st 1904230 tr	$6.29{\cdot}10^6$	TO	0.0000	TO
50	5930620 st 8191590 tr	$1.39{\cdot}10^6$	TO	0.0000	TO
150	17285120 st 23909990 tr	$4.86{\cdot}10^5$	TO	0.0000	TO
250	28639620 st 39628390 tr	$2.73{\cdot}10^5$	TO	0.0000	TO
256	N/A st N/A tr	N/A	OOM	N/A	OOM

Table 7.9: Experimental results for probabilistic BRP (256 retries) mean times to failure and bounded reachability probabilities.

Manual invariant analysis of Bounded Retransmission Protocol

In the case of the Bounded Retransmission Protocol case study, a parametric invariant chosen was that the number of retries performed while transmitting a single file was less than $max_{retries}$. We ran experiments for different values of $max_{retries}$ ranging up to the true maximum number of retries (256). A failure entails that the invariant does not hold for $max_{retries} < 256$. For $retries = max_{retries}$ the resulting invariant-driven submodel is the whole model.

Again, we show a selection of submodels ranging from the very small upwards to almost the complete model. Results for these experiments are depicted in Table 7.9. Estimation results are even more significant than for the previous case study considering that analysis of the full model with 256 retries was not possible within the memory budget. However, the trend indicates that augmenting the number of retries considered does not yield better MTTF and in fact, a very low number of retries gives a much higher MTTF. A similar conclusion can be obtained from the reachability properties, where for a low retry limit we get a bound of 0.01319. Larger models fail to converge in a timely fashion.

We also performed the same analysis for the non-deterministic version of the protocol environment, with similar results, depicted in Table 7.10.

Manual invariant analysis of WLAN Collision Avoidance

Although the WLAN collision avoidance protocol could be verified in its totality, we nevertheless ventured an invariant that we thought would be useful in reducing the state space. It turns out in this case that our proposed invariant is much simpler than those inferred by the automatic approach, as our initial belief was that bounding the time a sending station is forced to backoff, the model would be reduced. This interpretation, however, turned out to be erroneous. In fact, regardless of how many times a sending station found a collision, the backoff time is chosen uniformly over the whole possible range. The results we obtained by applying these invariants are

retries	Size	MTTF				Bounded reach. prob.			
		Min.	Time	Max.	Time	Min.	Time	Max.	Time
1	279582 st 358393 tr	9965.87	46.26 s	OOM	N/A	0.99805	12.74 h	0.99871	12.92 h
2	393127 st 515577 tr	9998.93	57.51 s	OOM	N/A	0.01239	17.52 h	0.01321	16.22 h
5	733762 st 987129 tr	9999.00	126.25 s	OOM	N/A	0.00000	TO	0.00000	TO
10	1301487 st 1773049 tr	9999.00	190.43 s	OOM	N/A	0.00000	TO	0.00000	TO
50	5843287 st 8060409 tr	9999.00	904.36 s	OOM	N/A	0.00000	TO	0.00000	TO
150	17197787 st 23778809 tr	9999.00	2943.54 s	OOM	N/A	0.00000	TO	0.00000	TO
250	28552287 st 39497209 tr	9999.00	4412.72 s	OOM	N/A	0.00000	TO	0.00000	TO
256	N/A st N/A tr	N/A	OOM	OOM	N/A	OOM	N/A	OOM	N/A

Table 7.10: Experimental results for non-deterministic BRP (256 retries) mean times to failure and bounded reachability probabilities.

`backoff1` and `backoff2` bounding		Model checking			
Max. backoff time	States	Min. TAT	Time	Max. TAT	Time
0	59185713	465.97	109.36s	1201.71	176.88s
5	64160812	559.68	206.37s	1273.44	224.98s
10	68239697	686.78	304.47s	1460.94	286.30s
15	71431132	901.65	440.29s	1764.45	364.90s
20	73735117	1157.81	614.94s	2244.19	435.67s
25	75151652	1392.19	641.49s	2922.23	781.37s
30	75680737	1665.63	490.05s	3846.17	1085.87s

Table 7.11: Selection of WLAN submodel TAT evaluation results for different manual invariants.

presented in Table 7.11. Note that even restricting the backoff time to just one value (zero) does not really reduce the size of the model. Although for smaller values of this bound the verification time is reduced drastically, these execution times are still much larger than those that result from the automatically inferred invariants. Further, the turnaround times obtained, both minimum and maximum, are very poor contrasted with those that resulted from the automatic approach.

Manual invariant analysis of Virus infection

Finally, we turn our attention to the Virus infection model. The manually stated invariants in this case deal with limiting the number of infected nodes that can coexist at once. We first applied these invariants to the original, unconstrained model. As was the case with the results obtained with our approach, these manually inferred invariants can't restrict the model size enough. Setting the limit to just two infected nodes, we quickly obtained a bound to minimum mean time to failure of ~ 200, the same value obtained with our approach. However, raising this limit to three infected nodes makes analysis infeasible.

Consequently, we applied these same manual invariants to the constrained infection model. The results of these analyses are pictured in Tables 7.12 and 7.13.

In this case, it can be seen that these manually posed invariants perform slightly better than the automatically inferred ones. More specifically, increasing the limit of

# infected	Size	Min. time to infection		Max. bounded reach. prob.	
		Value	Time	Value	Time
1	74 st 222 tr	199.88	$\sim$ 0.00 s	1.00000	0.02 s
2	1269 st 5233 tr	399.69	0.07 s	1.00000	0.23 s
3	19181 st 99607 tr	599.55	0.64 s	1.00000	3.97 s
4	351990 st 2215026 tr	799.43	17.88 s	1.00000	93.51 s
5	6035220 st 44517828 tr	999.32	414.14	1.00000	1822.42 s
≥ 6	N/A st N/A tr	N/A	OOM	N/A	OOM

Table 7.12: Experimental results for mean times to total infection; and its bounded reachability probability.

infected nodes by one results in model size increases that do not grow as dramatically as in the case of growing the number of traces and their length in the automatic approach. This allows for better submodels to be obtained and therefore better bounds, up to 5 infected nodes. On the other hand, the obtained bounds on times to failure and probabilities are still far from the actual values.

# infected	Size	Min. time to infection		Max. bounded reach. prob.	
		Value	Time	Value	Time
1	74 st 222 tr	199.88	$\sim$ 0.00 s	0.99992	$\sim$ 0.00 s
2	1269 st 5233 tr	399.68	0.07 s	0.99777	0.05 s
3	19181 st 99607 tr	599.55	2.79 s	0.97998	0.92 s
4	351990 st 2215026 tr	799.43	51.71 s	0.91072	21.56 s
5	6035220 st 44517828 tr	999.32	1241.86 s	0.75805	420.44 s
≥ 6	N/A st N/A tr	N/A	OOM	N/A	OOM

Table 7.13: Experimental results for mean times to corner infection; and its bounded reachability probability.

Summary of manually-inferred invarianty analyses

Here we compare the performance of our automatic approach to that of manual invariants. In each case we take the best result for both the automatic approach and the manual one. Note that in both cases we tried several parameters to arrive to this best result, and in both cases it is difficult to predict which parameters will perform best.

In the cases where the manual invariants did succeed, the Tandem Queue and BRP protocol, it is interesting to note that for relatively small submodels (e.g. $c = 80$

on the Tandem Queue case study, and $max_{retries} < 2$ for BRP) the estimated MTTF is much higher than the MTTF computed over the complete model. Still, while the manual invariant approach did provide useful bounds, it turns out that the best MTTF values generated by the automatic approach obtains slightly higher bounds for the same time budget. For the Tandem Queue study, the best automatically estimated MTTF is of $\sim 7 \times 10^7$ against $\sim 5.5 \times 10^7$.

For the fully probabilistic BRP case study the best automatic estimation is $\sim 2.5 \times 10^7$ versus $\sim 1.69 \times 10^7$ when manual intervention is applied. In the case of non-deterministic environments for the BRP system, the results obtained with manual invariants are notoriously different from the ones yielded by our automatic technique. Manual invariants fail to obtain good bounds to both minimum and maximum mean times to failure. Our interpretation of these results is that bounding the sizes of the sent files yields better submodels than bounding only the number of retries.

The results obtained for the WLAN case study are also far from those of the automatic approach, suggesting that there are complex interactions between the model variables that may be out of reach to a manual inspection and attempt at suggesting invariants.

The case of the Virus infection model is atypical, as the manually posed invariants slightly outperformed the automatically inferred ones.

An initial interpretation of the results would suggest that, except for the non-deterministic BRP and WLAN case studies, automatically inferred invariants do not have an added advantage over manually suggested ones. This is evidenced more starkly in the Virus infection case. However, there is an added cost in understanding a protocol model and being able to suggest which factors are the most relevant in increasing a model size or in making numerical computation infeasible. This cost is in general not trivial, and requires a thorough understanding of the modelling formalisms as well as the verification procedures under the hood. These are not, a priori, traits that every engineer can be reasonably expected to have.

Summary of case studies results by technique

Table 7.14 summarises the results obtained for each case study and property, with each of the established approaches, including our partial exploration one. We highlight the best performer for each case. We first provide, if it was attainable, the actual value of the property analysed. This was either obtained analytically or through a full model check that converged, as described in each case study section. We then compare the results obtained through each approach, as follows

- **Full** is a model checking effort over the full model. We always report on bounds on rewards obtained, if any. Bounds on probabilities are only reported if convergence was attained.
- **Partial** denotes our approach, ignoring simulation times. Since we also perform a model checking step, we omit probabilities that did not converge in time.
- **Monte Carlo** denotes the statistical estimation based on trace simulation.
- **Manual** describes the best result obtained by any of the manual invariants posed for the case study.

In the case of ties, or very close results with very disparaging running times, we opted to report the fastest performer as the best result. We mark these cases with an asterisk.

Tandem Queue (mean time to failure)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	4.2×10^5	TO	**7×10^7**	**TO**	N/A	TO	5.5×10^7	TO
Tandem Queue (bounded reachability probability)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	0.0000	TO	0.0713	TO	N/A	TO	**2.28×10^{-6}**	**19 hs**
Fully probabilistic BRP (mean time to failure)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	OOM	TO	**2.5×10^7**	**TO**	N/A	TO	1.69×10^7	TO
Fully probabilistic BRP (bounded reachability probability)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	OOM	TO	0.0680	22 hs	N/A	TO	**0.01319**	**7.9 hs**
Non-deterministic BRP (*minimum* mean time to failure)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	OOM	TO	**5.6×10^6**	**TO**	N/A	TO	9999	126.25 s
Non-deterministic BRP (*maximum* mean time to failure)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	OOM	TO	**9.8×10^6**	**TO**	N/A	TO	9965.87	46.26 s
Non-deterministic BRP (*minimum* bounded reachability probability)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	OOM	TO	**0.02382***	**8.6 hs***	N/A	TO	0.01239	17.5 hs
Non-deterministic BRP (*maximum* bounded reachability probability)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
Unknown	OOM	TO	0.71205	TO	N/A	TO	**0.01321**	**16.2 hs**
WLAN (*minimum* mean turnaround time)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
1725.00	1725.00	628.00 s	**1725.00**	**0.98 s**	N/A	N/A	1665.63	490.05 s
WLAN (*maximum* mean turnaround time)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
4301.65	4301.65	54149 s	**4300.67***	**2 s***	N/A	N/A	3846.17	1085.87 s
Constrained Virus (*minimum* mean time to total infection)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
5200.00	OOM	TO	500.54	2771 s	N/A	N/A	**999.32**	**414 s**
Constrained Virus (*minimum* mean time to corner infection)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
1200.00	OOM	TO	599.54	1452 s	N/A	N/A	**999.32**	**1242 s**
Constrained Virus (*maximum* bounded probability to total infection before 5200 steps)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
0.51872	OOM	TO	**1.0000**	**$\sim$ 0 s**	N/A	N/A	**1.0000**	**$\sim$ 0 s**
Constrained Virus (*maximum* bounded probability to corner infection before 1200 steps)								
Actual value	Full		Partial		Monte Carlo		Manual	
	Result	Time	Result	Time	Result	Time	Result	Time
0.53898	OOM	TO	0.97997	1004 s	N/A	N/A	**0.75805**	**420 s**

Table 7.14: Summary of (best) results for each technique and case study. TO denotes timeout at 24 hours. N/A denotes results that could not be obtained before timeout or were erroneous due to technique shortcomings.

7.4. Threats to validity

As usually happens with any experimental attempts at validating a new technique, our experiments and their results are subject to threats regarding their validity. We do not foresee threats to construct validity, since our comparison scores in each case are precisely the results of the verification procedures, which is exactly what we want to measure. We do not establish a score function that could confound an outside factor with this measure.

7.4.1. Threats to external validity

The main threat to our experimental approach is that of *external validity*, that is, whether the present experimentation allows us to generalise our conclusions. We have done our best in attempting to perform validating experiments for a range of potential systems. The models which we analysed are very different in nature and the functions they provide are different as well, as are the measures under analysis for each one. Although we cannot affirm that our approach is sure to perform for an arbitrary system as well as we have shown in our experimentation, this variety in case studies under analysis does provide potential users with confidence that the approach may work to their advantage in their setting.

A first threat is that, although the model probabilities present in the model seem to be representative of usual behaviour, they might be inaccurate. They are a result of informed estimations rather than the result of significant observation. We have observed, however, that systems for which their probabilities are very uniform, or for which there exists a large degree of non-determinism, are not extremely well suited to our approach. As we have discussed, this is due to the simulation step failing to discriminate much of the behaviour and exploring so much of the system that a useful size reduction is not achieved.

There also exists a potential threat regarding the convergence of the iterative linear equation resolution methods employed. As was noted, not every one of these methods will converge in every case. In particular, convergence for the best performer (Gauss-Seidel method) is not guaranteed.

This potential problem, however, can be mitigated or downright avoided in several ways. The most obvious way to avoid the problem altogether is abstaining from using iterative methods that do not always converge. For example, the powers method can be safely used in all cases, although its convergence is usually much slower than with the Jacobi or Gauss-Seidel methods. Alternatively, variations of the Gauss-Seidel method such as the Block Gauss-Seidel are known to be convergent if the underlying DTMC is ergodic [SI97, Lan10].

The systems studied in this section are a mixed bag with respect to this property. We have verified through graph-based analysis that both the BRP and WLAN collision avoidance protocol are ergodic. This is not the case of the Tandem Queue system model which has a period of 2. This is intuitive, since to return to the empty queue state it is ostensibly needed to perform the same number of *push* and *pop* operations. However, this does not necessarily mean that the numerical procedure will not converge, since the Tandem Queue may actually have a finite, unknown mean service time.

Our partial exploration approach obtains system models that are clearly not irreducible. In fact, the trap state constitutes by itself a bottom strongly connected component. This does not mean that ergodicity is lost, but it cannot be guaranteed. However, for those cases where the estimation needed to be cut short because it had reached the limit of its allotted verification time, the iterative methods were clearly

monotonous increasing. Although this does not guarantee convergence, it has at least provided in every case a good argument for lower bounds on the desired reliability metrics.

7.4.2. Threats to internal validity

We also need to confront threats to *internal validity*. This is an issue in the analyses we performed, since the non-convergence of numerical methods constitutes a problem.

This non-convergence is a problem especially for the rewards estimations, as several of our experimentation runs on partial state spaces did not converge to a fixed value. In fact, those runs that provided our best bounds did not converge. Additionally, it is expected that, had the experiments run for further time, the obtained bounds would have increased.

However, we are confident in the contribution of our results for two reasons. First, whenever the verification failed to converge, the obtained bounds were much more larger than what could be obtained from the verification of the complete models. As a further confirmation of the reliability of the systems, we also verified bounded reachability properties over the system models, using a bound that is in the order of the obtained results. The results derived from these verifications were very good in terms of establishing reliability.

Adding to the confidence of the previous results, it is good news that, in the case of the bounded reachability properties convergence was not an issue. For each case study we obtained at least several good probability bounds on verification runs that *did* converge. Even though we witnessed some runs that did not converge, these were vastly outnumbered by those that converged.

CHAPTER 8

Discussion

In these last few chapters, we have presented a fully automated technique for estimation of probabilistic reachability properties and reward values of system models. Experimental results have shown that this approach may provide more useful estimations than both standard probabilistic model checking and Monte Carlo verification, at a fraction of the cost required by such techniques. We have also observed that these results are especially notorious when the properties under analysis are probabilistically rare. We believe that these results can be explained by the fact that the simulation traces capture a significant part of the most probable behaviour of the system model. Additionally, since we choose to characterise this partial state spaces through the use of invariants, it is likely that states similar to those visited during simulation are captured. This results in several behavioural loops to be present in these submodels. These loops capture a greater probability mass, while at the same time not increasing the state space size in a significant way.

However, some parameters exist that need to be set for the approach to work. First, there is the matter of the size of the simulation set and the length of the simulated traces; and second, in the case where non-determinism is present in the model under analysis, a strategy is necessary for solving these non-deterministic choices during the simulation phase.

Regarding the size of the simulation set and its traces, good news is that our experimentation has shown that, at least for the examples studied, very good results can be obtained through a relatively small set of short traces. Results have shown that there may be a broad combination of parameter values for which high estimation results are obtained in reasonable time. Further, overshooting these parameters does not have a dramatic impact in the resulting submodel size, so erring in the side of caution and choosing larger parameters does not seem to be a cause for concern. It is important to note that exploration of an appropriate parameter space can be done concurrently, taking as the final reward estimation the highest of the bounds obtained. Full model probabilistic checking cannot exploit concurrent computation in such a way. Monte Carlo verification can be applied concurrently, however we believe that the significant time cost for sample generation would not be outweighed by concurrent execution; further experimentation is needed to address this point.

As was previously mentioned, most probabilistic model checkers [KKZ05, HKNP06, SVA05b, You05] provide functionality that may either reduce the time required to obtain results, or reduce the memory footprint required for verification, such as sym-

metry reductions [KNP06], *lumping* [DG97] and several numerical methods. All these optimizations are orthogonal to the model checking procedure itself. Our work relies on probabilistic model checking and the experiments were run on PRISM, which implements some of these optimizations. In this way, our technique complements state-space reduction approaches. A related approach is that of [ZVB11, CBvB12], which aim at providing a measure of how much a model checker progressed towards the verification of a property. The results of such an approach however, lack an understandable link between the progress measure and the property that is being verified.

In those settings where exhaustive probabilistic model checking of models is intractable due to required memory size or verification time, statistical simulation has proven to be an effective technique. As was mentioned in the previous Chapter, an important issue with simulation approaches is that they tend to work well mostly in the case that the specified properties are bounded in time, i.e. when these properties can be written in the form $\psi \mathcal{U}^{\leq T} \rho$ for a fixed T. This is so because estimation of the random variable X_ϕ by means of a sample of traces σ_i requires that the question of whether $M, \sigma_i \models \phi$ or not be answered in a definite way for each trace σ_i in the sample set. If the formula ϕ is temporally bounded, then termination is guaranteed when evaluating its truth for the traces, but for temporally unbounded formulae such termination is threatened.

In such cases, generating traces within acceptable length bounds that answer the property definitively can be very unlikely. To address this problem *biased sampling* [SVA05a, RP09, LP06, BGH09] has been studied. However, bias to sampling must be done manually resulting in an impact on the analysis results that cannot be quantified in general. The result obtained by our approach is guaranteed to be a true bound to the reward values being sought after.

Related work by Younes et. al. [YCZ11] proposes two novel Monte Carlo approaches that do not rely on biased sampling. However, one of them may require an inordinate number of samples to produce results; while the other relies on reachability analysis, which requires the full model to be constructed, relinquishing one of the key advantages of Monte Carlo model checking over probabilistic model checking. The work in [HJB+10] also presents a bounded statistical approach for checking unbounded properties that does not need the full model to be constructed. However, the bound on the necessary trace length is excessively large, as traces may be as long as the total number of states in the model. Other works [KJD02] acknowledge the problem of generating traces exhibiting the failure (or guaranteeing its absence). This approach relies on extreme value theory to produce results. Unfortunately, extreme values techniques still require a good number of actual samples exhibiting the property, as these techniques require the inference of a fitting distribution. Having too few samples to work with usually results in fitting distributions that are actually different than the one being analysed [Col01].

As noted, an additional point for analysis lies in the strategies for resolving non-deterministic choices during simulation. Several works have attempted to solve this problem, especially in the context of generating simulations for Monte Carlo estimation. In these cases, it is critical that the simulation of non-deterministic transitions is performed in such a way that there is no bias in the generation (or alternatively, in such a way that this bias can be controlled and quantified), as doing so otherwise would introduce errors in the final estimation. In [HMZ+12] the authors leverage on the fact that, usually, verification is performed while looking for the worst and best cases. In that sense, only the two schedulers that induce the best and worst results are of interest, and the authors propose a self-adjusting simulation algorithm

that converges to these extremes.

In [BFFHH11], rather than focusing on the problem of biasing scheduler selection, the authors aim at detecting whether non-determinism can be ignored safely. As the authors point out, it is often the case that non-deterministic choices are actually behaviour-equivalent. By detecting these situations via partial order methods, it can be used to identify situations where non-determinism can be ignored while keeping only one of the possible choices when performing simulation.

In our present work, we have opted to resolve non-determinism by simply assuming an equiprobable distribution over the possible non-deterministic choices at a given state. However, it must be noted that, in the context of our work, *any* method of resolving non-determinism would have been acceptable, as we always produce a lower bound to the actual reward value, regardless of the procedure used for simulation. This is not to say that any non-determinism resolution method will produce the same outcome, as variations in these choices may lead to different invariants. Although the results presented in this thesis are promising, it still remains to be seen if different approaches to the initial simulation might produce even better results. In particular, the choice of simulating via equiprobable distribution of non-deterministic transitions is a double-edged sword. On the one hand, by establishing a balanced choice, it maximises the chance of exploring most of the non-deterministic alternatives so that verification of all of them is carried out at a later step. But, on the other hand, some of this explored behaviour might possibly be irrelevant when calculating the maximum (or minimum) rewards, as the best and/or worst schedulers might never take some of the explored non-deterministic transitions. In this sense, adapting the approach of [HMZ+12] to the simulation step of our framework might prove to be beneficial. Although that proposed approach is geared towards model checking of probabilistic properties rather than reward calculations, it may be adapted to our needs. It is worth noting, however, that such an approach would need to carry out two simulation steps as opposed to one. This is because the approach in [HMZ+12] aims at simulating executions that resemble those of the extreme scheduler that is of interest, which may be either the one providing the minimum value, or the maximum, but not both at the same time. In that sense, if we are interested in calculating both extreme values, we would need different simulation sets, one for each extreme.

The analysis of system behaviour that exhibits *rare* yet relevant events (e.g. failures) is the subject of focused study within the simulation community as well. A technique that is usually used in conjunction with stochastic processes that have rare events is that of *importance sampling* [RK08]. Roughly speaking, the idea of importance sampling is to replace the original process's distribution for another more likely to generate the (originally) rare event during the sample generation. The distribution replacement is chosen so that results from analyses for the new distribution can be translated back to results valid for the original distribution. Although this is a promising approach, finding suitable replacement distributions is a complex and ad-hoc task for which further research and expertise is necessary, as different system models possibly require different sampling distributions. Further, special care is required when proposing importance sampling distributions. In fact, it is possible to choose a replacement distribution such that it makes the simulation process *more costly* and requiring even more samples than the original one. In practice, choosing optimal replacement distributions is extremely difficult and not suitable for a general, complex process model. In this context of Importance Sampling, the work in [MSW12, MSW13] is closely related to ours. In this work, the authors use Importance Sampling in order to drive the exploration of the model by truncating explorations that do not contribute to the rare event probability, sharing our idea of

partial model exploration. However, the reliance on Importance Sampling requires an external understanding of the model. In contrast, our technique is agnostic with respect to the system being modelled.

Another promising simulation technique that also focuses on rare events is that of *sample splitting* [RK08, RC05], most notably the RESTART implementation [VAVA94] which, roughly, rather than starting each simulation from the initial state, it does so from a state s visited in a previous simulation and from which reaching a rare event is more likely. The likelihood of reaching state s from the initial state is taken into account for producing the final analysis results. Key to the application of these techniques is making appropriate decisions on where to restart simulations. These decisions demand deep understanding of both the model and the underlying splitting technique, as naïve splitting may not help the verification effort. Worse, it could even hamper the effort if the splits are not done in such a way that they are incrementally closer to fulfilling the rare event. Another interesting approach is that of [RdBSH13], which is geared toward simulating rare events, although restricted to Stochastic Petri Nets.

Finally, common to both the Monte Carlo approach and the simulation techniques discussed is the fact that they are inherently statistical results. As such, there is always a non-zero probability that the results obtained are completely off the mark. Further reducing this error probability may require excessive amount of additional traces to be sampled in order to obtain the guarantee. Our technique, though conservative in the bounds it obtains, is definitive in its answers.

The technique we introduce in this thesis is concerned with the verification of systems that are specified through the use of automata-like languages. We believe our approach can be extended in order to analyse source code as well. In this regard, there have been promising advances similar to our work. For example in [FPV13, BFd+14], symbolic execution is used to analyse the source code, and that information is used to direct a sampling approach towards interesting portions of the source code. The setting for this work is different and complementary, though, as it focuses on non-reactive, non-probabilistic software (by quantifying the usage profile of *program variables*); and the inference of conditions for reaching a given portion of the code. Further, this approach requires the solution space to be built and available for analysis; we argue that this, in our setting, is prohibitive in size.

On a related note, [LPD+14] has tackled the problem of synthesising appropriate schedulers for attaining a desired probability, a goal that is closely related to finding the extrema probabilities in the presence of non-determinism. Approaches such as this could benefit our technique by resolving non-determinism in a way that later directs verification to the more extreme (and interesting) values.

8.1. Conclusions and Further Work

In this Part of the thesis we have proposed an approach to estimating mean reward values and reachability probabilities for probabilistic system models. The approach is a novel combination of simulation, invariant inference and probabilistic model checking. We report on experiments that suggest that reward estimation using this technique can be more effective than (full model) probabilistic and statistical model checking for system models. This increase in effectiveness is most evident in the case of models where the properties under analysis are rare events, or else are unbounded in time. In addition, our estimation approach also supports non-determinism besides probabilistic behaviour.

We believe the notion of reliability analysis over partial yet systematic explo-

rations offers an alternative to, and hence complements, exhaustive model exploration–as in probabilistic model checking–and partial random exploration–as in statistical model checking.

The experimental results presented in the previous Chapter are promising. Our experiments show that, for system models extracted from reliability and probabilistic verification literature, bounds to probabilities and reward values can be obtained with little effort compared to full model verification. More specifically, we have shown that we can obtain reliability values that allow for strong dependability arguments, while only performing an exploration of typically less than 5% of the projected total state space of the system. These savings also translate into verification time as well, and the additional effort required for inferring submodels remains a good trade-off taking into account the quality of the obtained results.

The obtained results are more striking when the behaviours under analysis are rare events, and they have not been witnessed in the (already small) submodel being explored. However, experiments have also shown that our technique is effective even in the case of systems where the behaviour of interest is not rare, and even when some of the states exhibiting this behaviour are present in the obtained submodels. This evidence provides encouragement towards arguing for generalisation of results.

We also believe that further experimentation is required to achieve a better understanding of the influence of parameter choices in the process. In particular, an area that calls for future work is looking for a better understanding of the relationship between the simulated set of traces (both its size as the trace length) and the submodels that result from them, as well as the estimations that can be expected from them. This understanding should lead to heuristics for setting appropriate values to these parameters in order to achieve more cost-effective submodels.

CHAPTER 9

Conclusions and lookout

The main contributions and conclusions of this thesis have, in some way or another, already been discussed in Chapters 5 and 8. We will recall them here, as well as introduce research lines that can be derived using this thesis as a basis.

In this work, we tackled in the first place the problem of qualitative property verification, which is more often than not threatened by the state explosion problem. We propose a technique that can obtain partial, quantitative information related to the property under verification. This technique involves a careful modelling of the interfacing components, especially the operating profile of the environment with which the system interacts. This modelling can aid in quantifying partial explorations, which can then be thoroughly analysed. In summary, although we cannot provide a definite answer to the original qualitative question, we can provide bounds on related quantitative questions.

If the original property of interest is a safety predicate, expressed as "is it true that the following, failing, state is not reachable?", and this question cannot be resolved in a timely manner by a classic model checking approach, our proposed technique can establish several quantitative bounds on the property. Obtaining information such as the minimum operational time that the system will run before reaching the failure state can be useful when arguing a reliability case, or as additional information when weighing deployment risks. Similarly, it is also possible to bound the maximum probability of the failure manifesting itself. This also adds to a reliability case if this maximum probability is low enough to be accepted by the stakeholders.

In this thesis we have also attempted to characterise those systems for which our proposed technique consistently outperforms established approaches such as probabilistic model checking and statistical (Monte Carlo) approaches. Through our case studies we have evidence that, for systems where these failures are rare events and their state spaces are large and complex, our technique can offer more information, given the same time and memory budget, than both probabilistic model checkers and statistical verification approaches.

Non-determinism remains an important challenge to our technique, especially in the cases that this causes the behaviour of the system under analysis to be uniform. In these situations, the simulation approach fails to identify a portion of the state space that is both small enough to be manageable, and probabilistically dense enough to offer valuable information. We believe there is work to do in this aspect, especially on the topic of simulation strategies for non-deterministic systems. Specifically, sim-

ulation strategies that aim at mimicking the behaviour of the scheduler that makes the interesting states more likely are of special interest, as are those that make these interesting states be the least likely. This is closely related to the fact that, when performing quantitative validation of systems, we are most often interested in the extreme values of the failure probabilities or times to failure.

An interesting extension of this work would be its applicability to software deliverables closer to the end product. We envision our approach could be used in the context of software model checkers, such as Java Pathfinder [HP00]. Challenges toward this goal are the increased memory requirements which come as a result of the modelled states being much more fine grained than in our abstract case. This goal is also closely related to the previous simulation one, since achieving a Java Pathfinder implementation would also require simulation optimisations to be added to the JPF virtual machine.

Our work is also underpinned by the formalisms used to model probabilistic behaviour. We have presented Probabilistic Interface Automata as a suitable formalism that allows for compositional behaviour modelling and validation. We also relaxed some of the modelling restrictions posed by the underlying Interface Automata formalism, introducing fairness conditions to allow for delayed synchronisations. In the context of this work, we imposed strong fairness conditions that can be overly restrictive. More study is required in this sense, as there could exist fairness conditions that are not as stringent but that still enforce delayed synchronisation. Open questions are whether finding these fairness conditions requires a deep understanding of the model under analysis, or if these conditions can be established in general.

We have presented experimental evidence that allows us to argue that our approach can obtain useful results in the cases where a full verification effort is infeasible. However, there are some limitations to this approach, which we aim to tackle in a near future. First, the technique is dependent on a simulation step for which there are some parameters that must be set. Namely, the number of simulation traces and the length of these traces are crucial. Different combinations of these parameters can yield very diverse submodels. The results obtained through these submodels, in turn, can vary in their usefulness. The problem is that it can be difficult to ascertain which parameter combination will yield submodels that will perform well in their estimation. This topic requires further study.

An additional concern is that of the invariants that we aim to obtain. In this work, we have restricted ourselves to a very simple and small class of invariants. In particular, all invariants consist of the conjunction of arithmetical comparisons of variables and constants. Although these invariants have performed well in general, the question remains whether more complex invariants can help in those cases where our approach struggled, such as the Virus Infection case study. More study is required on other classes of invariants, even including temporal rather than static invariants.

Finally, we have argued that automatic approaches are more desirable than approaches that require manual intervention. However, we have not validated these claims in a controlled, user populated environment. There is a need for the design of a user experience experiment, that could answer whether users would be more comfortable sacrificing understanding (since the submodels and the invariants that obtain them are not necessarily intuitive) in exchange for a potentially more efficient partial model exploration.

APPENDIX A

Additional tables

Traces	Length	States	Invariant
1000	1000	10662	$cliC \leq 65 \wedge cliM \leq 14 \wedge state \leq 9$
2000	1000	14158	$cliC \leq 77 \wedge cliM \leq 16 \wedge state \leq 9$
3000	1000	16334	$cliC \leq 80 \wedge cliM \leq 18 \wedge state \leq 9$
4000	1000	15990	$cliC \leq 71 \wedge cliM \leq 20 \wedge state \leq 9$
5000	1000	14134	$cliC \leq 69 \wedge cliM \leq 18 \wedge state \leq 9$
6000	1000	14698	$cliC \leq 80 \wedge cliM \leq 16 \wedge state \leq 9$
7000	1000	20334	$cliC \leq 100 \wedge cliM \leq 18 \wedge state \leq 9$
8000	1000	15134	$cliC \leq 74 \wedge cliM \leq 18 \wedge state \leq 9$
9000	1000	17446	$cliC \leq 71 \wedge cliM \leq 22 \wedge state \leq 9$
10000	1000	16086	$cliC \leq 83 \wedge cliM \leq 17 \wedge state \leq 9$
1000	2000	18734	$cliC \leq 92 \wedge cliM \leq 18 \wedge state \leq 9$
2000	2000	13370	$cliC \leq 77 \wedge cliM \leq 15 \wedge state \leq 9$
3000	2000	17970	$cliC \leq 80 \wedge cliM \leq 20 \wedge state \leq 9$
4000	2000	21270	$cliC \leq 95 \wedge cliM \leq 20 \wedge state \leq 9$
5000	2000	23388	$cliC \leq 100 \wedge cliM \leq 21 \wedge state \leq 9$
6000	2000	27924	$cliC \leq 110 \wedge cliM \leq 23 \wedge state \leq 9$
7000	2000	19730	$cliC \leq 88 \wedge cliM \leq 20 \wedge state \leq 9$
8000	2000	24886	$cliC \leq 102 \wedge cliM \leq 22 \wedge state \leq 9$
9000	2000	21050	$cliC \leq 94 \wedge cliM \leq 20 \wedge state \leq 9$
10000	2000	22486	$cliC \leq 92 \wedge cliM \leq 22 \wedge state \leq 9$
1000	3000	18788	$cliC \leq 80 \wedge cliM \leq 21 \wedge state \leq 9$
2000	3000	19708	$cliC \leq 84 \wedge cliM \leq 21 \wedge state \leq 9$
3000	3000	22150	$cliC \leq 99 \wedge cliM \leq 20 \wedge state \leq 9$
4000	3000	20858	$cliC \leq 89 \wedge cliM \leq 21 \wedge state \leq 9$
5000	3000	20932	$cliC \leq 98 \wedge cliM \leq 19 \wedge state \leq 9$
6000	3000	23206	$cliC \leq 95 \wedge cliM \leq 22 \wedge state \leq 9$
7000	3000	22928	$cliC \leq 98 \wedge cliM \leq 21 \wedge state \leq 9$
8000	3000	21050	$cliC \leq 94 \wedge cliM \leq 20 \wedge state \leq 9$
9000	3000	28432	$cliC \leq 90 \wedge cliM \leq 29 \wedge state \leq 9$
10000	3000	25228	$cliC \leq 108 \wedge cliM \leq 21 \wedge state \leq 9$
1000	4000	20134	$cliC \leq 99 \wedge cliM \leq 18 \wedge state \leq 9$
2000	4000	17992	$cliC \leq 84 \wedge cliM \leq 19 \wedge state \leq 9$
3000	4000	20168	$cliC \leq 86 \wedge cliM \leq 21 \wedge state \leq 9$
4000	4000	18334	$cliC \leq 90 \wedge cliM \leq 18 \wedge state \leq 9$
5000	4000	24538	$cliC \leq 105 \wedge cliM \leq 21 \wedge state \leq 9$
6000	4000	23388	$cliC \leq 100 \wedge cliM \leq 21 \wedge state \leq 9$
7000	4000	22370	$cliC \leq 100 \wedge cliM \leq 20 \wedge state \leq 9$
8000	4000	29038	$cliC \leq 102 \wedge cliM \leq 26 \wedge state \leq 9$
9000	4000	23910	$cliC \leq 107 \wedge cliM \leq 20 \wedge state \leq 9$
10000	4000	24882	$cliC \leq 94 \wedge cliM \leq 24 \wedge state \leq 9$

Table A.1: Tandem Queue system submodel sizes and invariants for different parameter configurations.

Traces	Length	States	Invariant
1000	5000	15418	$cliC \leq 84 \wedge cliM \leq 16 \wedge state \leq 9$
2000	5000	19730	$cliC \leq 88 \wedge cliM \leq 20 \wedge state \leq 9$
3000	5000	22486	$cliC \leq 92 \wedge cliM \leq 22 \wedge state \leq 9$
4000	5000	26990	$cliC \leq 121 \wedge cliM \leq 20 \wedge state \leq 9$
5000	5000	26424	$cliC \leq 104 \wedge cliM \leq 23 \wedge state \leq 9$
6000	5000	22006	$cliC \leq 90 \wedge cliM \leq 22 \wedge state \leq 9$
7000	5000	27174	$cliC \leq 107 \wedge cliM \leq 23 \wedge state \leq 9$
8000	5000	25424	$cliC \leq 100 \wedge cliM \leq 23 \wedge state \leq 9$
9000	5000	25958	$cliC \leq 91 \wedge cliM \leq 26 \wedge state \leq 9$
10000	5000	23686	$cliC \leq 97 \wedge cliM \leq 22 \wedge state \leq 9$
1000	6000	21088	$cliC \leq 90 \wedge cliM \leq 21 \wedge state \leq 9$
2000	6000	17134	$cliC \leq 84 \wedge cliM \leq 18 \wedge state \leq 9$
3000	6000	25402	$cliC \leq 96 \wedge cliM \leq 24 \wedge state \leq 9$
4000	6000	25142	$cliC \leq 95 \wedge cliM \leq 24 \wedge state \leq 9$
5000	6000	26182	$cliC \leq 99 \wedge cliM \leq 24 \wedge state \leq 9$
6000	6000	25366	$cliC \leq 104 \wedge cliM \leq 22 \wedge state \leq 9$
7000	6000	27190	$cliC \leq 99 \wedge cliM \leq 25 \wedge state \leq 9$
8000	6000	34174	$cliC \leq 135 \wedge cliM \leq 23 \wedge state \leq 9$
9000	6000	30430	$cliC \leq 111 \wedge cliM \leq 25 \wedge state \leq 9$
10000	6000	31902	$cliC \leq 121 \wedge cliM \leq 24 \wedge state \leq 9$
1000	7000	21088	$cliC \leq 90 \wedge cliM \leq 21 \wedge state \leq 9$
2000	7000	22008	$cliC \leq 94 \wedge cliM \leq 21 \wedge state \leq 9$
3000	7000	25674	$cliC \leq 101 \wedge cliM \leq 23 \wedge state \leq 9$
4000	7000	23848	$cliC \leq 102 \wedge cliM \leq 21 \wedge state \leq 9$
5000	7000	29926	$cliC \leq 123 \wedge cliM \leq 22 \wedge state \leq 9$
6000	7000	27742	$cliC \leq 105 \wedge cliM \leq 24 \wedge state \leq 9$
7000	7000	27924	$cliC \leq 110 \wedge cliM \leq 23 \wedge state \leq 9$
8000	7000	28262	$cliC \leq 107 \wedge cliM \leq 24 \wedge state \leq 9$
9000	7000	26182	$cliC \leq 99 \wedge cliM \leq 24 \wedge state \leq 9$
10000	7000	30674	$cliC \leq 121 \wedge cliM \leq 23 \wedge state \leq 9$
1000	8000	24886	$cliC \leq 102 \wedge cliM \leq 22 \wedge state \leq 9$
2000	8000	28726	$cliC \leq 118 \wedge cliM \leq 22 \wedge state \leq 9$
3000	8000	23924	$cliC \leq 94 \wedge cliM \leq 23 \wedge state \leq 9$
4000	8000	25366	$cliC \leq 104 \wedge cliM \leq 22 \wedge state \leq 9$
5000	8000	23910	$cliC \leq 107 \wedge cliM \leq 20 \wedge state \leq 9$
6000	8000	26702	$cliC \leq 101 \wedge cliM \leq 24 \wedge state \leq 9$
7000	8000	28522	$cliC \leq 108 \wedge cliM \leq 24 \wedge state \leq 9$
8000	8000	31414	$cliC \leq 103 \wedge cliM \leq 28 \wedge state \leq 9$
9000	8000	30076	$cliC \leq 102 \wedge cliM \leq 27 \wedge state \leq 9$
10000	8000	29424	$cliC \leq 116 \wedge cliM \leq 23 \wedge state \leq 9$
1000	9000	22192	$cliC \leq 104 \wedge cliM \leq 19 \wedge state \leq 9$
2000	9000	20830	$cliC \leq 93 \wedge cliM \leq 20 \wedge state \leq 9$
3000	9000	23924	$cliC \leq 94 \wedge cliM \leq 23 \wedge state \leq 9$
4000	9000	27674	$cliC \leq 109 \wedge cliM \leq 23 \wedge state \leq 9$
5000	9000	29924	$cliC \leq 118 \wedge cliM \leq 23 \wedge state \leq 9$
6000	9000	27046	$cliC \leq 111 \wedge cliM \leq 22 \wedge state \leq 9$
7000	9000	31126	$cliC \leq 128 \wedge cliM \leq 22 \wedge state \leq 9$
8000	9000	30214	$cliC \leq 99 \wedge cliM \leq 28 \wedge state \leq 9$
9000	9000	32590	$cliC \leq 119 \wedge cliM \leq 25 \wedge state \leq 9$
10000	9000	29926	$cliC \leq 123 \wedge cliM \leq 22 \wedge state \leq 9$
1000	10000	20398	$cliC \leq 87 \wedge cliM \leq 21 \wedge state \leq 9$
2000	10000	23032	$cliC \leq 108 \wedge cliM \leq 19 \wedge state \leq 9$
3000	10000	26330	$cliC \leq 118 \wedge cliM \leq 20 \wedge state \leq 9$
4000	10000	25230	$cliC \leq 113 \wedge cliM \leq 20 \wedge state \leq 9$
5000	10000	27174	$cliC \leq 107 \wedge cliM \leq 23 \wedge state \leq 9$
6000	10000	33214	$cliC \leq 109 \wedge cliM \leq 28 \wedge state \leq 9$
7000	10000	25402	$cliC \leq 96 \wedge cliM \leq 24 \wedge state \leq 9$
8000	10000	28522	$cliC \leq 108 \wedge cliM \leq 24 \wedge state \leq 9$
9000	10000	29562	$cliC \leq 112 \wedge cliM \leq 24 \wedge state \leq 9$
10000	10000	27460	$cliC \leq 100 \wedge cliM \leq 25 \wedge state \leq 9$

Table A.2: Tandem Queue system submodel sizes and invariants for different parameter configurations (cont.).

Traces	Length	States	Invariant
1000	1000	85304	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 1 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge nrtr \leq r \wedge nrtr \leq rrep \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge i \geq k \wedge r \geq l$
2000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	1000	85612	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 1 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
1000	2000	209228	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge nrtr \leq r \wedge nrtr \leq rrep \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge i \geq k \wedge r \geq l$
2000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
1000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	3000	196344	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 1 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge nrtr \leq r \wedge nrtr \leq rrep \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	3000	276133	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$

Table A.3: BRP system (fully probabilistic) submodel sizes and invariants for different parameter configurations.

Traces	Length	States	Invariant
1000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	4000	405252	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 3 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	4000	315191	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
1000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	5000	333099	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	5000	333099	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	5000	333099	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
1000	6000	347634	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge nrtr \leq r \wedge nrtr \leq rrep \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	6000	447706	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 3 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	6000	348129	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$

Table A.4: BRP system (fully probabilistic) submodel sizes and invariants for different parameter configurations (cont.).

Traces	Length	States	Invariant
1000	7000	362664	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge nrtr \leq r \wedge nrtr \leq rrep \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	7000	362664	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge nrtr \leq r \wedge nrtr \leq rrep \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	7000	362728	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1166 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
1000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	8000	486334	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 3 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
1000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	9000	393127	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	9000	505704	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 3 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	9000	505704	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 3 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$

Table A.5: BRP system (fully probabilistic) submodel sizes and invariants for different parameter configurations (cont.).

Traces	Length	States	Invariant
1000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	10000	505704	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 3 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	10000	505704	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 3 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$

Table A.6: BRP system (fully probabilistic) submodel sizes and invariants for different parameter configurations (cont.).

Traces	Length	States	Invariant
1000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	1000	85612	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 1 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	1000	120010	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 1 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
1000	2000	149352	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$

Table A.7: BRP system (non-deterministic) submodel sizes and invariants for different parameter configurations.

Traces	Length	States	Invariant
8000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	2000	209646	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
1000	3000	196080	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 1 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge nrtr \leq r \wedge nrtr \leq rrep \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge i \geq k \wedge r \geq l$
2000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	3000	354910	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 3 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	3000	275792	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
1000	4000	314432	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge nrtr \leq r \wedge nrtr \leq rrep \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge i \geq k \wedge r \geq l$
2000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	4000	314850	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	4000	315191	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	4000	315191	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 667 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
1000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	5000	428334	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 3 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$

Table A.8: BRP system (non-deterministic) submodel sizes and invariants for different parameter configurations (cont.).

Traces	Length	States	Invariant
8000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	5000	332758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	5000	428334	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 3 \wedge fileSize \leq 1500 \wedge i \leq 833 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
1000	6000	347634	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge nrtr \leq r \wedge nrtr \leq rrep \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	6000	347788	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1000 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
1000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	7000	362818	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	7000	363159	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1167 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
1000	8000	377604	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge nrtr \leq r \wedge nrtr \leq rrep \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	8000	486334	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 3 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	8000	377758	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1333 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$

Table A.9: BRP system (non-deterministic) submodel sizes and invariants for different parameter configurations (cont.).

Traces	Length	States	Invariant
1000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	9000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
1000	10000	392632	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge nrtr \leq r \wedge nrtr \leq rrep \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
2000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
3000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
4000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
5000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
6000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
7000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
8000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
9000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$
10000	10000	392786	$s \leq 7 \wedge srep \leq 3 \wedge nrtr \leq 2 \wedge fileSize \leq 1500 \wedge i \leq 1500 \wedge r \leq 4 \wedge rrep \leq 3 \wedge k \leq 2 \wedge l \leq 2 \wedge s \geq k \wedge s \geq l \wedge srep \leq fileSize \wedge srep \leq i \wedge srep \leq r \wedge srep \leq rrep \wedge nrtr \leq fileSize \wedge nrtr \leq i \wedge fileSize \geq r \wedge fileSize \geq rrep \wedge fileSize \geq k \wedge fileSize \geq l \wedge r \geq l$

Table A.10: BRP system (non-deterministic) submodel sizes and invariants for different parameter configurations (cont.).

Traces	Length	States	Invariant
1000	1000	118252	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 2 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 2 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 < s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
2000	1000	118272	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 2 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 2 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
3000	1000	470448	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
4000	1000	468424	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
5000	1000	463087	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$

Table A.11: WLAN system submodel sizes and invariants for different parameter configurations.

Traces	Length	States	Invariant
6000	1000	124134	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
7000	1000	463087	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
8000	1000	118105	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
9000	1000	470448	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
10000	1000	108964	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
1000	2000	118232	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 2 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 2 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
2000	2000	122650	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
3000	2000	126310	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
4000	2000	125575	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
5000	2000	127016	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
6000	2000	471284	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
7000	2000	127610	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
8000	2000	119096	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
9000	2000	464017	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
10000	2000	468306	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
1000	3000	118232	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 2 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 2 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 < s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
2000	3000	126907	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
3000	3000	126010	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
4000	3000	472595	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
5000	3000	468306	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
6000	3000	125575	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
7000	3000	119096	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
8000	3000	455726	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$

Table A.12: WLAN system submodel sizes and invariants for different parameter configurations (cont.).

Traces	Length	States	Invariant
9000	3000	468306	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
10000	3000	118393	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
1000	4000	118252	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 2 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 2 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
2000	4000	126907	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
3000	4000	127610	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
4000	4000	128157	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
5000	4000	119096	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
6000	4000	473779	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 4 \wedge backoff1 \leq 31 \wedge bc1 \leq 4 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 5 \wedge backoff2 \leq 31 \wedge bc2 \leq 4 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 \geq slot1 \wedge s1 \geq bc1 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge s2 \geq bc2$
7000	4000	470448	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
8000	4000	464017	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
9000	4000	508342	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 4 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 4 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge s2 > slot2 \wedge slot2 \leq bc2$
10000	4000	116549	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
1000	5000	137248	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 < s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
2000	5000	125434	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
3000	5000	118272	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 2 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 2 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
4000	5000	126907	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
5000	5000	116549	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
6000	5000	463087	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
7000	5000	467376	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
8000	5000	463087	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
9000	5000	125434	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
10000	5000	464017	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
1000	6000	127610	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
2000	6000	126010	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$

Table A.13: WLAN system submodel sizes and invariants for different parameter configurations (cont.).

Traces	Length	States	Invariant
3000	6000	126010	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
4000	6000	463087	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
5000	6000	108964	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
6000	6000	467376	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
7000	6000	467494	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
8000	6000	119096	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
9000	6000	119096	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
10000	6000	468306	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
1000	7000	118272	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 2 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 2 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
2000	7000	131553	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
3000	7000	128380	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
4000	7000	117284	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
5000	7000	127195	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
6000	7000	116549	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
7000	7000	126010	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
8000	7000	455726	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
9000	7000	118105	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
10000	7000	118393	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
1000	8000	118124	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 2 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 2 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 < s2 \wedge bc1 \geq slot2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
2000	8000	118232	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 2 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 2 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 < s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
3000	8000	119096	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
4000	8000	126010	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
5000	8000	463087	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
6000	8000	464017	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$

Table A.14: WLAN system submodel sizes and invariants for different parameter configurations (cont.).

Traces	Length	States	Invariant
7000	8000	463087	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
8000	8000	464017	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
9000	8000	127610	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
10000	8000	471284	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
1000	9000	118232	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 2 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 2 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 > bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 < s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
2000	9000	118393	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
3000	9000	464017	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
4000	9000	118393	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
5000	9000	126363	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
6000	9000	470662	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
7000	9000	455726	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
8000	9000	118393	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
9000	9000	119096	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
10000	9000	127195	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
1000	10000	126010	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
2000	10000	130818	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 < s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
3000	10000	118393	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
4000	10000	463087	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
5000	10000	126010	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 1 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
6000	10000	125575	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 2 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 1 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \geq slot1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 > bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 < s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
7000	10000	119096	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 > bc2 \wedge slot2 \leq bc2$
8000	10000	455726	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 \geq slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
9000	10000	464017	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 3 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$
10000	10000	463087	$col \leq 2 \wedge c1 \leq 2 \wedge c2 \leq 2 \wedge x1 \leq 10 \wedge s1 \leq 12 \wedge s1 \geq 1 \wedge slot1 \leq 3 \wedge backoff1 \leq 31 \wedge bc1 \leq 3 \wedge x2 \leq 10 \wedge s2 \leq 12 \wedge s2 \geq 1 \wedge slot2 \leq 2 \wedge backoff2 \leq 31 \wedge bc2 \leq 3 \wedge col \leq s1 \wedge col \leq s2 \wedge col \geq slot2 \wedge c1 < s1 \wedge c1 \leq s2 \wedge c2 \leq s1 \wedge c2 < s2 \wedge s1 > slot1 \wedge s1 \geq bc1 \wedge s1 > slot2 \wedge s1 \geq bc2 \wedge slot1 \leq bc1 \wedge slot1 \leq s2 \wedge bc1 \leq s2 \wedge s2 > slot2 \wedge s2 \geq bc2 \wedge slot2 \leq bc2$

Table A.15: WLAN system submodel sizes and invariants for different parameter configurations (cont.).

Traces	Length	States	Invariant
1000	100	7728	$s123 = s132 \wedge s123 = s133 \wedge s123 = s213 \wedge s123 = s222 \wedge s123 = s223 \wedge s123 = s231 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s312 \wedge s123 = s313 \wedge s123 = s321 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 1 \wedge s121 \leq 2 \wedge s122 \leq 1 \wedge s123 \leq 0 \wedge s131 \leq 1 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s221 \leq 1 \wedge s311 \leq 1 \wedge s111 \geq s112 \wedge s111 > s113 \wedge s111 \geq s121 \wedge s111 > s122 \wedge s111 > s123 \wedge true \wedge s111 > s131 \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s221 \wedge s111 > s311 \wedge s112 \geq s113 \wedge s112 \geq s122 \wedge s112 \geq s123 \wedge s113 \geq s123 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s122 \geq s123 \wedge s123 \leq s131 \wedge s123 \leq s211 \wedge s123 \leq s212 \wedge s123 \leq s221 \wedge s123 \leq s311 \wedge s211 \geq s311$
2000	100	10790	$s123 = s132 \wedge s123 = s133 \wedge s123 = s213 \wedge s123 = s223 \wedge s123 = s231 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s312 \wedge s123 = s313 \wedge s123 = s321 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 1 \wedge s121 \leq 2 \wedge s122 \leq 1 \wedge s123 \leq 0 \wedge s131 \leq 1 \wedge s211 \leq 2 \wedge s212 \leq 1 \wedge s221 \leq 2 \wedge s222 \leq 1 \wedge s311 \leq 1 \wedge s111 \geq s112 \wedge s111 > s113 \wedge s111 \geq s121 \wedge s111 > s122 \wedge s111 > s123 \wedge true \wedge s111 > s131 \wedge s111 \geq s211 \wedge s111 > s212 \wedge s111 \geq s221 \wedge s111 > s222 \wedge s111 > s311 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s113 \geq s123 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s121 \geq s222 \wedge s121 \geq s311 \wedge s122 \geq s123 \wedge s123 \leq s131 \wedge s123 \leq s211 \wedge s123 \leq s212 \wedge s123 \leq s221 \wedge s123 \leq s222 \wedge s123 \leq s311 \wedge s211 \geq s311 \wedge s221 \geq s222$
3000	100	12992	$s123 = s132 \wedge s123 = s133 \wedge s123 = s213 \wedge s123 = s222 \wedge s123 = s223 \wedge s123 = s231 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s312 \wedge s123 = s313 \wedge s123 = s321 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 1 \wedge s121 \leq 2 \wedge s122 \leq 1 \wedge s123 \leq 0 \wedge s131 \leq 1 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s221 \leq 2 \wedge s311 \leq 1 \wedge s111 \geq s112 \wedge s111 > s113 \wedge s111 \geq s121 \wedge s111 > s122 \wedge s111 > s123 \wedge true \wedge s111 > s131 \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s221 \wedge s111 > s311 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s113 \geq s123 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s122 \geq s123 \wedge s123 \leq s131 \wedge s123 \leq s211 \wedge s123 \leq s212 \wedge s123 \leq s221 \wedge s123 \leq s311 \wedge s211 \geq s311$
4000	100	35031	$s123 = s132 \wedge s123 = s133 \wedge s123 = s213 \wedge s123 = s222 \wedge s123 = s223 \wedge s123 = s231 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s312 \wedge s123 = s313 \wedge s123 = s321 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 0 \wedge s131 \leq 1 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s221 \leq 2 \wedge s311 \leq 2 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge true \wedge s111 > s131 \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s221 \wedge s111 \geq s311 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s113 \geq s123 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s122 \geq s123 \wedge s123 \leq s131 \wedge s123 \leq s211 \wedge s123 \leq s212 \wedge s123 \leq s221 \wedge s123 \leq s311 \wedge s211 \geq s311$
5000	100	17378	$s123 = s132 \wedge s123 = s133 \wedge s123 = s213 \wedge s123 = s222 \wedge s123 = s223 \wedge s123 = s231 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s312 \wedge s123 = s313 \wedge s123 = s321 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 1 \wedge s123 \leq 0 \wedge s131 \leq 1 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s221 \leq 2 \wedge s311 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 > s122 \wedge s111 > s123 \wedge true \wedge s111 > s131 \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s221 \wedge s111 > s311 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s113 \geq s123 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s122 \geq s123 \wedge s123 \leq s131 \wedge s123 \leq s211 \wedge s123 \leq s212 \wedge s123 \leq s221 \wedge s123 \leq s311 \wedge s211 \geq s311$
6000	100	14760	$s123 = s132 \wedge s123 = s133 \wedge s123 = s213 \wedge s123 = s222 \wedge s123 = s223 \wedge s123 = s231 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s312 \wedge s123 = s313 \wedge s123 = s321 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 0 \wedge s131 \leq 1 \wedge s211 \leq 2 \wedge s212 \leq 1 \wedge s221 \leq 1 \wedge s311 \leq 2 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge true \wedge s111 > s131 \wedge s111 \geq s211 \wedge s111 > s212 \wedge s111 > s221 \wedge s111 \geq s311 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s113 \geq s123 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s122 \geq s123 \wedge s123 \leq s131 \wedge s123 \leq s211 \wedge s123 \leq s212 \wedge s123 \leq s221 \wedge s123 \leq s311 \wedge s211 \geq s311$
7000	100	52442	$s123 = s132 \wedge s123 = s133 \wedge s123 = s213 \wedge s123 = s223 \wedge s123 = s231 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s312 \wedge s123 = s313 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 0 \wedge s131 \leq 1 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s221 \leq 2 \wedge s222 \leq 1 \wedge s311 \leq 2 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge true \wedge s111 > s131 \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 \geq s311 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s113 \geq s123 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s122 \geq s123 \wedge s123 \leq s131 \wedge s123 \leq s211 \wedge s123 \leq s212 \wedge s123 \leq s221 \wedge s123 \leq s222 \wedge s123 \leq s311 \wedge s211 \geq s222 \wedge s211 \geq s311 \wedge s221 \geq s222$
8000	100	35031	$s123 = s132 \wedge s123 = s133 \wedge s123 = s213 \wedge s123 = s222 \wedge s123 = s223 \wedge s123 = s231 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s312 \wedge s123 = s313 \wedge s123 = s321 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 0 \wedge s131 \leq 2 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s221 \leq 2 \wedge s311 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge true \wedge s111 \geq s131 \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s221 \wedge s111 > s311 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s113 \geq s123 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s122 \geq s123 \wedge s123 \leq s131 \wedge s123 \leq s211 \wedge s123 \leq s212 \wedge s123 \leq s221 \wedge s123 \leq s311 \wedge s211 \geq s311$
9000	100	45641	$s123 = s132 \wedge s123 = s133 \wedge s123 = s213 \wedge s123 = s222 \wedge s123 = s223 \wedge s123 = s231 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s312 \wedge s123 = s313 \wedge s123 = s321 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 0 \wedge s131 \leq 2 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s221 \leq 2 \wedge s311 \leq 2 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge true \wedge s111 \geq s131 \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s221 \wedge s111 \geq s311 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s113 \geq s123 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s122 \geq s123 \wedge s123 \leq s131 \wedge s123 \leq s211 \wedge s123 \leq s212 \wedge s123 \leq s221 \wedge s123 \leq s311 \wedge s211 \geq s311$
10000	100	45641	$s123 = s132 \wedge s123 = s133 \wedge s123 = s213 \wedge s123 = s222 \wedge s123 = s223 \wedge s123 = s231 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s312 \wedge s123 = s313 \wedge s123 = s321 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 0 \wedge s131 \leq 2 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s221 \leq 2 \wedge s311 \leq 2 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge true \wedge s111 \geq s131 \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s221 \wedge s111 \geq s311 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s113 \geq s123 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s122 \geq s123 \wedge s123 \leq s131 \wedge s123 \leq s211 \wedge s123 \leq s212 \wedge s123 \leq s221 \wedge s123 \leq s311 \wedge s211 \geq s311$
1000	200	64878	$s123 = s132 \wedge s123 = s133 \wedge s123 = s213 \wedge s123 = s223 \wedge s123 = s231 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s312 \wedge s123 = s313 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 0 \wedge s131 \leq 2 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s221 \leq 2 \wedge s222 \leq 1 \wedge s311 \leq 2 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge true \wedge s111 \geq s131 \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s221 \wedge s111 > s222 \wedge s111 \geq s311 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s113 \geq s123 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s122 \geq s123 \wedge s123 \leq s131 \wedge s123 \leq s211 \wedge s123 \leq s212 \wedge s123 \leq s221 \wedge s123 \leq s222 \wedge s123 \leq s311 \wedge s211 \geq s311 \wedge s212 \geq s222$

Table A.16: Virus infection system submodel sizes and invariants for different parameter configurations.

Traces	Length	States	Invariant
2000	200	358163	$s123 = s133 \wedge s123 = s223 \wedge s123 = s231 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s313 \wedge s123 = s321 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 0 \wedge s131 \leq 2 \wedge s132 \leq 1 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 1 \wedge s221 \leq 2 \wedge s222 \leq 2 \wedge s311 \leq 2 \wedge s312 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge true \wedge s111 \geq s131 \wedge s111 > s132 \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 \geq s311 \wedge s111 > s312 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s112 \geq s213 \wedge s112 \geq s312 \wedge s113 \geq s123 \wedge s113 \geq s213 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s121 \geq s132 \wedge s122 \geq s123 \wedge s123 \leq s131 \wedge s123 \leq s132 \wedge s123 \leq s211 \wedge s123 \leq s212 \wedge s123 \leq s213 \wedge s123 \leq s221 \wedge s123 \leq s222 \wedge s123 \leq s311 \wedge s123 \leq s312 \wedge s131 \geq s132 \wedge s211 \geq s311 \wedge s212 \geq s312$
3000	200	182675	$s132 = s133 \wedge s132 = s213 \wedge s132 = s223 \wedge s132 = s232 \wedge s132 = s233 \wedge s132 = s312 \wedge s132 = s313 \wedge s132 = s322 \wedge s132 = s323 \wedge s132 = s331 \wedge s132 = s332 \wedge s132 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 1 \wedge s131 \leq 2 \wedge s132 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s221 \leq 2 \wedge s222 \leq 1 \wedge s231 \leq 1 \wedge s311 \leq 2 \wedge s321 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge s111 \geq s131 \wedge s111 > s132 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s221 \wedge s111 > s222 \wedge s111 > s231 \wedge s111 \geq s311 \wedge s111 > s321 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s112 \geq s132 \wedge s112 \geq s222 \wedge s113 \geq s132 \wedge s121 \geq s131 \wedge s121 \geq s132 \wedge s121 \geq s222 \wedge s122 \geq s123 \wedge s122 \geq s132 \wedge s123 \geq s132 \wedge s123 \leq s211 \wedge s131 \geq s132 \wedge s132 \leq s211 \wedge s132 \leq s212 \wedge s132 \leq s221 \wedge s132 \leq s222 \wedge s132 \leq s231 \wedge s132 \leq s311 \wedge s132 \leq s321 \wedge s211 \geq s231 \wedge s211 \geq s311 \wedge s211 \geq s321 \wedge s212 \geq s222 \wedge s221 \geq s231 \wedge s221 \geq s321$
4000	200	262023	$s133 = s222 \wedge s133 = s223 \wedge s133 = s232 \wedge s133 = s233 \wedge s133 = s313 \wedge s133 = s322 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 1 \wedge s131 \leq 2 \wedge s132 \leq 1 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 1 \wedge s221 \leq 2 \wedge s231 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 1 \wedge s321 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge s111 \geq s131 \wedge s111 > s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 > s221 \wedge s111 \geq s231 \wedge s111 > s311 \wedge s111 > s312 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s112 \geq s132 \wedge s112 \geq s133 \wedge s112 \geq s213 \wedge s113 \geq s133 \wedge s113 \geq s213 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s121 \geq s132 \wedge s121 \geq s133 \wedge s121 \geq s213 \wedge s122 \geq s123 \wedge s122 \geq s132 \wedge s122 \geq s133 \wedge s123 \leq s131 \wedge s123 \leq s133 \wedge s123 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s321 \wedge s211 \geq s213 \wedge s211 \geq s231 \wedge s211 \geq s311 \wedge s212 \geq s222 \wedge s221 \geq s231 \wedge s221 \geq s311 \wedge s311 \geq s321$
5000	200	157951	$s132 = s133 \wedge s132 = s213 \wedge s132 = s223 \wedge s132 = s231 \wedge s132 = s232 \wedge s132 = s233 \wedge s132 = s313 \wedge s132 = s322 \wedge s132 = s323 \wedge s132 = s331 \wedge s132 = s332 \wedge s132 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 1 \wedge s131 \leq 2 \wedge s132 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s221 \leq 2 \wedge s222 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 1 \wedge s321 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge s111 \geq s131 \wedge s111 > s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 > s221 \wedge s111 > s231 \wedge s111 \geq s311 \wedge s111 > s312 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s112 \geq s132 \wedge s112 \geq s133 \wedge s112 \geq s213 \wedge s113 \geq s133 \wedge s113 \geq s213 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s121 \geq s132 \wedge s121 \geq s133 \wedge s121 \geq s213 \wedge s122 \geq s123 \wedge s122 \geq s132 \wedge s122 \geq s133 \wedge s123 \leq s131 \wedge s123 \leq s133 \wedge s123 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s321 \wedge s211 \geq s213 \wedge s211 \geq s231 \wedge s211 \geq s311 \wedge s212 \geq s222 \wedge s221 \geq s231 \wedge s221 \geq s311 \wedge s311 \geq s321$
6000	200	702579	$s123 = s132 \wedge s123 = s133 \wedge s123 = s223 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s313 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 0 \wedge s131 \leq 2 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 1 \wedge s221 \leq 2 \wedge s222 \leq 2 \wedge s231 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 1 \wedge s321 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge true \wedge s111 \geq s131 \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 > s231 \wedge s111 \geq s311 \wedge s111 > s312 \wedge s111 > s321 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s112 \geq s131 \wedge s122 \geq s123 \wedge s123 \leq s131 \wedge s123 \leq s211 \wedge s123 \leq s212 \wedge s123 \leq s213 \wedge s123 \leq s221 \wedge s123 \leq s222 \wedge s123 \leq s231 \wedge s123 \leq s311 \wedge s123 \leq s312 \wedge s123 \leq s321 \wedge s211 \geq s311 \wedge s211 \geq s312 \wedge s212 \geq s312 \wedge s221 \geq s231$
7000	200	242870	$s133 = s223 \wedge s133 = s232 \wedge s133 = s233 \wedge s133 = s312 \wedge s133 = s313 \wedge s133 = s321 \wedge s133 = s322 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 1 \wedge s131 \leq 2 \wedge s132 \leq 1 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 1 \wedge s221 \leq 2 \wedge s222 \leq 1 \wedge s231 \leq 1 \wedge s311 \leq 2 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge s111 \geq s131 \wedge s111 > s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 > s231 \wedge s111 \geq s311 \wedge s111 > s312 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s112 \geq s132 \wedge s112 \geq s133 \wedge s112 \geq s213 \wedge s113 \geq s133 \wedge s121 \geq s131 \wedge s121 \geq s132 \wedge s121 \geq s133 \wedge s121 \geq s231 \wedge s122 \geq s132 \wedge s122 \geq s133 \wedge s123 \geq s133 \wedge s131 \geq s133 \wedge s131 \geq s231 \wedge s132 \geq s133 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s211 \geq s231 \wedge s211 \geq s311 \wedge s212 \geq s222 \wedge s221 \geq s231$
8000	200	742218	$s133 = s223 \wedge s133 = s232 \wedge s133 = s233 \wedge s133 = s313 \wedge s133 = s322 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 1 \wedge s131 \leq 2 \wedge s132 \leq 1 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 1 \wedge s221 \leq 2 \wedge s222 \leq 1 \wedge s231 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 1 \wedge s321 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge s111 \geq s131 \wedge s111 > s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 > s221 \wedge s111 > s222 \wedge s111 > s231 \wedge s111 \geq s311 \wedge s111 > s312 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s112 \geq s132 \wedge s112 \geq s133 \wedge s112 \geq s213 \wedge s113 \geq s133 \wedge s121 \geq s131 \wedge s121 \geq s133 \wedge s121 \geq s231 \wedge s122 \geq s123 \wedge s122 \geq s132 \wedge s122 \geq s133 \wedge s123 \geq s133 \wedge s131 \geq s133 \wedge s131 \geq s231 \wedge s132 \geq s133 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s211 \geq s213 \wedge s211 \geq s231 \wedge s211 \geq s311 \wedge s212 \geq s222 \wedge s221 \geq s231 \wedge s221 \geq s311 \wedge s311 \geq s321$
9000	200	598846	$s123 = s133 \wedge s123 = s223 \wedge s123 = s232 \wedge s123 = s233 \wedge s123 = s313 \wedge s123 = s322 \wedge s123 = s323 \wedge s123 = s331 \wedge s123 = s332 \wedge s123 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 0 \wedge s131 \leq 2 \wedge s132 \leq 1 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 1 \wedge s221 \leq 2 \wedge s222 \leq 1 \wedge s231 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 1 \wedge s321 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge s111 \geq s131 \wedge s111 > s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 \geq s221 \wedge s111 > s222 \wedge s111 > s231 \wedge s111 \geq s311 \wedge s111 > s312 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s112 \geq s132 \wedge s112 \geq s133 \wedge s112 \geq s213 \wedge s113 \geq s133 \wedge s121 \geq s131 \wedge s121 \geq s231 \wedge s122 \geq s123 \wedge s122 \geq s132 \wedge s122 \geq s133 \wedge s123 \geq s133 \wedge s131 \geq s133 \wedge s132 \geq s133 \wedge s132 \leq s211 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s321 \wedge s211 \geq s231 \wedge s211 \geq s311 \wedge s211 \geq s321 \wedge s212 \geq s222 \wedge s221 \geq s231 \wedge s221 \geq s311 \wedge s311 \geq s321$
10000	200	645738	$s133 = s223 \wedge s133 = s232 \wedge s133 = s233 \wedge s133 = s313 \wedge s133 = s322 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 1 \wedge s131 \leq 2 \wedge s132 \leq 1 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 1 \wedge s221 \leq 2 \wedge s222 \leq 1 \wedge s231 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 1 \wedge s321 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge s111 \geq s131 \wedge s111 > s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 \geq s221 \wedge s111 > s222 \wedge s111 > s231 \wedge s111 \geq s311 \wedge s111 > s312 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s112 \geq s132 \wedge s112 \geq s133 \wedge s112 \geq s213 \wedge s113 \geq s133 \wedge s121 \geq s131 \wedge s121 \geq s132 \wedge s121 \geq s133 \wedge s121 \geq s231 \wedge s122 \geq s132 \wedge s122 \geq s133 \wedge s122 \geq s133 \wedge s123 \geq s131 \wedge s123 \geq s133 \wedge s132 \geq s133 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s321 \wedge s211 \geq s311 \wedge s212 \geq s213 \wedge s212 \geq s312 \wedge s213 \leq s222 \wedge s221 \geq s231$

Table A.17: Virus infection system submodel sizes and invariants for different parameter configurations (cont.).

Traces	Length	States	Invariant
1000	300	355686	$s132 = s133 \wedge s132 = s223 \wedge s132 = s232 \wedge s132 = s233 \wedge s132 = s313 \wedge s132 = s322 \wedge s132 = s323 \wedge s132 = s331 \wedge s132 = s332 \wedge s132 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 1 \wedge s131 \leq 2 \wedge s132 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 1 \wedge s221 \leq 2 \wedge s222 \leq 1 \wedge s231 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 1 \wedge s321 \leq 2 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge s111 \geq s131 \wedge s111 > s132 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 \geq s221 \wedge s111 > s222 \wedge s111 > s231 \wedge s111 \geq s311 \wedge s111 > s312 \wedge s111 \geq s321 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s112 \geq s132 \wedge s112 \geq s213 \wedge s112 \geq s222 \wedge s112 \geq s312 \wedge s113 \geq s123 \wedge s113 \geq s132 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s121 \geq s132 \wedge s121 \geq s213 \wedge s121 \geq s222 \wedge s121 \geq s231 \wedge s122 \geq s132 \wedge s122 \geq s222 \wedge s122 \geq s231 \wedge s123 \geq s132 \wedge s123 \leq s211 \wedge s131 \geq s132 \wedge s131 \geq s231 \wedge s132 \leq s211 \wedge s132 \leq s212 \wedge s132 \leq s213 \wedge s132 \leq s221 \wedge s132 \leq s222 \wedge s132 \leq s231 \wedge s132 \leq s311 \wedge s132 \leq s312 \wedge s132 \leq s321 \wedge s211 \geq s311 \wedge s211 \geq s312 \wedge s211 \geq s321 \wedge s212 \geq s213 \wedge s212 \geq s312 \wedge s311 \geq s321$
2000	300	808516	$s133 = s223 \wedge s133 = s232 \wedge s133 = s233 \wedge s133 = s313 \wedge s133 = s322 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 2 \wedge s131 \leq 2 \wedge s132 \leq 1 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 1 \wedge s221 \leq 2 \wedge s222 \leq 1 \wedge s231 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 1 \wedge s321 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 \geq s123 \wedge s111 \geq s131 \wedge s111 > s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 \geq s221 \wedge s111 > s222 \wedge s111 > s231 \wedge s111 \geq s311 \wedge s111 > s312 \wedge s111 \geq s321 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s112 \geq s133 \wedge s113 \geq s133 \wedge s121 \geq s131 \wedge s121 \geq s132 \wedge s121 \geq s133 \wedge s121 \geq s231 \wedge s121 \geq s321 \wedge s122 \geq s132 \wedge s122 \geq s133 \wedge s123 \geq s133 \wedge s131 \geq s132 \wedge s131 \geq s133 \wedge s132 \geq s133 \wedge s132 \leq s221 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s321 \wedge s211 \geq s311 \wedge s211 \geq s321 \wedge s212 \geq s213 \wedge s221 \geq s231 \wedge s311 \geq s321$
3000	300	2803116	$s133 = s223 \wedge s133 = s231 \wedge s133 = s233 \wedge s133 = s313 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 1 \wedge s131 \leq 2 \wedge s132 \leq 2 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 1 \wedge s221 \leq 2 \wedge s222 \leq 2 \wedge s232 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 2 \wedge s321 \leq 1 \wedge s322 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 \geq s123 \wedge s111 \geq s131 \wedge s111 \geq s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 \geq s231 \wedge s111 \geq s311 \wedge s111 \geq s312 \wedge s111 > s321 \wedge s111 > s322 \wedge s112 \geq s113 \wedge s112 \geq s133 \wedge s112 \geq s213 \wedge s113 \geq s133 \wedge s121 \geq s131 \wedge s121 \geq s133 \wedge s122 \geq s133 \wedge s123 \geq s133 \wedge s131 \geq s133 \wedge s132 \geq s133 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s321 \wedge s133 \leq s322 \wedge s211 \geq s213 \wedge s211 \geq s232 \wedge s212 \geq s213 \wedge s212 \geq s232 \wedge s212 \geq s322 \wedge s213 \leq s222 \wedge s213 \leq s232 \wedge s213 \leq s322 \wedge s221 \geq s321 \wedge s222 \geq s232 \wedge s222 \geq s322 \wedge s232 \leq s322$
4000	300	1982414	$s133 = s223 \wedge s133 = s232 \wedge s133 = s233 \wedge s133 = s322 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 2 \wedge s131 \leq 2 \wedge s132 \leq 1 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 2 \wedge s221 \leq 2 \wedge s222 \leq 1 \wedge s231 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 1 \wedge s313 \leq 1 \wedge s321 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 \geq s123 \wedge s111 \geq s131 \wedge s111 > s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 \geq s221 \wedge s111 > s222 \wedge s111 > s231 \wedge s111 \geq s311 \wedge s111 > s312 \wedge s111 > s313 \wedge s111 > s321 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s112 \geq s133 \wedge s112 \geq s313 \wedge s113 \geq s133 \wedge s121 \geq s131 \wedge s121 \geq s133 \wedge s121 \geq s313 \wedge s122 \geq s123 \wedge s122 \geq s133 \wedge s123 \geq s133 \wedge s131 \geq s133 \wedge s132 \geq s133 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s313 \wedge s133 \leq s321 \wedge s211 \geq s311 \wedge s212 \geq s313 \wedge s213 \geq s313 \wedge s312 \geq s313$
5000	300	3256902	$s133 = s223 \wedge s133 = s232 \wedge s133 = s233 \wedge s133 = s313 \wedge s133 = s322 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 2 \wedge s131 \leq 2 \wedge s132 \leq 1 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 1 \wedge s221 \leq 2 \wedge s222 \leq 2 \wedge s231 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 1 \wedge s321 \leq 2 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 \geq s123 \wedge s111 \geq s131 \wedge s111 > s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 \geq s221 \wedge s111 > s222 \wedge s111 > s231 \wedge s111 \geq s311 \wedge s111 > s312 \wedge s111 > s321 \wedge s112 \geq s113 \wedge s112 \geq s123 \wedge s112 \geq s133 \wedge s113 \geq s133 \wedge s121 \geq s131 \wedge s121 \geq s133 \wedge s122 \geq s133 \wedge s123 \geq s133 \wedge s131 \geq s133 \wedge s132 \geq s133 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s321 \wedge s211 \geq s311 \wedge s211 \geq s321$
7000	300	4523552	$s133 = s223 \wedge s133 = s232 \wedge s133 = s233 \wedge s133 = s313 \wedge s133 = s322 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 1 \wedge s131 \leq 2 \wedge s132 \leq 1 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 2 \wedge s221 \leq 2 \wedge s222 \leq 2 \wedge s231 \leq 2 \wedge s311 \leq 2 \wedge s312 \leq 1 \wedge s321 \leq 1 \wedge s331 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge s111 \geq s131 \wedge s111 > s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s213 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 \geq s231 \wedge s111 \geq s311 \wedge s111 \geq s312 \wedge s111 \geq s321 \wedge s112 \geq s113 \wedge s112 \geq s133 \wedge s113 \geq s133 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s121 \geq s132 \wedge s121 \geq s133 \wedge s121 \geq s331 \wedge s122 \geq s133 \wedge s123 \geq s133 \wedge s131 \geq s133 \wedge s132 \geq s133 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s321 \wedge s133 \leq s331 \wedge s211 \geq s311 \wedge s211 \geq s331 \wedge s231 \geq s331$
8000	300	8315973	$s133 = s223 \wedge s133 = s232 \wedge s133 = s233 \wedge s133 = s313 \wedge s133 = s322 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 2 \wedge s131 \leq 2 \wedge s132 \leq 2 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 1 \wedge s221 \leq 2 \wedge s222 \leq 2 \wedge s231 \leq 2 \wedge s311 \leq 2 \wedge s312 \leq 2 \wedge s321 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 \geq s123 \wedge s111 \geq s131 \wedge s111 \geq s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 \geq s231 \wedge s111 \geq s311 \wedge s111 \geq s312 \wedge s111 \geq s321 \wedge s112 \geq s113 \wedge s112 \geq s133 \wedge s113 \geq s133 \wedge s121 \geq s123 \wedge s121 \geq s131 \wedge s121 \geq s132 \wedge s121 \geq s133 \wedge s121 \geq s231 \wedge s121 \geq s321 \wedge s122 \geq s132 \wedge s122 \geq s133 \wedge s123 \geq s133 \wedge s131 \geq s133 \wedge s132 \geq s133 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s321 \wedge s211 \geq s311 \wedge s211 \geq s321 \wedge s212 \geq s213 \wedge s221 \geq s231 \wedge s311 \geq s321$
10000	300	9141232	$s133 = s223 \wedge s133 = s233 \wedge s133 = s313 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 2 \wedge s131 \leq 2 \wedge s132 \leq 2 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 2 \wedge s221 \leq 2 \wedge s222 \leq 1 \wedge s231 \leq 2 \wedge s232 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 1 \wedge s321 \leq 2 \wedge s322 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 \geq s123 \wedge s111 \geq s131 \wedge s111 \geq s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 \geq s221 \wedge s111 > s222 \wedge s111 > s231 \wedge s111 > s232 \wedge s111 \geq s311 \wedge s111 > s312 \wedge s111 \geq s321 \wedge s111 > s322 \wedge s112 \geq s113 \wedge s112 \geq s133 \wedge s113 \geq s133 \wedge s121 \geq s131 \wedge s121 \geq s133 \wedge s121 \geq s232 \wedge s121 \geq s322 \wedge s122 \geq s133 \wedge s122 \geq s232 \wedge s123 \geq s133 \wedge s131 \geq s133 \wedge s131 \geq s232 \wedge s132 \geq s133 \wedge s132 \geq s232 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s321 \wedge s133 \leq s322 \wedge s211 \geq s311 \wedge s211 \geq s322 \wedge s311 \geq s322 \wedge s312 \geq s322 \wedge s321 \geq s322$

Table A.18: Virus infection system submodel sizes and invariants for different parameter configurations (cont.).

Traces	Length	States	Invariant
1000	400	3128661	$s133 = s223 \wedge s133 = s233 \wedge s133 = s313 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 1 \wedge s131 \leq 2 \wedge s132 \leq 2 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 1 \wedge s221 \leq 2 \wedge s222 \leq 2 \wedge s231 \leq 1 \wedge s232 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 1 \wedge s321 \leq 1 \wedge s322 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 > s123 \wedge s111 \geq s131 \wedge s111 \geq s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 > s213 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 > s231 \wedge s111 > s232 \wedge s111 \geq s311 \wedge s111 > s312 \wedge s111 > s321 \wedge s111 > s322 \wedge s112 \geq s113 \wedge s112 \geq s133 \wedge s112 \geq s232 \wedge s112 \geq s322 \wedge s113 \geq s133 \wedge s121 \geq s131 \wedge s121 \geq s133 \wedge s121 \geq s232 \wedge s121 \geq s322 \wedge s122 \geq s133 \wedge s122 \geq s232 \wedge s122 \geq s322 \wedge s123 \geq s133 \wedge s123 \geq s322 \wedge s131 \geq s133 \wedge s131 \geq s322 \wedge s132 \geq s133 \wedge s132 \geq s232 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s231 \wedge s133 \leq s232 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s321 \wedge s133 \leq s322 \wedge s211 \geq s231 \wedge s211 \geq s311 \wedge s211 \geq s321 \wedge s211 \geq s322 \wedge s212 \geq s213 \wedge s212 \geq s312 \wedge s221 \geq s232 \wedge s222 \geq s232 \wedge s222 \geq s322$
3000	400	7925064	$s133 = s223 \wedge s133 = s232 \wedge s133 = s233 \wedge s133 = s313 \wedge s133 = s322 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 2 \wedge s131 \leq 2 \wedge s132 \leq 2 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 2 \wedge s221 \leq 2 \wedge s222 \leq 2 \wedge s231 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 2 \wedge s321 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 \geq s123 \wedge s111 \geq s131 \wedge s111 \geq s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s213 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 > s223 \wedge s111 \geq s231 \wedge s111 \geq s311 \wedge s111 \geq s312 \wedge s111 > s321 \wedge s112 \geq s113 \wedge s112 \geq s133 \wedge s112 \geq s223 \wedge s113 \geq s133 \wedge s121 \geq s131 \wedge s121 \geq s133 \wedge s121 \geq s223 \wedge s122 \geq s133 \wedge s123 \geq s133 \wedge s131 \geq s133 \wedge s132 \geq s133 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s223 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s321 \wedge s211 \geq s311 \wedge s211 \geq s312 \wedge s211 \geq s313 \wedge s213 \geq s313 \wedge s221 \geq s223 \wedge s222 \geq s223$
4000	400	13385277	$s223 = s232 \wedge s223 = s233 \wedge s223 = s313 \wedge s223 = s322 \wedge s223 = s323 \wedge s223 = s331 \wedge s223 = s332 \wedge s223 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 2 \wedge s131 \leq 2 \wedge s132 \leq 2 \wedge s133 \leq 1 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 2 \wedge s221 \leq 2 \wedge s222 \leq 2 \wedge s223 \leq 0 \wedge s231 \leq 2 \wedge s311 \leq 2 \wedge s312 \leq 2 \wedge s321 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 \geq s123 \wedge s111 \geq s131 \wedge s111 \geq s132 \wedge s111 > s133 \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s213 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 > s223 \wedge true \wedge s111 \geq s231 \wedge s111 \geq s311 \wedge s111 \geq s312 \wedge s111 > s321 \wedge s112 \geq s113 \wedge s112 \geq s133 \wedge s112 \geq s223 \wedge s113 \geq s223 \wedge s121 \geq s131 \wedge s121 \geq s133 \wedge s121 \geq s223 \wedge s122 \geq s133 \wedge s122 \geq s223 \wedge s123 \geq s133 \wedge s123 \geq s223 \wedge s131 \geq s133 \wedge s131 \geq s223 \wedge s132 \geq s223 \wedge s133 \leq s211 \wedge s133 \geq s223 \wedge s211 \geq s223 \wedge s211 \geq s311 \wedge s212 \geq s223 \wedge s213 \geq s223 \wedge s221 \geq s223 \wedge s222 \geq s223 \wedge s223 \leq s231 \wedge s223 \leq s311 \wedge s223 \leq s312 \wedge s223 \leq s321$
1000	500	10495696	$s133 = s232 \wedge s133 = s233 \wedge s133 = s313 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 2 \wedge s131 \leq 2 \wedge s132 \leq 2 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 2 \wedge s221 \leq 2 \wedge s222 \leq 2 \wedge s223 \leq 1 \wedge s231 \leq 1 \wedge s311 \leq 2 \wedge s312 \leq 2 \wedge s321 \leq 1 \wedge s322 \leq 1 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 \geq s123 \wedge s111 \geq s131 \wedge s111 \geq s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s213 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 > s223 \wedge s111 \geq s231 \wedge s111 \geq s311 \wedge s111 \geq s312 \wedge s111 > s321 \wedge s112 \geq s113 \wedge s112 \geq s133 \wedge s112 \geq s213 \wedge s112 \geq s223 \wedge s113 \geq s133 \wedge s113 \geq s223 \wedge s121 \geq s131 \wedge s121 \geq s133 \wedge s121 \geq s223 \wedge s122 \geq s133 \wedge s122 \geq s223 \wedge s123 \geq s133 \wedge s123 \geq s223 \wedge s131 \geq s133 \wedge s132 \geq s133 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s223 \wedge s211 \geq s231 \wedge s133 \leq s311 \wedge s133 \leq s321 \wedge s211 \geq s322 \wedge s311 \geq s211 \geq s322 \wedge s212 \geq s223 \wedge s212 \geq s322 \wedge s213 \geq s223 \wedge s312 \geq s322$
2000	500	21603820	$s133 = s232 \wedge s133 = s233 \wedge s133 = s322 \wedge s133 = s323 \wedge s133 = s331 \wedge s133 = s332 \wedge s133 = s333 \wedge s111 \leq 2 \wedge s111 \geq 2 \wedge true \wedge s112 \leq 2 \wedge s113 \leq 2 \wedge s121 \leq 2 \wedge s122 \leq 2 \wedge s123 \leq 2 \wedge s131 \leq 2 \wedge s132 \leq 2 \wedge s133 \leq 0 \wedge s211 \leq 2 \wedge s212 \leq 2 \wedge s213 \leq 2 \wedge s221 \leq 2 \wedge s222 \leq 2 \wedge s223 \leq 1 \wedge s231 \leq 2 \wedge s311 \leq 2 \wedge s312 \leq 2 \wedge s313 \leq 1 \wedge s321 \leq 2 \wedge s111 \geq s112 \wedge s111 \geq s113 \wedge s111 \geq s121 \wedge s111 \geq s122 \wedge s111 \geq s123 \wedge s111 \geq s131 \wedge s111 \geq s132 \wedge s111 > s133 \wedge true \wedge s111 \geq s211 \wedge s111 \geq s212 \wedge s111 \geq s213 \wedge s111 \geq s221 \wedge s111 \geq s222 \wedge s111 > s223 \wedge s111 \geq s231 \wedge s111 \geq s311 \wedge s111 \geq s312 \wedge s111 > s321 \wedge s112 \geq s113 \wedge s112 \geq s133 \wedge s112 \geq s223 \wedge s113 \geq s313 \wedge s121 \geq s131 \wedge s121 \geq s133 \wedge s121 \geq s223 \wedge s121 \geq s313 \wedge s122 \geq s133 \wedge s122 \geq s223 \wedge s123 \geq s133 \wedge s131 \geq s133 \wedge s132 \geq s133 \wedge s133 \leq s211 \wedge s133 \leq s212 \wedge s133 \leq s213 \wedge s133 \leq s221 \wedge s133 \leq s222 \wedge s133 \leq s223 \wedge s133 \leq s231 \wedge s133 \leq s311 \wedge s133 \leq s312 \wedge s133 \leq s313 \wedge s133 \leq s321 \wedge s211 \geq s223 \wedge s211 \geq s311 \wedge s211 \geq s312 \wedge s211 \geq s313 \wedge s213 \geq s313 \wedge s221 \geq s223 \wedge s222 \geq s223$

Table A.19: Virus infection system submodel sizes and invariants for different parameter configurations (cont.).

Bibliography

[BdA95] Andrea Bianco and Luca de Alfaro. Model checking of probabilistic and nondeterministic systems. *Proc. Foundations of Software Technology and Theoretical Computer Science*, 1026:499–513, 1995.

[BFd+14] Mateus Borges, Antonio Filieri, Marcelo d'Amorim, Corina S. Păsăreanu, and Willem Visser. Compositional solution space quantification for probabilistic software analysis. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '14, pages 123–132, New York, NY, USA, 2014. ACM.

[BFFHH11] J. Bogdoll, L.M. Ferrer Fioriti, A. Hartmanns, and H. Hermanns. Partial order methods for statistical model checking and simulation. In *FMOODS/FORTE*, pages 59–74, 2011.

[BGC09] Christel Baier, Marcus Groesser, and Frank Ciesinski. Quantitative Analysis under Fairness Constraints. In *ATVA*, pages 135–150, 2009.

[BGH09] S. Basu, A. Ghosh, and R. He. Approximate model checking of PCTL involving unbounded path properties. *ICFEM'09*, pages 326–346, 2009.

[BH97] Christel Baier and Holger Hermanns. Weak Bisimulation for Fully Probabilistic Processes. In *Computer Aided Verification*, pages 119 – 130, 1997.

[BK98] Christel Baier and Marta Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11(3):125–155, August 1998.

[BK08] C. Baier and J.P. Katoen. *Principles of model checking*. MIT press, 2008.

[CBvB12] Elise Cormie-Bowins and Franck van Breugel. Measuring progress of probabilistic LTL model checking. *arXiv preprint arXiv:1207.0870*, 2012.

[CGMP99] Edmund M Clarke, Orna Grumberg, Marius Minea, and Doron Peled. State space reduction using partial order techniques. *International Journal on Software Tools for Technology Transfer*, 2(3):279–287, 1999.

[CGP99] Edmund Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. The MIT Press, 1999.

[Che52] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[Che80] R. C. Cheung. A user-oriented software reliability model. *IEEE Transactions on Software Engineering*, 6(2):118–125, March 1980.

[Chr90] Ivan Christoff. Testing equivalences and fully abstract models for probabilistic processes. In *CONCUR 1990*, pages 126–138, 1990.

[Col01] S. Coles. *An Introduction to Statistical Modelling of Extreme Values*. Springer Series in Statistics. Springer, 2001.

[CS02] Stefano Cattani and Roberto Segala. Decision Algorithms for Probabilistic Bisimulation. In *CONCUR 2002 - Concurrency Theory*, pages 371–386, 2002.

[dA97] Luca de Alfaro. *Formal Verification of Probabilistic Systems*. Ph.D., Stanford University, 1997.

[DCL11] Benoît Delahaye, Benoît Caillaud, and Axel Legay. Probabilistic contracts: a compositional reasoning methodology for the design of systems with stochastic and/or non-deterministic aspects. *Formal Methods in System Design*, 2011.

[DG97] T. Dean and R. Givan. Model minimization in Markov decision processes. In *Proceedings of the National Conference on Artificial Intelligence*, pages 106–111, 1997.

[DHK99] Pedro D'Argenio, Holger Hermanns, and Joost-Pieter Katoen. On Generative Parallel Composition. *Electronic Notes in Theoretical Computer Science*, 22:30–54, 1999.

[DJJL01] P. D'Argenio, B. Jeannet, H. Jensen, and K. Larsen. Reachability analysis of probabilistic systems by successive refinements. In *PAPM/PROBMIV*, volume 2165 of *LNCS*, pages 39–56. Springer, 2001.

[DNKLM06] Rocco De Nicola, Joost-Pieter Katoen, Diego Latella, and Mieke Massink. Towards a logic for performance and mobility. *Electronic Notes in Theoretical Computer Science*, 153(2):161–175, 2006.

[DV90] Rocco De Nicola and Frits Vaandrager. Action versus state based logics for transition systems. *Semantics of Systems of Concurrent Processes*, 469:407–419, 1990.

[EC82] E. Allen Emerson and Edmund Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, December 1982.

[EGMT09] Ilenia Epifani, Carlo Ghezzi, Raffaela Mirandola, and Giordano Tamburrelli. Model Evolution by Run-Time Parameter Adaptation. In *International Conference on Software Engineering (ICSE)*, pages 111–121. IEEE, 2009.

[EPG+07] Michael D. Ernst, Jeff H. Perkins, Philip J. Guo, Stephen McCamant, Carlos Pacheco, Matthew S. Tschantz, and Chen Xiao. The Daikon system for dynamic detection of likely invariants. *Sci. Comput. Program.*, 69(1-3):35–45, December 2007.

[Fel08] William Feller. *An introduction to probability theory and its applications*, volume 1. John Wiley & Sons, 2008.

[FPV13] Antonio Filieri, Corina S. Pasareanu, and Willem Visser. Reliability analysis in symbolic pathfinder. In *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*, pages 622–631, 2013.

[HdA01] Thomas Henzinger and Luca de Alfaro. Interface automata. *ACM SIGSOFT Software Engineering Notes*, 26(5):109–120, 2001.

[HJ89] Hans Hansson and Bengt Jonsson. A framework for reasoning about time and reliability. In *Proceedings Real-Time Systems Symposium 1989*, pages 102–111. IEEE Comput. Soc. Press, 1989.

[HJ90] Hans Hansson and Bengt Jonsson. A calculus for communicating systems with time and probabilities. In *Real-Time Systems Symposium, 1990. Proceedings., 11th*, pages 278–287. IEEE, 1990.

[HJB+10] Ru He, Paul Jennings, Samik Basu, Arka P Ghosh, and Huaiqing Wu. A bounded statistical approach for model checking of unbounded until properties. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, pages 225–234. ACM, 2010.

[HK09] Holger Hermanns and Joost-Pieter Katoen. The How and Why of Interactive Markov Chains. In *Formal Methods for Components and Objects*, pages 311–337, 2009.

[HKK13] Holger Hermanns, Jan Krčál, and Jan Křetínský. Compositional Verification and Optimization of Interactive Markov Chains. In *CONCUR 2013 - Concurrency Theory*, pages 364–379. Springer, 2013.

[HKNP06] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 3920, pages 441–444. Springer, Springer, 2006.

[HMKS99] H. Hermanns, J. Meyer-Kayser, and M. Siegle. Multi terminal binary decision diagrams to represent and analyse continuous time Markov chains. In *Proc. NSMC'99*, pages 188–207. Prensas Universitarias de Zaragoza, 1999.

[HMZ+12] D. Henriques, J. Martins, P. Zuliani, A. Platzer, and E. Clarke. Statistical model checking for markov decision processes. In *QEST*, pages 84–93, 2012.

[Hoa78] Charles Anthony Richard Hoare. Communicating Sequential Processes. *Communications of the ACM*, 21(8):666–677, 1978.

[HP00] Klaus Havelund and Thomas Pressburger. Model checking java programs using java pathfinder. *International Journal on Software Tools for Technology Transfer*, 2(4):366–381, 2000.

[HSV94] L. Helmink, M. Sellink, and F. Vaandrager. Proof-checking a data link protocol. In *Proc. International Workshop on Types for Proofs and Programs (TYPES'93)*, volume 806 of *LNCS*. Springer, 1994.

[Ins97] Institute of Electrical and Electronic Engineers. IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1997.

[JD07] L.H. Jamieson and B.C. Dean. Weighted alliances in graphs. *Congressus Numerantium*, 187:76, 2007.

[JS99] JP Jarvis and Douglas R Shier. Graph-theoretic analysis of finite markov chains. *Applied mathematical modeling: a multidisciplinary approach*, 1999.

[Kel76] Robert Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, 1976.

[KJD02] L.M. Kaufman, B.W. Johnson, and J.B. Dugan. Coverage estimation using statistics of the extremes for when testing reveals no failures. *IEEE Transactions on Computers*, pages 3–12, 2002.

[KKZ05] J.P. Katoen, M. Khattri, and IS Zapreevt. A Markov reward model checker. In *QEST'05*, pages 243–244. IEEE, 2005.

[KNP06] M. Kwiatkowska, G. Norman, and D. Parker. Symmetry reduction for probabilistic model checking. In *Computer Aided Verification*, pages 234–248. Springer, 2006.

[KNPQ10] Marta Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. Assume-guarantee verification for probabilistic systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 23–37. Springer, 2010.

[KNPV09] Marta Kwiatkowska, Gethin Norman, David Parker, and Maria Grazia Vigliotti. Probabilistic mobile ambients. *Theoretical Computer Science*, 410(12):1272–1303, 2009.

[Kul09] Vidyadhar Kulkarni. *Modeling and Analysis of Stochastic Systems*. CRC Press, 2009.

[Lan10] Kenneth Lange. *Applied probability*. Springer Science & Business Media, 2010.

[LLPY97] Kim Guldstrand Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. Efficient verification of real-time systems: compact data structure and state-space reduction. In *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, pages 14–24. IEEE, 1997.

[LP06] R. Lassaigne and S. Peyronnet. Probabilistic verification and approximation. *ENTCS*, 143:101–114, 2006.

[LPD+14] Kasper Luckow, Corina S. Păsăreanu, Matthew B. Dwyer, Antonio Filieri, and Willem Visser. Exact and approximate probabilistic symbolic execution for nondeterministic programs. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ASE '14, pages 575–586, New York, NY, USA, 2014. ACM.

[LT87] Nancy Lynch and Mark Tuttle. Hierarchical correctness proofs for distributed algorithms. In *6th ACM Symposium on Principles of Distributed Computing*, volume pages, pages 137–151, 1987.

[Lyu96] Michael R. Lyu. *Handbook of software reliability engineering*. McGraw-Hill, Inc., Hightstown, NJ, USA, 1996.

[Mil89] Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[MSW12] Linar Mikeev, Werner Sandmann, and Verena Wolf. Efficient calculation of rare event probabilities in markovian queueing networks. ICST, 6 2012.

[MSW13] Linar Mikeev, Werner Sandmann, and Verena Wolf. Numerical Approximation of Rare Event Probabilities in Biochemically Reacting Systems. In Ashutosh Gupta and ThomasA. Henzinger, editors, *Computational Methods in Systems Biology*, volume 8130 of *Lecture Notes in Computer Science*, pages 5–18. Springer Berlin Heidelberg, 2013.

[Mus93] John Musa. Operational Profiles in Software Reliability Engineering. *IEEE Software*, 10(March):14–32, 1993.

[Nim10] V. Nimal. Statistical approaches for probabilistic model checking. M.Sc. Dissertation, Oxford University Computing Laboratory, 2010.

[PBU09] Esteban Pavese, Víctor Braberman, and Sebastian Uchitel. Probabilistic environments in the quantitative analysis of (non-probabilistic) behaviour models. In *Proceedings of ESEC-FSE 2009*, page 335, New York, New York, USA, 2009. ACM Press.

[PBU10] Esteban Pavese, Víctor Braberman, and Sebastian Uchitel. My model checker died!: how well did it do? In *QUOVADIS/ICSE'10*, pages 33–40. ACM, 2010.

[PBU13] Esteban Pavese, Víctor Braberman, and Sebastian Uchitel. Automated reliability estimation over partial systematic explorations. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 602–611. IEEE Press, 2013.

[PLS00] Anna Philippou, Insup Lee, and Oleg Sokolsky. Weak Bisimulation for Probabilistic Systems. *Lecture Notes in Computer Science*, 1877:334–349, 2000.

[QS96] Muhammad A Qureshi and William H Sanders. A new methodology for calculating distributions of reward accumulated during a finite interval. In *Fault Tolerant Computing, 1996., Proceedings of Annual Symposium on*, pages 116–125. IEEE, 1996.

[RC05] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag New York, 2005.

[RdBSH13] Daniël Reijsbergen, Pieter-Tjerk de Boer, Werner Scheinhardt, and Boudewijn Haverkort. Automated rare event simulation for stochastic petri nets. In Kaustubh Joshi, Markus Siegle, Mariëlle Stoelinga, and Pedro R. D'Argenio, editors, *Quantitative Evaluation of Systems*, volume 8054 of *Lecture Notes in Computer Science*, pages 372–388. Springer Berlin Heidelberg, 2013.

[RK08] R.Y. Rubinstein and D.P. Kroese. *Simulation and the Monte Carlo method (Series in Probability and Statistics)*, volume 707. Wiley, 2008.

[RM04] Roshanak Roshandel and Nenad Medvidovic. Toward Architecture-Based Reliability Estimation. In *ICSE/WADS*, pages 2–6, 2004.

[RP09] D. Rabih and N. Pekergin. Statistical model checking using perfect simulation. In *Proc. ATVA'09*, pages 120–134. Springer-Verlag, 2009.

[Saw03] Shlomo S Sawilowsky. You think you've got trivials? *Journal of Modern Applied Statistical Methods*, 2(1):21, 2003.

[SdV04] Ana Sokolova and Erik de Vink. Probabilistic automata: system types, parallel composition and comparison. *Validation of Stochastic Systems*, pages 377–385, 2004.

[Seg95] Roberto Segala. *Modelling and verification of randomized distributed real time systems*. PhD thesis, Massachusetts Institute of Technology, 1995.

[SI97] Ushio Sumita and Nobuko Igaki. Necessary and Sufficient Conditions for Global Geometric Convergence of Block Gauss-Seidel Iteration Algorithm Applied to Markov Chains. *Journal of the Operations Research Society of Japan-Keiei Kagaku*, 40(3):283–293, 1997.

[SJ90] Scott Smolka and Chi-Chang Jou. Equivalences, congruences, and complete axiomatizations for probabilistic processes. In *CONCUR '90 Theories of Concurrency: Unification and Extension*, pages 367–383, 1990.

[SL95] Roberto Segala and Nancy Lynch. Probabilistic Simulations for Probabilistic Processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

[SVA05a] K. Sen, M. Viswanathan, and G. Agha. On statistical model checking of stochastic systems. In *Proc. CAV'05*, pages 266–280. Springer, 2005.

[SVA05b] K. Sen, M. Viswanathan, and G. Agha. VESTA: A statistical model-checker and analyzer for probabilistic systems. In *QEST'05*, pages 251–252. IEEE, 2005.

[Var85] Moshe Vardi. Automatic verification of probabilistic concurrent finite state programs. In *26th Annual Symposium on Foundations of Computer Science (SFCS 1985)*, pages 327–338. IEEE, October 1985.

[VAVA94] Manuel Villén-Altamirano and José Villén-Altamirano. RESTART: a straightforward method for fast simulation of rare events. In *Proc. WSC'94*, pages 282–289, San Diego, USA, 1994.

[vGSS95] Rob van Glabbeek, Scott Smolka, and Bernhard Steffen. Reactive, generative, and stratified models of probabilistic processes. *Information and Computation*, 121(1):59–80, 1995.

[WSS97] Sue-Hwey Wu, Scott Smolka, and Eugene Stark. Composition and Behaviors of Probabilistic I/O Automata. *Theoretical Computer Science*, 176(1):1–38, 1997.

[YCZ11] H. Younes, E. Clarke, and P. Zuliani. Statistical verification of probabilistic properties with unbounded until. *Formal Methods: Foundations and Applications*, pages 144–160, 2011.

[You05] H. Younes. Ymer: A statistical model checker. In *Computer Aided Verification*, pages 171–179. Springer, 2005.

[ZVB11] Xin Zhang and Franck Van Breugel. A progress measure for explicit-state probabilistic model-checkers. In *Automata, Languages and Programming*, pages 283–294. Springer, 2011.

List of Figures

List of Tables