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

Validación Experimental de Abstracciones Modales para Contratos Inteligentes

Tesis de Licenciatura en Ciencias de la Computación

Matías Nicolás Incem y Alejandra Alicia Rodríguez

Director: Dr. Diego Garbervetsky

Buenos Aires, 2025

VALIDACIÓN EXPERIMENTAL DE ABSTRACCIONES MODALES PARA CONTRATOS INTELIGENTES

Los contratos inteligentes (*Smart Contracts*) son programas que se ejecutan en *Blockchain* y administran activos de gran valor. Constituyen la base de las finanzas descentralizadas (DeFi), y su carácter inmutable, derivado de la propia *Blockchain*, implica que un error en su código puede derivar en pérdidas significativas. Esto vuelve esencial su verificación antes del despliegue.

Uno de los tipos más frecuentes de errores en los *Smart Contracts* son los de lógica de negocio. Para analizar este tipo de errores, *Predicate Abstractions* [1] y *Modal Abstractions* [2] proponen generar modelos abstractos a partir de contratos escritos en el lenguaje de programación Solidity, enfocándose en la verificación del comportamiento lógico de negocio.

Para esto, la herramienta *Alloy4PA* [3] implementa la propuesta introducida inicialmente en [1] y extendida en [2], utilizando *Alloy* en entornos Docker y Java.

Esta tesis se centra en comprender, reproducir y validar los experimentos descritos en el paper de *Modal Abstractions*, detallando el proceso de reproducción mediante la herramienta *Alloy4PA*, y ampliando la evaluación con nuevos casos de estudio que permitan explorar la generalidad y la robustez del enfoque propuesto.

Palabras claves: Contratos inteligentes, Blockchain, Solidity, Predicate Abstraction, Modal Abstraction, Verificación formal, Alloy4PA, Alloy.

EXPERIMENTAL VALIDATION OF MODAL ABSTRACTIONS FOR SMART CONTRACTS

Smart Contracts are programs that run on the *Blockchain* and manage high-value assets. They form the basis of decentralized finance (DeFi), and their immutable nature, inherited from the *Blockchain*, means that any error in their code can lead to significant losses. This makes their verification before deployment essential.

One of the most common types of errors in *Smart Contracts* are business logic flaws. To analyze this kind of issue, *Predicate Abstractions* [1] and *Modal Abstractions* [2] propose generating abstract models from contracts written in the Solidity programming language, focusing on verifying their logical business behavior.

For this purpose, the *Alloy4PA* tool [3] implements the approach originally introduced in [1] and extended in [2], using the *Alloy* language within Docker and Java environments.

This thesis focuses on understanding, reproducing, and validating the experiments described in the *Modal Abstractions* paper, detailing the reproduction process using the *Alloy4PA* tool, and extending the evaluation with new case studies to explore the generality and robustness of the proposed approach.

Keywords: Smart Contracts, Blockchain, Solidity, Predicate Abstraction, Modal Abstraction, Formal Verification, Alloy4PA, Alloy.

Índice general

1. Introducción 1
   1.1. Contexto y motivación 1
   1.2. Objetivo y alcance 1
   1.3. Estructura del documento 2

2. Marco teórico 3
   2.1. Contratos inteligentes y verificación formal 3
   2.2. El lenguaje Alloy 3
   2.3. Predicate Abstractions 3
   2.4. Modal Abstractions 4
   2.5. Alloy4PA 4
      2.5.1. Fases de abstracción en Alloy4PA 4
   2.6. Resumen y análisis del enfoque de abstracciones modales 5
      2.6.1. Ejemplo motivador: contrato de subasta 5
      2.6.2. Abstracciones may y must 5
      2.6.3. Transiciones con restricción 6
      2.6.4. Formalización del modelo y diseño del prototipo 6
      2.6.5. Evaluación experimental 7
      2.6.6. Resultados detallados 7
      2.6.7. Amenazas a la validez 8
      2.6.8. Trabajos relacionados 9
      2.6.9. Conclusiones del enfoque modal 9

3. Metodología de reproducción 11
   3.1. Preliminares y preparación del entorno 11
   3.2. Ejecución básica de la herramienta Alloy4PA 11
      3.2.1. Ejecución mediante JAR precompilado 11
      3.2.2. Ejecución compilando desde fuentes 11
      3.2.3. Ejecución mediante imagen de Docker 11
      3.2.4. Ejecución generando imagen de Docker propia 12
   3.3. Ejecución completa de la herramienta 12
   3.4. Outputs de la ejecución completa de la herramienta 12

4. Reproducción 13
   4.1. Verificación de Prerrequisitos 13
   4.2. Ejecución básica en Windows 13
      4.2.1. Inicialización del entorno 13
      4.2.2. Modalidad JAR precompilado 15
      4.2.3. Compilar archivo JAR desde el código fuente 20
      4.2.4. Modalidad generando una imagen de Docker 24
      4.2.5. Modalidad con imagen de Docker prearmada 28
   4.3. Ejecución básica en macOS 31
      4.3.1. Inicialización del entorno 31

4.3.2. Modalidad JAR precompilado . . . 32
4.3.3. Compilar archivo JAR desde el código fuente . . . 36
4.3.4. Modalidad con imagen de Docker prearmada . . . 38
4.3.5. Modalidad generando una imagen de Docker . . . 42
4.4. Reproducción completa de los experimentos . . . 44
4.4.1. Ejecución de ambos Benchmarks completos . . . 45
4.4.2. Reproducción de los datos obtenidos para RQ#1 . . . 51
4.4.3. Reproducción de los datos obtenidos para RQ#2 . . . 54
4.4.4. Reproducción de los datos obtenidos para RQ#3 . . . 56
4.5. Problemas encontrados . . . 56
4.5.1. Problema con rutas de archivos en Windows . . . 57

5. Nuevos ejemplos . . . 59
5.1. Metodología . . . 59
5.2. Simple Marketplace . . . 59
5.2.1. Contrato original . . . 60
5.2.2. Variante con bug de aceptar sin ofertas . . . 60
5.2.3. Comparación de los resultados . . . 60
5.3. Rock-Paper-Scissors . . . 61
5.3.1. Contrato original . . . 61
5.3.2. Variante con bug de pozo no vacío . . . 62
5.3.3. Comparación de los resultados . . . 63
5.3.4. Refinamiento de la variante . . . 64
5.3.5. Resultados de la variante refinada . . . 66
5.4. AssetTransfer . . . 66
5.4.1. Introducción al experimento . . . 66
5.4.2. Contrato original . . . 67
5.4.3. Bug en la función `reject()` . . . 67
5.4.4. Bug en `makeOffer` con EPA y particiones de estado . . . 72
5.4.5. Variante con transición `makeOffer` *must* y `accept` *may* . . . 79

6. Discusión . . . 83
6.1. Resultados de los experimentos . . . 83
6.2. Errores encontrados . . . 83
6.3. Limitaciones de usabilidad . . . 84
6.4. Análisis de las preguntas de investigación . . . 84
6.5. Síntesis general . . . 85

7. Conclusiones . . . 86
7.1. Síntesis del trabajo realizado . . . 86
7.2. Conclusiones respecto de las RQ . . . 86
7.3. Observaciones sobre desempeño y usabilidad . . . 86
7.4. Aportes y proyección futura . . . 86

1. INTRODUCCIÓN

1.1. Contexto y motivación

Blockchain es una estructura de datos agrupada en bloques encadenados. La información de un bloque solo puede ser modificada alterando todos los bloques posteriores, lo cual la vuelve prácticamente inmutable.

Esta propiedad permite su aplicación en entornos distribuidos, donde la Blockchain puede funcionar como una base de datos compartida con un histórico irrefutable de información.

Los contratos inteligentes (*Smart Contracts*) son programas que se ejecutan sobre Blockchain y administran activos de gran valor. Permiten establecer acuerdos confiables con menor intervención de intermediarios, reduciendo el riesgo de errores accidentales o comportamientos maliciosos.

Dado que los contratos inteligentes son inmutables una vez desplegados, un error en su código puede derivar en pérdidas significativas. Esto hace esencial su verificación antes del despliegue, especialmente considerando su papel central en las finanzas descentralizadas (DeFi).

En este contexto, las *abstracciones modales* (may/must) se proponen como un instrumento intermedio entre la verificación formal completa y la auditoría manual. Este enfoque genera modelos ejecutables y legibles que distinguen entre comportamientos posibles (may) y garantizados (must), facilitando la detección de defectos de lógica de negocio que no suelen evidenciarse en métodos puramente estáticos.

1.2. Objetivo y alcance

Esta tesis se centra en comprender, reproducir y validar los experimentos descritos en el paper *Modal Abstractions for Smart Contract Validation* [2], detallando el proceso de reproducción mediante la herramienta Alloy4PA [3], y ampliando la evaluación con nuevos casos de estudio que permitan explorar la generalidad y robustez del enfoque propuesto.

Además de reproducir los resultados del trabajo original, se busca analizar la aplicabilidad del enfoque modal a distintos contextos, evaluar la sensibilidad del modelo a variaciones en las condiciones experimentales (como el uso de *EPA=True* o *EPA=False*), y examinar los compromisos entre exhaustividad, granularidad y costo computacional.

Como parte del análisis, también se retoman las tres preguntas de investigación (*Research Questions*, RQ) del paper original, evaluando sus respuestas a la luz de los experimentos reproducidos y de las pruebas adicionales desarrolladas. Estas preguntas funcionan como un marco de referencia, complementado con observaciones y hallazgos propios surgidos durante el proceso de tesis:

- **RQ1.** ¿Qué tan frecuentes son las transiciones *must* en las abstracciones modales de contratos inteligentes?
- **RQ2.** ¿El uso de *constraint transitions* ayuda a validar contratos inteligentes cuando se dispone de roles de usuario permitidos?

- **RQ3.** ¿Los auditores aprovechan la distinción entre transiciones *may* y *must* para revelar problemas de comportamiento que no pueden capturarse utilizando únicamente abstracciones *may*?

Responder estas preguntas implica no solo reproducir los experimentos del trabajo original, sino también realizar un análisis empírico que permita valorar las ventajas, limitaciones y posibles mejoras del enfoque en términos de escalabilidad, interpretabilidad y utilidad práctica.

Como parte de garantizar la reproducibilidad y transparencia del proceso experimental, se proporciona acceso público a los artefactos generados durante esta tesis. El material utilizado y producido en el análisis —incluyendo modelos Alloy, contratos Solidity cuando corresponde, y los outputs obtenidos— se encuentra disponible en un repositorio público de GitHub[4], a fin de facilitar su trazabilidad, replicación y verificación independiente.

1.3. Estructura del documento

Este documento se organiza de la siguiente manera:

- **Capítulo 2 – Marco teórico.** Presenta los conceptos fundamentales de Blockchain, los contratos inteligentes y los principales enfoques de verificación formal. Luego describe en detalle el trabajo de *Modal Abstractions* y su implementación en la herramienta Alloy4PA.
- **Capítulo 3 – Metodología de reproducción.** Explica el procedimiento seguido para reproducir los experimentos del paper original, incluyendo la configuración del entorno (Windows y macOS), el uso de la herramienta Alloy4PA y las verificaciones de resultados.
- **Capítulo 4 – Reproducción.** Detalla la replicación de los experimentos del trabajo original y contrasta los resultados obtenidos con los publicados por los autores, evaluando su validez y reproducibilidad.
- **Capítulo 5 – Nuevos ejemplos.** Presenta ejemplos adicionales propuestos para extender la evaluación y explorar la capacidad del enfoque para detectar errores en contratos no incluidos en el trabajo original. Se incluyen comparaciones entre configuraciones con *EPA=True* y *EPA=False*, analizando su impacto en el tiempo de cómputo y la granularidad de las abstracciones.
- **Capítulo 6 – Discusión.** Analiza la reproducibilidad y extensión de los resultados, responde las preguntas de investigación (RQ1–RQ3) y sistematiza las limitaciones observadas en términos de usabilidad, performance y exhaustividad.
- **Capítulo 7 – Conclusiones.** Resume los aportes del trabajo, presenta las conclusiones generales, responde explícitamente las RQ y propone líneas de trabajo futuro orientadas a mejorar la automatización, escalabilidad y accesibilidad de la herramienta.

2. MARCO TEÓRICO

Este capítulo presenta los conceptos y trabajos previos relevantes para poner en contexto la herramienta y experimentos que se validarán y extenderán en esta tesis.

2.1. Contratos inteligentes y verificación formal

Los *Smart Contracts* se implementan mediante lenguajes de programación específicos para *Blockchain*. El más utilizado es *Solidity*, diseñado para la red Ethereum y convertido en el estándar del ecosistema. Otro ejemplo es *Move*, menos extendido, pero más orientado a la verificabilidad.

Un tipo frecuente de errores en los contratos inteligentes son los de lógica de negocio, es decir, diferencias entre el comportamiento esperado y la implementación concreta.

Habitualmente, los contratos son auditados por empresas especializadas que combinan métodos automáticos de análisis con revisiones manuales. La verificación formal busca aumentar la confianza en estos contratos mediante técnicas matemáticas que permiten razonar sobre todas las posibles ejecuciones de un programa.

2.2. El lenguaje Alloy

Alloy [5] es un lenguaje de modelado declarativo con soporte de verificación automática, ampliamente utilizado en el análisis formal de software. Permite especificar sistemas mediante relaciones y restricciones, y verificar automáticamente si dichas especificaciones cumplen ciertas propiedades o comportamientos esperados. Para ello, analiza sus propiedades utilizando solucionadores lógicos (*SAT solvers* [6]), que buscan automáticamente ejemplos o contraejemplos que satisfagan las condiciones definidas en el modelo.

2.3. Predicate Abstractions

La verificación formal de sistemas secuenciales tiene sus raíces en los autómatas finitos [7], los sistemas de transiciones etiquetadas (LTS) [8], y su evolución hacia los sistemas de transición modal (MTS) [9, 10].

En [1], los autores utilizan abstracción por predicados para construir un sistema de transiciones finito etiquetado (LTS) a partir del código fuente de contratos inteligentes. Este enfoque abstrae las secuencias válidas de llamadas a funciones aplicables al contrato, representando el flujo lógico de su comportamiento.

En dichas abstracciones, una transición etiquetada con f entre dos estados abstractos s_1 y s_2 indica que existe al menos un estado concreto en s_1 desde el cual puede invocarse la función f con ciertos parámetros adecuados, y que la ejecución termina exitosamente en un estado concreto de s_2.

Estas abstracciones ayudan a los auditores a analizar el comportamiento del contrato, identificar defectos y aumentar la confianza en que cumple con los requisitos esperados.

Por ejemplo, pueden revelar defectos como la imposibilidad de que un postor no ganador retire su depósito al finalizar una subasta, o la posibilidad de que un postor siga ofertando cuando la subasta ya está cerrada.

En este trabajo, las abstracciones obtenidas se denominan *may abstractions*, ya que describen comportamientos posibles pero no necesariamente garantizados.

2.4. Modal Abstractions

En [2], los autores extienden el enfoque anterior incorporando las llamadas *abstracciones modales*, que distinguen entre transiciones posibles (*may*) y garantizadas (*must*).

Introducen además el concepto de *transiciones con restricción*, que permite limitar la cuantificación existencial de ciertos parámetros a valores definidos por el usuario.

Por ejemplo, al restringir los retiros a un usuario o rol en particular (p. ej., `msg.sender = 0xFA0...`), la abstracción may-must resultante puede mostrar que el postor `0xFA0...` tiene garantizado poder retirar sus fondos si pierde la subasta.

En este trabajo, las principales contribuciones son:

1. una nueva abstracción modal para la validación de contratos inteligentes;
2. un prototipo para la construcción de estas abstracciones (*Alloy4PA*); y
3. una evaluación empírica sobre dos *benchmarks* establecidos: uno basado en Microsoft Azure Blockchain Workbench y otro sobre un conjunto de contratos previamente utilizados en [1].

2.5. Alloy4PA

El programa *Alloy4PA* [3] implementa la propuesta de [1, 2] utilizando el lenguaje *Alloy* dentro de entornos Docker y Java. Esta herramienta permite analizar las abstracciones modales de contratos inteligentes escritos en Solidity. Esto permite automatizar parte del proceso de validación, si bien requiere un paso manual previo de migrar el contrato de Solidity a Alloy.

2.5.1. Fases de abstracción en Alloy4PA

La herramienta Alloy4PA [3] implementa el enfoque de *Modal Abstractions* [2] mediante tres fases principales de abstracción. Estas fases permiten determinar, para cada operación del contrato, si la transición correspondiente debe clasificarse como *may* o *must* en la abstracción modal.

EPA (Enumerated Predicate Abstraction). EPA genera automáticamente todas las combinaciones posibles de valores booleanos para los predicados seleccionados del modelo Alloy. Cada combinación define una partición del espacio de estados concretos. Si se utilizan n predicados, EPA produce 2^n particiones. Esta fase es responsable de cubrir de manera sistemática las distintas configuraciones semánticas relevantes del contrato.

Nota sobre restricciones. En algunos modelos, los predicados utilizados por EPA pueden incluir condiciones contextuales (*constraints*), por ejemplo restricciones de roles o valores específicos de estado. Aunque esto afecta el espacio de particiones, no constituye una fase adicional del procedimiento, sino una variante del uso de EPA.

BEPA (Bounded Exhaustive Predicate Abstraction). A partir de las particiones generadas por EPA, la fase BEPA evalúa de manera exhaustiva qué transiciones se verifican en *todas* las abstracciones. Las transiciones que se cumplen en todas las particiones se clasifican como *must*, mientras que aquellas que ocurren solo en algunas se clasifican como *may*. Esta fase es la más costosa computacionalmente, ya que requiere analizar todas las combinaciones generadas.

SBEPA (Strong BEPA o Hyper-Must). La fase SBEPA refina aún más el conjunto de transiciones *must*. Su objetivo es identificar transiciones *hyper-must*, es decir, transiciones que permanecen obligatorias incluso bajo condiciones más estrictas. Este refinamiento permite distinguir niveles de fortaleza dentro del conjunto de transiciones garantizadas.

Estas tres fases en conjunto permiten a Alloy4PA producir un grafo modal donde cada transición entre estados abstractos está etiquetada como *may*, *must* o *hyper-must*, reflejando el comportamiento posible, garantizado y fuertemente garantizado del contrato inteligente.

2.6. Resumen y análisis del enfoque de abstracciones modales

2.6.1. Ejemplo motivador: contrato de subasta

Para ilustrar la utilidad del enfoque de abstracciones modales, los autores de [2] presentan un contrato de subasta (*Auction*) escrito en *Solidity*. El contrato permite que los usuarios realicen ofertas (*bids*) durante un período determinado. Una vez finalizado dicho período, la subasta se cierra, el beneficiario puede retirar la oferta ganadora descontando una tarifa (*fee*), el propietario del contrato puede retirar la tarifa acumulada y los usuarios no ganadores pueden retirar sus depósitos.

El caso de estudio incluye intencionalmente un defecto en la implementación: el beneficiario recibe la oferta ganadora completa, sin descontar la tarifa correspondiente. Este error sirve para demostrar cómo las abstracciones modales pueden poner en evidencia fallas de lógica de negocio que no son detectadas mediante análisis estructurales o estáticos tradicionales.

2.6.2. Abstracciones may y must

A partir del contrato de subasta, se construye un *Labeled Transition System* (LTS) que representa su comportamiento abstracto. Cada estado abstracto describe las acciones disponibles en una etapa determinada del contrato, como `bid`, `withdraw` o `auctionEnd`. Asimismo, se incluye una transición especial τ (tau), que representa el paso del tiempo o eventos externos de la *blockchain* ajenos al contrato.

En las abstracciones iniciales, denominadas *may abstractions*, una transición etiquetada con una acción f entre dos estados abstractos s_1 y s_2 indica que existe al menos un estado concreto dentro de s_1 desde el cual es posible ejecutar f con ciertos parámetros que conducen al estado s_2. Esto proporciona información valiosa al auditor sobre los posibles comportamientos del contrato, aunque presenta una limitación fundamental: el modelo no garantiza que esas acciones sean siempre posibles.

En el ejemplo de la subasta, las abstracciones *may* no diferencian entre la versión correcta y la versión defectuosa del contrato. El modelo resultante parece idéntico en

ambos casos, ocultando el bug que causa que el beneficiario reciba un monto mayor al esperado.

Esta limitación motiva la introducción de un segundo tipo de transición, la denominada *must*. Mientras que las transiciones *may* indican acciones que pueden ser posibles en algunos estados concretos, las transiciones *must* representan acciones que deben ser posibles en todos los estados concretos representados por un estado abstracto.

De esta forma, si una acción aparece sólo como *may*, significa que existen escenarios en los cuales dicha acción no puede ejecutarse. En el contrato de subasta, esto permite evidenciar que la transición `auctionEnd` (cierre de subasta) no siempre puede realizarse correctamente, revelando el bug en la versión defectuosa del contrato.

Sin embargo, la existencia de transiciones *may* no implica necesariamente la presencia de un error. Existen situaciones en las que una transición opcional corresponde al comportamiento esperado del sistema. Por ejemplo, si un usuario realiza una oferta con el valor máximo permitido (`MAX_INT`), es natural que no se admitan nuevas ofertas, ya que el contrato debe impedir sobrepasar dicho límite.

Además, los autores definen un subtipo denominado *hyper-must*, que se aplica cuando una misma acción puede tener múltiples resultados posibles según el estado concreto, pero siempre es ejecutable con éxito desde cualquier configuración representada por el estado abstracto.

2.6.3. Transiciones con restricción

Otro aporte relevante del trabajo es la incorporación del concepto de *constraint transitions*. Estas transiciones permiten restringir la cuantificación de los parámetros a valores definidos por el auditor, con el fin de capturar comportamientos dependientes de roles o usuarios específicos.

Por ejemplo, al aplicar una restricción del tipo `msg.sender = 0xFA0...`, la abstracción resultante puede mostrar que un postor determinado tiene garantizado el derecho a retirar sus fondos si pierde la subasta. Este mecanismo facilita validar políticas de control de acceso o roles, como verificar que sólo el propietario puede finalizar la subasta o que únicamente el beneficiario puede retirar la oferta ganadora.

Las restricciones son especialmente útiles en auditorías donde la documentación funcional especifica distintos permisos o capacidades según el rol del participante.

2.6.4. Formalización del modelo y diseño del prototipo

El trabajo define formalmente un sistema de transición modal extendido para representar los contratos inteligentes, con estados abstractos derivados de predicados y transiciones clasificadas como *may*, *must* o *constraint*.

Para construir dichas abstracciones, se extiende la herramienta semiautomática presentada en [1]. Esta herramienta, desarrollada sobre *Alloy* [5] y su analizador, recibe como entrada:

1. una especificación en *Alloy* derivada del contrato original en *Solidity*;
2. un conjunto de predicados de abstracción definidos manualmente; y
3. opcionalmente, restricciones que limitan los parámetros de ciertas funciones.

El analizador de *Alloy* no permite expresar directamente el operador *must*. Para simularlo, se invierte la fórmula correspondiente y se prueba su insatisfacibilidad hasta cierta

cota. Si la fórmula negada resulta insatisfactible, la transición se clasifica como *must*. En el caso de transiciones con múltiples posibles destinos, se evalúa si cumplen las condiciones de *hyper-must*.

El proceso implementado consta de tres etapas:

1. Construcción del LTS inicial, asignando por defecto la etiqueta *may* a todas las transiciones.
2. Evaluación individual de cada transición para determinar si puede clasificarse como *must*.
3. Análisis adicional para identificar transiciones *hyper-must* con múltiples resultados posibles.

Aunque la implementación inicial no está optimizada para desempeño, los autores mencionan la posibilidad de aplicar mejoras en etapas posteriores.

2.6.5. Evaluación experimental

La evaluación experimental del trabajo se diseña en torno a tres preguntas principales:

RQ1: ¿Qué tan prevalentes son las transiciones *must* en los contratos inteligentes?

RQ2: ¿El uso de restricciones por roles mejora la capacidad de validación de los contratos?

RQ3: ¿Los auditores pueden detectar nuevos comportamientos problemáticos al diferenciar entre transiciones *may* y *must*?

Para responderlas, se utilizan dos *benchmarks* establecidos:

- **B1:** *Microsoft Azure Blockchain Workbench*, que incluye diagramas formales y datos de roles;
- **B2:** un conjunto de contratos previamente analizados en [1].

Cada caso se analiza construyendo las abstracciones modales correspondientes y, en algunos escenarios, aplicando restricciones de roles para evaluar la RQ2. Para la RQ3, se llevó a cabo una inspección manual independiente realizada por un auditor externo y uno de los autores.

2.6.6. Resultados detallados

Los resultados del estudio empírico reportado en [2] se organizaron en función de las tres preguntas de investigación propuestas, RQ1, RQ2 y RQ3.

RQ1 — Prevalencia de las transiciones *must*

Las transiciones *must* estuvieron presentes en todos los contratos analizados. Algunos tipos de transición, como las de tiempo y las asociadas al constructor, resultaron ser siempre *must*. En ciertos contratos, estas fueron las únicas transiciones obligatorias, mientras que en otros casos más de la mitad de las transiciones totales pertenecían a esta categoría.

Aproximadamente la mitad de los casos analizados contenía al menos una transición *hyper-must*.

En conjunto, los resultados sugieren que las transiciones *must* son altamente prevalentes en contratos inteligentes, y que su identificación permite distinguir con precisión las operaciones garantizadas de las opcionales dentro del flujo del contrato.

RQ2 — Validación con restricciones de roles

Para evaluar el impacto de las restricciones de roles en la validación, los autores compararon la cantidad de transiciones esperadas para cada rol según la documentación (#Exp) con la cantidad de transiciones *must* observadas con la restricción de que el rol sea ese (#M). Las interpretaciones posibles son las siguientes:

- Si #Exp = #M, el comportamiento del contrato coincide con lo esperado.
- Si #Exp > #M, existe al menos una acción esperada que no es posible ejecutar según la implementación.
- Si #Exp < #M, el contrato permite acciones no contempladas o la documentación omite restricciones relevantes.

De los nueve casos de prueba, en tres se detectaron discrepancias significativas:

- **DigitalLocker**: se observó #Exp > #M. Se recomienda analizar con los desarrolladores si el contrato está restringiendo indebidamente ciertas acciones o si la documentación es incompleta.
- **RefrigeratedTransp**: se observó #Exp < #M. El contrato permite operaciones adicionales no especificadas formalmente.
- **AssetTransfer**: se observó #Exp < #M, pero la causa fue un error en el modelo *Alloy* generado, no en el contrato *Solidity*.

En conclusión, el uso de abstracciones modales con restricciones permitió descubrir dos defectos no reportados previamente en los contratos *DigitalLocker* y *RefrigeratedTransp*, además de identificar un error en la especificación del caso *AssetTransfer*.

RQ3 — Análisis manual de las abstracciones

Finalmente, para evaluar la utilidad práctica del enfoque, un auditor externo y uno de los autores inspeccionaron manualmente las abstracciones generadas, concentrándose en las transiciones que no eran *must*. Ambos detectaron comportamientos inesperados en dos contratos:

- **EscrowVault**: el proceso puede iniciarse con un balance igual a cero, lo que impide realizar retiros posteriores de fondos. Este comportamiento es una anomalía no contemplada en la especificación.
- **ValidatorAuction**: la subasta puede comenzar sin usuarios registrados en la *allow-list*, impidiendo cualquier oferta y dejando el contrato en un estado no funcional.

Estos hallazgos muestran que la distinción entre transiciones *may* y *must* ayuda a los auditores a detectar anomalías de comportamiento que no pueden observarse mediante abstracciones puramente *may*. En ambos casos, el análisis manual complementó eficazmente los resultados automáticos, destacando la utilidad del enfoque en auditorías asistidas.

2.6.7. Amenazas a la validez

Los autores reconocen posibles amenazas a la validez de su estudio, tanto internas como externas. Entre las internas se encuentran:

- posibles errores manuales en la traducción del código *Solidity* a *Alloy*;

- limitaciones derivadas del uso de cotas finitas en el análisis de satisfacibilidad; y
- la fuerte dependencia del conjunto de predicados elegidos para definir la abstracción.

Entre las amenazas externas, se destacan:

- el uso de *benchmarks* experimentales que pueden no representar todos los escenarios reales de producción; y
- la evaluación empírica basada en un número limitado de auditores (uno de ellos coautor del trabajo).

Los autores mitigaron estos riesgos publicando todos los modelos y resultados generados, y sugirieron realizar futuros estudios con un conjunto más amplio de contratos y evaluadores independientes.

2.6.8. Trabajos relacionados

El trabajo de [2] se apoya directamente en el enfoque de abstracción por predicados propuesto en [1], extendiéndolo mediante la incorporación de transiciones modales y restricciones sobre parámetros.

Además, los autores mencionan diversas líneas de investigación complementarias. Algunos trabajos se centran en la generación automática de casos de prueba para contratos inteligentes, con el objetivo de explorar sistemáticamente las posibles ejecuciones del código. Otros enfoques transforman los contratos en redes de Petri, permitiendo validar propiedades de comportamiento de manera automática mediante modelos de concurrencia bien establecidos. También existen herramientas orientadas al análisis estático del código fuente, que buscan detectar vulnerabilidades comunes (como desbordamientos, condiciones de carrera o errores de reentrada) sin necesidad de ejecutar el contrato.

Estas aproximaciones comparten el objetivo general de aumentar la confiabilidad de los contratos inteligentes, aunque difieren en el nivel de abstracción, el grado de automatización y el tipo de propiedades que pueden verificar.

A diferencia de estos métodos, que buscan la verificación completamente automática del contrato mediante técnicas de *model checking* o análisis estático, el enfoque de abstracciones modales propuesto por [2] se orienta a asistir al auditor humano, brindándole modelos abstractos que facilitan la inspección, la interpretación y la detección de comportamientos lógicos no deseados.

2.6.9. Conclusiones del enfoque modal

El trabajo de [2] demuestra que las abstracciones modales constituyen una herramienta efectiva para apoyar la validación de contratos inteligentes. El uso conjunto de transiciones *may*, *must* y *constraint* permite representar con mayor fidelidad el comportamiento del contrato, identificar errores de lógica de negocio y validar los requisitos asociados a roles y permisos.

El enfoque se apoya en la experiencia del auditor humano, quien define los predicados y restricciones, y en la capacidad de razonamiento automático de *Alloy* para construir y analizar las abstracciones resultantes. La evaluación empírica mostró que estas técnicas no solo reproducen correctamente los comportamientos esperados, sino que además ayudan a detectar defectos previamente no reportados en casos de estudio conocidos.

A futuro, se proyecta automatizar la generación de abstracciones a partir del código fuente en *Solidity* y la extracción automática de predicados, con el objetivo de reducir la

intervención manual y mejorar la escalabilidad del enfoque.

En el siguiente capítulo se describe la metodología empleada para reproducir experimentalmente el enfoque presentado en [2] mediante la herramienta *Alloy4PA*.

3. METODOLOGÍA DE REPRODUCCIÓN

En esta sección se detalla la metodología que se seguirá para reproducir los experimentos presentados en el paper "Abstracciones Modales para Contratos Inteligentes"[2]. El objetivo es garantizar la replicabilidad de los resultados utilizando la herramienta Alloy4PA[3] en diferentes entornos y versiones.

3.1. Preliminares y preparación del entorno

En primer lugar, se procederá a la lectura del paper para comprender el contexto y los objetivos experimentales. También se revisará el archivo `README` del repositorio oficial del proyecto Alloy4PA, para entender el uso de la herramienta y los requisitos necesarios para su ejecución. Se documentarán las versiones de software y dependencias requeridas.

Para comprobar la correcta ejecución de la herramienta, se realizarán pruebas en los sistemas operativos Windows y MacOS. Para comenzar, se clonará el repositorio del proyecto y se leerá el `README` para identificar los requisitos específicos de instalación. Una vez verificados, se instalarán las dependencias necesarias en cada sistema.

3.2. Ejecución básica de la herramienta Alloy4PA

A continuación, se evaluarán las distintas modalidades de ejecución de Alloy4PA, siguiendo las instrucciones provistas en el `README` del repositorio. Se hará una corrida de cada modalidad en ambos sistemas operativos, con el input más simple posible. En cada caso se documentarán los pasos seguidos, así como cualquier inconveniente encontrado.

3.2.1. Ejecución mediante JAR precompilado

En primer lugar, se probará la ejecución mediante el archivo `JAR` precompilado. Para ello, se instalará la versión de Java recomendada y se ejecutará el `JAR` utilizando el ejemplo simple provisto en el `README`. Se validará que el resultado sea exitoso.

3.2.2. Ejecución compilando desde fuentes

Luego, se procederá a compilar Alloy4PA desde las fuentes. Se instalará Maven y se compilarán las fuentes siguiendo las instrucciones del repositorio. Una vez generado el JAR, se ejecutará con el mismo ejemplo y se validará que el resultado coincida con el anterior.

3.2.3. Ejecución mediante imagen de Docker

También se evaluará la ejecución de Alloy4PA mediante una imagen de Docker provista por el proyecto. Se instalará Docker en el sistema y se ejecutará la imagen oficial utilizando el mismo ejemplo simple. Se comprobará que el resultado sea consistente con las ejecuciones previas.

3.2.4. Ejecución generando imagen de Docker propia

Finalmente, se generará una imagen de Docker propia a partir de las fuentes del proyecto, siguiendo las instrucciones del `README`. Se ejecutará la imagen generada con el ejemplo simple y se validará que el resultado sea el mismo.

3.3. Ejecución completa de la herramienta

Una vez probadas todas las modalidades de ejecución, y validado que son equivalentes, se realizará una ejecución completa usando una de ellas. Se procederá a ejecutar todos los casos de los benchmarks y se validarán los resultados en función de las preguntas de investigación (RQ) planteadas en el paper.

Se instalará Python y se ejecutarán los scripts requeridos para el análisis de los RQ, siguiendo las indicaciones provistas por los autores del enfoque.

3.4. Outputs de la ejecución completa de la herramienta

Los outputs generados durante nuestra reproducción de los benchmarks del trabajo original[2] se organizaron en un repositorio público de GitHub [4], con el fin de facilitar su consulta, análisis y verificación independiente.

4. REPRODUCCIÓN

4.1. Verificación de Prerrequisitos

Antes de instalar y usar Alloy4PA, se recopiló el listado de todos los requisitos necesarios, según el `README` del repositorio de la herramienta [3]. Se verificó que se cumplieran los requisitos de hardware y sistema operativo en cada máquina usada para la reproducción. Los requisitos de software se verificarán en el transcurso de las reproducciones.

Componente	**Versión recomendada**
Hardware	Procesador: 2 núcleos o más; RAM: 4GB o mayor; Espacio en disco: 1GB libre
Sistema operativo	Linux, macOS o Windows
Git	Para clonar el repositorio
JDK / JAVA	Java 18+
Maven (opcional)	Maven 3.6+ (Si se sigue la versión de compilación de los fuentes. Todas las dependencias están especificadas en `Alloy4PA/pom.xml` y resueltas automáticamente por Maven.)
Docker (opcional)	Si se usa la versión contenedorizada según instrucciones del repositorio.
Python	Python 3.9+

Tab. 4.1: Prerrequisitos para Alloy4PA

4.2. Ejecución básica en Windows

A continuación se realizó una ejecución básica de la herramienta Alloy4PA en sus distintas modalidades.

Esta reproducción se realizó en una Notebook MSI Katana AI Ryzen 9 (con procesador AMD Ryzen 9 8945HS, 16 GB RAM y placa NVIDIA GeForce RTX4070), sobre sistema operativo Windows 11 Home de 64 bits.

4.2.1. Inicialización del entorno

Se accedió al repositorio de GitHub que almacena el código de la herramienta [3], y se obtuvo la URL para clonar el repositorio localmente.

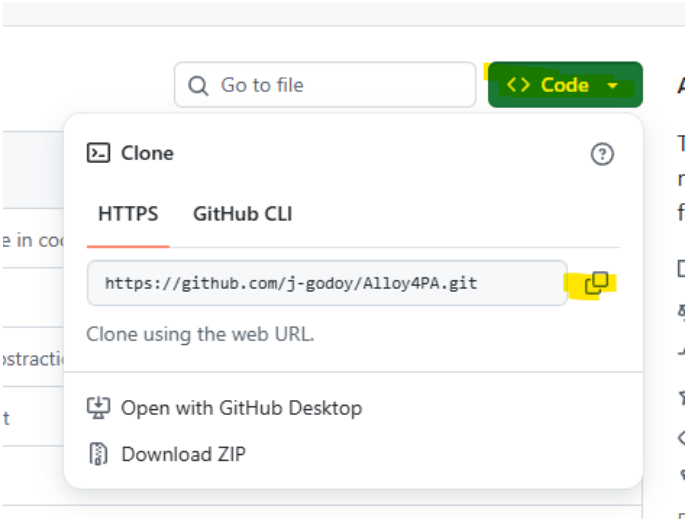
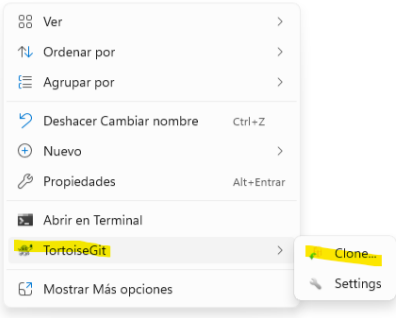
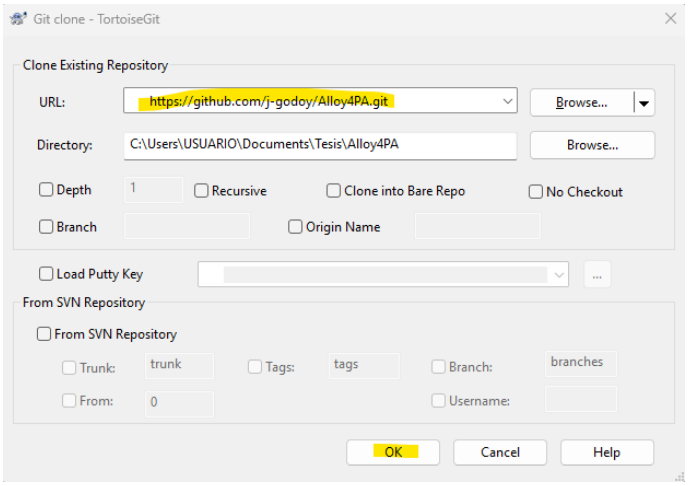


Fig. 4.1: Repositorio de GitHub de Alloy4PA

Se tenía ya previamente instalado TortoiseGit y se utilizó el mismo para clonar el repositorio. Para ello se navegó hasta el directorio donde se quería clonar el repositorio y se siguieron los siguientes pasos:



(a) Menú contextual



(b) Clonación del repositorio

Fig. 4.2: Uso de TortoiseGit

Con esto se obtuvo la copia del repositorio localmente. Desde allí se accedió a la terminal en el mismo directorio:

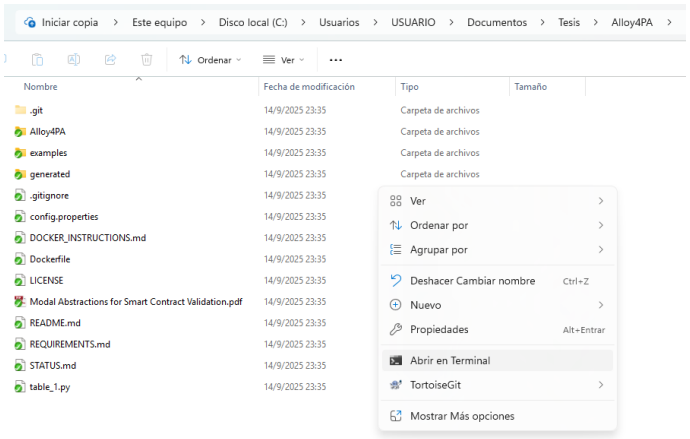


Fig. 4.3: Terminal en el directorio del repositorio clonado

A continuación se probó correr la herramienta bajo las cuatro modalidades explicadas en el repositorio.

4.2.2. Modalidad JAR precompilado

4.2.2.1. Instalación de prerrequisitos

Se verificó qué versión de JAVA estaba instalada:

```bash
> java --version
```

El resultado fue que se tenía instalado el jdk-17. Por lo cual se procedió a descargar la jdk-18 de la página oficial [11] (sin desinstalar la jdk-17 dado que era necesaria para otro proyecto en la máquina):

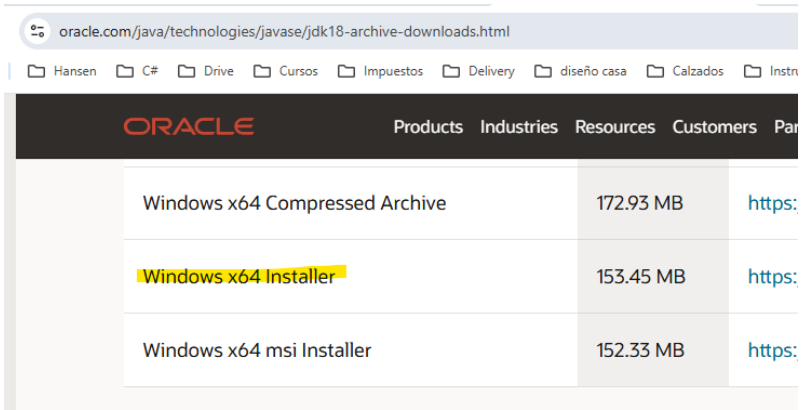
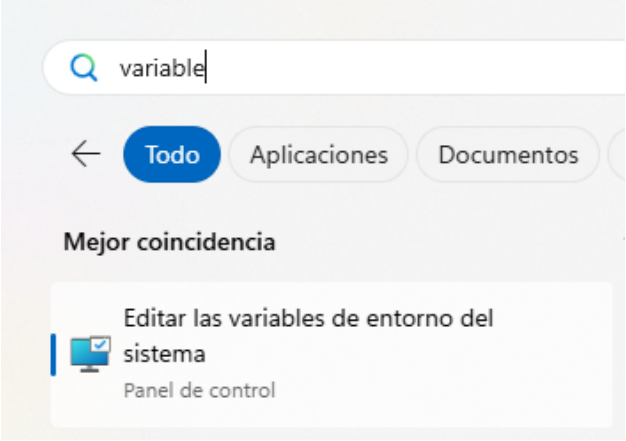
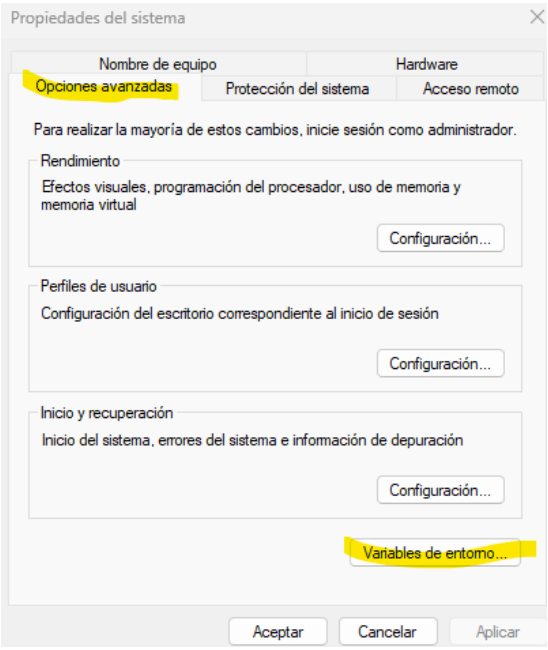


Fig. 4.4: Descarga de JDK 18

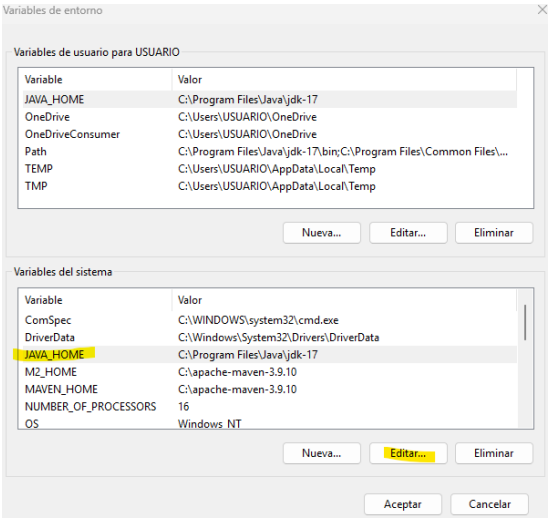
Se siguieron los pasos del instalador y luego se modificaron las variables de entorno `JAVA_HOME` y `Path` (de acuerdo al directorio de instalación correspondiente en la máquina) para retirar de las mismas las referencias a la jdk-17:



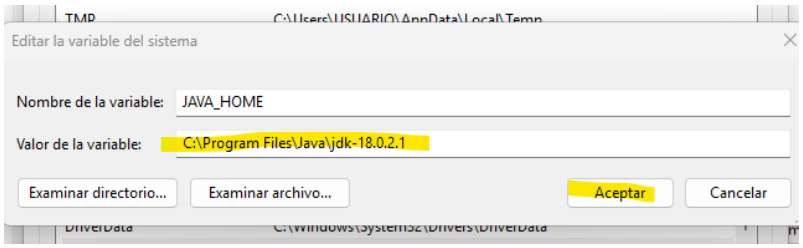
(a) Acceso a las variables de entorno (para modificar `JAVA_HOME` y Path)



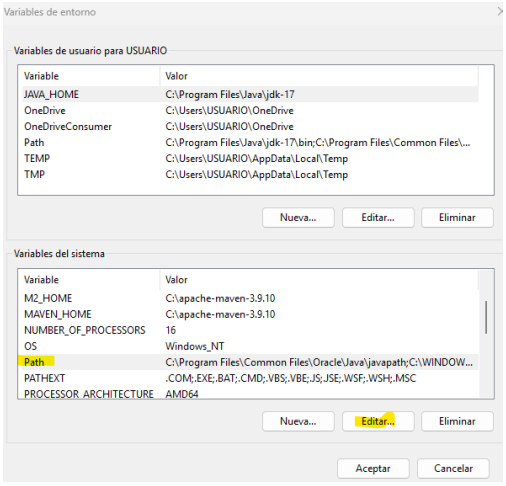
(b) Edición de variables de entorno



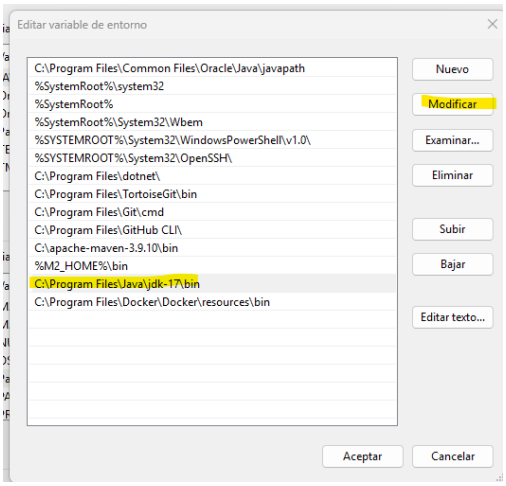
(c) Selección de `JAVA_HOME` (directorio de instalación)



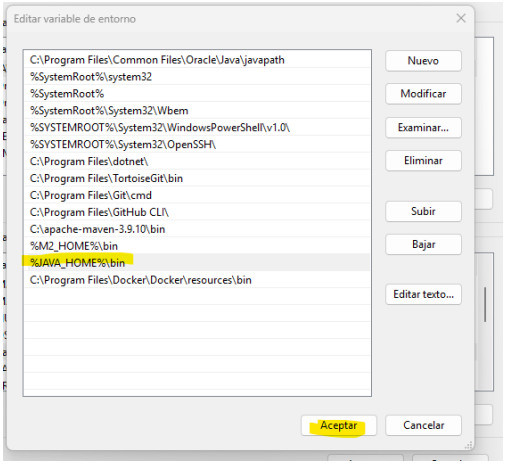
(d) Edición y confirmación de cambios en `JAVA_HOME`



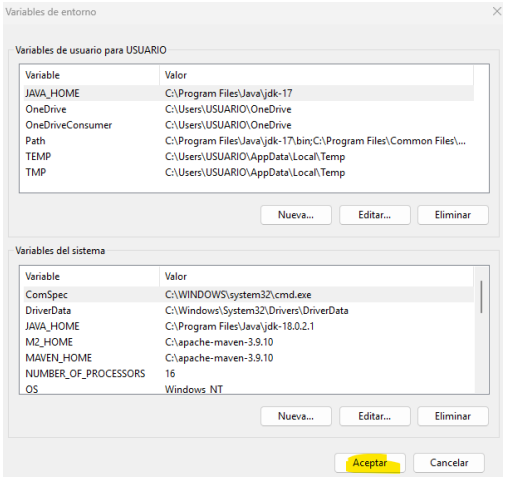
(e) Selección de Path (directorio de fuentes en el listado de paths)



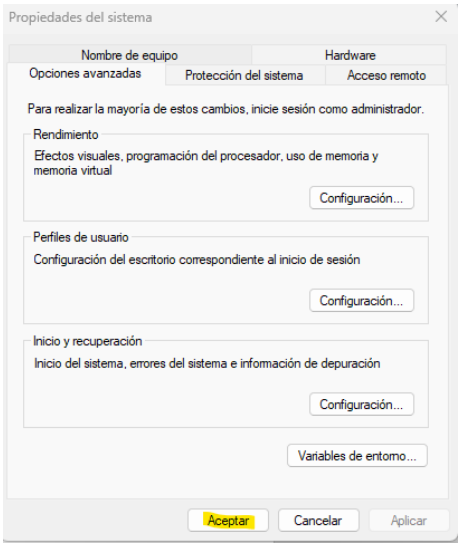
(f) Verificación de los valores de la variable Path



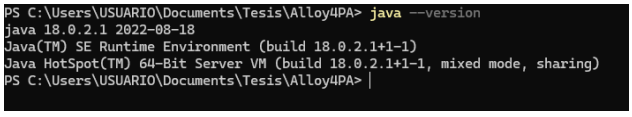
(g) Confirmación de modificación o agregado del path



(h) Confirmación de las modificaciones



(i) Aceptación de las modificaciones



(j) Entorno listo para uso de Alloy4PA

Fig. 4.5: Modificación y verificación de variables de entorno para usar JDK 18

4.2.2.2. Prueba de ejecución de Alloy4PA sobre Benchmark B1

Como primera etapa de validación, se ejecutaron algunos de los casos correspondientes al *benchmark* **B1**, entre ellos *BasicProvenance*, *DigitalLocker* y *AssetTransfer*, los cuales corrieron sin errores.

A continuación se presenta un ejemplo representativo:

bash

```
> java -jar Alloy4PA/target/Alloy4PA-1.0.0-fat-jar-with-dependencies.jar examples/Benchmarks/B1/alloy_models/AssetTransfer.als
========================================= Subjects to generate abstraction: =========================================
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\AssetTransfer.als
========================================= STARTING WITH FILE: C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\AssetTransfer.als
=============================================================
Configuration loaded successfully from config.properties
Max Thread Number: 4
Query Timeout Limit (secs): 600
Verbose: true
Avoid Must Constructor: false
Avoid Must Tau: false
Avoid Must All: false
Allow Address 0x0: true
Output Statistics Path: results/
Overwrite Subjects: false
=============================================================
Loading config C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\AssetTransferConfig.ini
Total time generating alloy file with partitions and preds: 0,00 min
========== Parsing+Typechecking C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\AssetTransfer\AssetTransfer_part3.als
==============
... Subject,EPA (secs), BEPA (secs), SBEPA
AssetTransfer,7.988,150.954,0.125
Total time all subjects (seconds): 159
End.
>
```

El proceso guardó los resultados en el directorio:
`C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1`,
en este caso en la subcarpeta AssetTransfer, y el diagrama que aplica las transacciones modales es el llamado `AssetTransferforked_arrows.dot`.

Graphviz DOT

```
digraph {
 S00->S01 [label="constructor", style="dotted", color="blue"]
 S02->S10 [label="terminate", style="dotted", color="blue"]
 S02->S01 [label="reject\nrescindOffer", style="dotted", color="blue"]
 S02->S02 [label="modifyOffer", style="dotted", color="blue"]
 S02->S03 [label="acceptOffer", style="dotted", color="blue"]
 S01->S02 [label="makeOffer", style="dotted", color="blue"]
```

```
  S01->S01 [label="modify", style="dotted", color="blue"]
  S01->S10 [label="terminate", style="dotted", color="blue"]
  S04->S10 [label="terminate", style="dotted", color="blue"]
  S04->S01 [label="reject\nrescindOffer", style="dotted", color="blue"]
  S04->S06 [label="markAppraised", style="dotted", color="blue"]
  S03->S10 [label="terminate", style="dotted", color="blue"]
  S03->S04 [label="markInspected", style="dotted", color="blue"]
  S03->S01 [label="reject\nrescindOffer", style="dotted", color="blue"]
  S03->S05 [label="markAppraised", style="dotted", color="blue"]
  S06->S08 [label="accept", style="dotted", color="blue"]
  S06->S07 [label="accept", style="dotted", color="blue"]
  S06->S10 [label="terminate", style="dotted", color="blue"]
  S06->S01 [label="reject\nrescindOffer", style="dotted", color="blue"]
  S05->S10 [label="terminate", style="dotted", color="blue"]
  S05->S06 [label="markInspected", style="dotted", color="blue"]
  S05->S01 [label="reject\nrescindOffer", style="dotted", color="blue"]
  S08->S09 [label="accept", style="dotted", color="blue"]
  S08->S10 [label="terminate", style="dotted", color="blue"]
  S08->S01 [label="reject", style="dotted", color="blue"]
  S07->S09 [label="accept", style="dotted", color="blue"]
  S07->S01 [label="rescindOffer", style="dotted", color="blue"]
}
```

Pegando el código del diagrama en la web [12] se genera el siguiente diagrama de transición de estados:

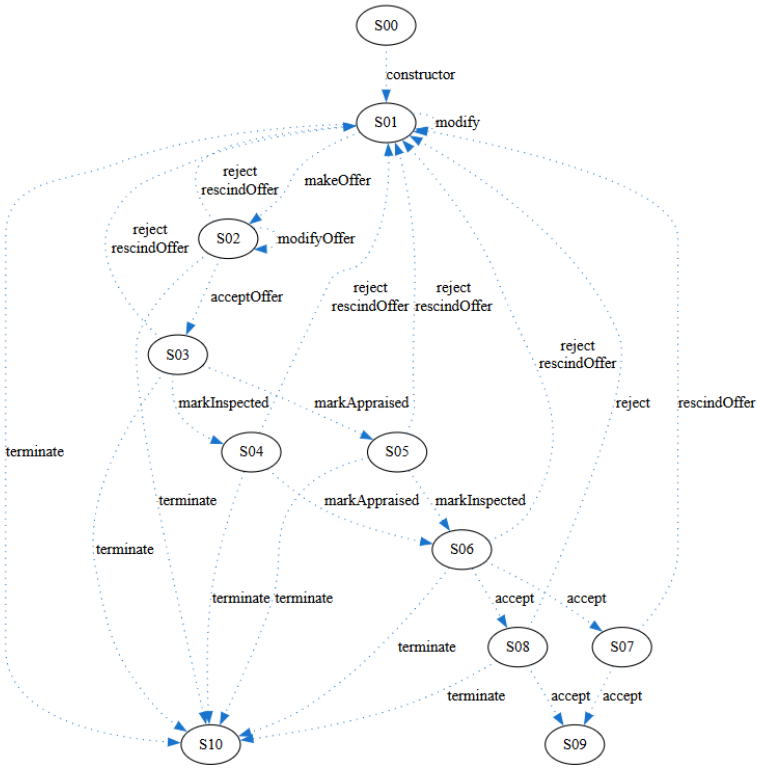


Fig. 4.6: Diagrama de transición de estados generado por Graphviz

4.2.3. Compilar archivo JAR desde el código fuente

4.2.3.1. Instalación de prerrequisitos

Siendo que la versión de Java ya era la correcta por la prueba anterior, se procedió a verificar la versión de Maven instalada, la cual ya cumplía los requisitos.

```
C:\Users\USUARIO>mvn --version
Apache Maven 3.9.10 (5f519b97e944483d878815739f519b2eade0a91d)
Maven home: C:\apache-maven-3.9.10
Java version: 17.0.6, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-17
Default locale: es_AR, platform encoding: Cp1252
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
```

Fig. 4.7: Verificación de versión de Maven instalada

También se verificó que las variables de entorno de Maven estuviesen bien configuradas (se presentan solo las capturas de los valores dado que ya se explicó previamente en 1.a cómo acceder a dichas variables):

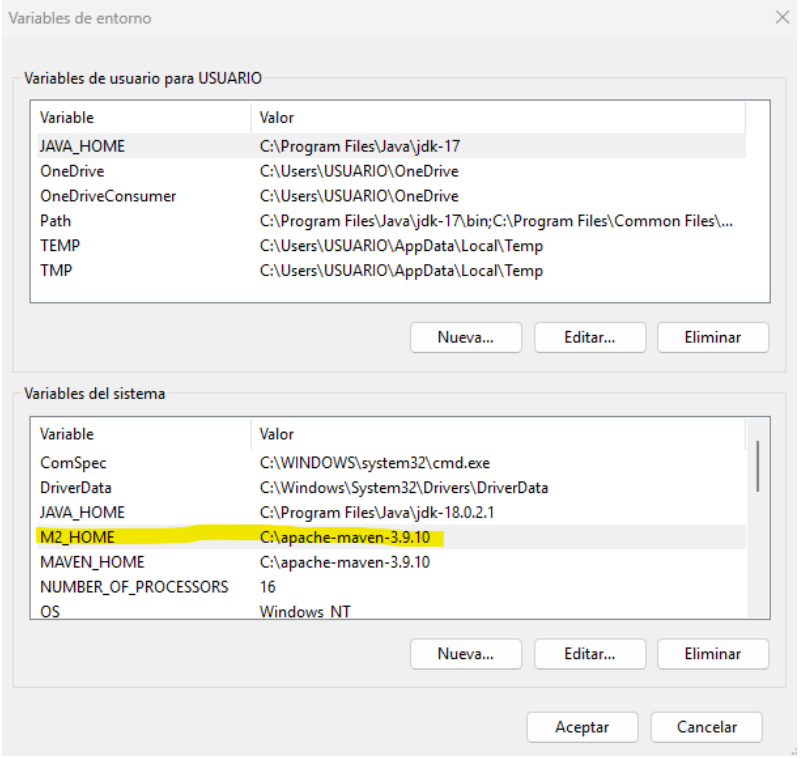


Fig. 4.8: Selección de la variable de entorno `MAVEN_HOME` (modificarla si no está correcto el path de instalación)

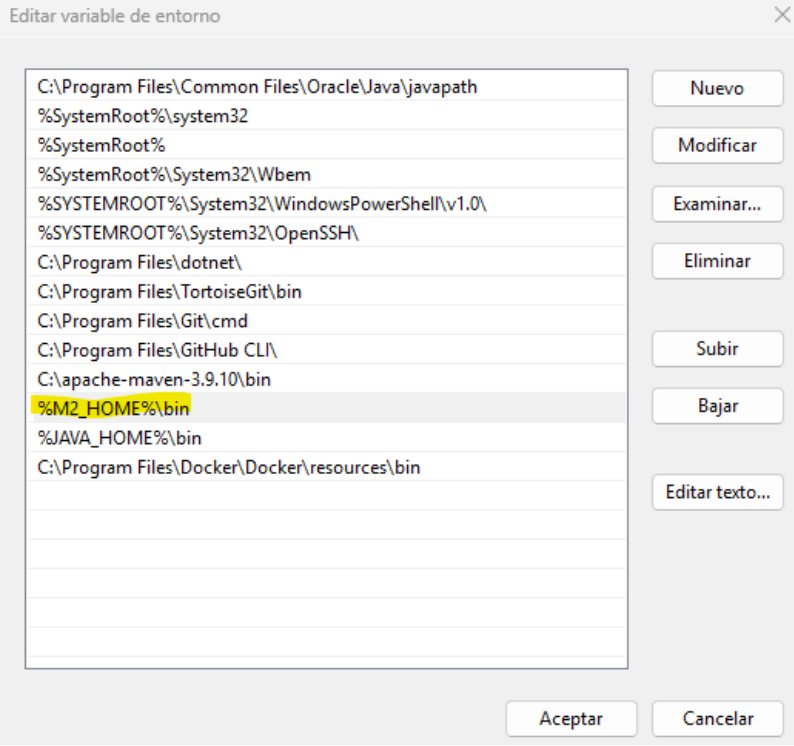


Fig. 4.9: Verificar si está presente la línea para Maven en el path (agregarla o modificarla de acuerdo al path de los fuentes)

Tener en cuenta que **Maven** es una herramienta de automatización de proyectos de software, utilizada principalmente en el ecosistema Java. Maven permite gestionar la compilación, el empaquetado y el despliegue de aplicaciones, así como administrar y automatizar la descarga de sus dependencias externas mediante un modelo declarativo basado en archivos XML (`pom.xml`). Su uso estandariza el ciclo de vida de construcción y facilita la reproducibilidad de los proyectos en diferentes entornos.

4.2.3.2. Compilado de las fuentes

Se accedió al subdirectorio también llamado Alloy4PA y desde allí se ejecutó el comando listado a continuación:

PowerShell

```
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA> cd .\Alloy4PA\
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA\Alloy4PA> mvn install:install-file -Dfile
    ↪="libs/org.alloytools.alloy6.dist.jar" -DgroupId="org.alloytools.alloy6.dist" -
    ↪DartifactId="alloy" -Dversion="6.0.0" -Dpackaging="jar"
[INFO] Scanning for projects...
[INFO]
[INFO] -------------------------< ar.uba.dc:Alloy4PA >-------------------------
[INFO] Building Alloy4PA 1.0.0
[INFO]  from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- install:3.1.2:install-file (default-cli) @ Alloy4PA ---
```

```
[INFO] Installing C:\Users\USUARIO\Documents\Tesis\Alloy4PA\Alloy4PA\libs\org.alloytools.alloy6.dist.jar to C:\Users\USUARIO\.m2\repository\org\alloytools\alloy6\dist\alloy\6.0.0\alloy-6.0.0.jar
[INFO] Installing C:\Users\USUARIO\AppData\Local\Temp\org.alloytools.alloy6.dist15153409767529295471.pom to C:\Users\USUARIO\.m2\repository\org\alloytools\alloy6\dist\alloy\6.0.0\alloy-6.0.0.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.488 s
[INFO] Finished at: 2025-09-15T01:30:43-03:00
[INFO] ------------------------------------------------------------------------
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA\Alloy4PA> mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] -------------------------< ar.uba.dc:Alloy4PA >-------------------------
[INFO] Building Alloy4PA 1.0.0
[INFO]  from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- clean:3.2.0:clean (default-clean) @ Alloy4PA ---
[INFO] Deleting C:\Users\USUARIO\Documents\Tesis\Alloy4PA\Alloy4PA\target
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ Alloy4PA ---
[INFO] skip non existing resourceDirectory C:\Users\USUARIO\Documents\Tesis\Alloy4PA\Alloy4PA\src\main\resources
[INFO]
[INFO] --- compiler:3.8.1:compile (default-compile) @ Alloy4PA ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 20 source files to C:\Users\USUARIO\Documents\Tesis\Alloy4PA\Alloy4PA\target\classes
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ Alloy4PA ---
[INFO] skip non existing resourceDirectory C:\Users\USUARIO\Documents\Tesis\Alloy4PA\Alloy4PA\src\test\resources
[INFO]
[INFO] --- compiler:3.8.1:testCompile (default-testCompile) @ Alloy4PA ---
[INFO] No sources to compile
[INFO]
[INFO] --- surefire:3.2.5:test (default-test) @ Alloy4PA ---
[INFO] No tests to run.
[INFO]
[INFO] --- jar:3.4.1:jar (default-jar) @ Alloy4PA ---
[INFO] Building jar: C:\Users\USUARIO\Documents\Tesis\Alloy4PA\Alloy4PA\target\Alloy4PA-1.0.0.jar
[INFO]
[INFO] --- assembly:3.3.0:single (make-assembly) @ Alloy4PA ---
[WARNING] Parameter 'finalName' is read-only, must not be used in configuration
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/felix/maven-bundle-plugin/maven-metadata.xml
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/felix/maven-bundle-plugin/maven-metadata.xml (1.6 kB at 5.4 kB/s)
[INFO] Building jar: C:\Users\USUARIO\Documents\Tesis\Alloy4PA\Alloy4PA\target\Alloy4PA-1.0.0-fat-jar-with-dependencies.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

```
[INFO] Total time: 12.308 s
[INFO] Finished at: 2025-09-15T01:37:08-03:00
[INFO] ------------------------------------------------------------------------
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA\Alloy4PA>
```

La primera vez que se ejecutó, este comando descargó todas las dependencias, las cuales fueron bastantes y duró mayor tiempo que en el ejemplo listado arriba (que fue una segunda prueba realizada para esta documentación). Se aclara para referencia, que en la primera prueba, todas las descargas de dependencias se llevaron sin ningún inconveniente.

4.2.3.3. Prueba de compilado exitoso con un caso de B1

Volviendo al directorio principal se ejecutó un caso, el mismo que mencionan en el repositorio [3] como ejemplo rápido:

PowerShell

```
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA\Alloy4PA> cd ..
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA> java -jar Alloy4PA/target/Alloy4PA
    ↪-1.0.0-fat-jar-with-dependencies.jar examples/Benchmarks/B1/alloy_models/
    ↪BasicProvenance.als
========================================= Subjects to generate abstraction:
    ↪=========================================
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\
    ↪BasicProvenance.als
========================================= STARTING WITH FILE: C:\Users\USUARIO\
    ↪Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\BasicProvenance.als
================================================================
Configuration loaded successfully from config.properties
Max Thread Number: 4
Query Timeout Limit (secs): 600
Verbose: true
Avoid Must Constructor: false
Avoid Must Tau: false
Avoid Must All: false
Allow Address 0x0: true
Output Statistics Path: results/
Overwrite Subjects: false
================================================================
Loading config C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\
    ↪alloy_models\BasicProvenanceConfig.ini
Total time generating alloy file with partitions and preds: 0,00 min
========== Parsing+Typechecking C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\
    ↪Benchmarks\B1\BasicProvenance\BasicProvenance_part3.als
==============
========== Parsing+Typechecking C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\
    ↪Benchmarks\B1\BasicProvenance\BasicProvenance_part1.als
==============
========== Parsing+Typechecking C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\
    ↪Benchmarks\B1\BasicProvenance\BasicProvenance_part4.als
==============
========== Parsing+Typechecking C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\
    ↪Benchmarks\B1\BasicProvenance\BasicProvenance_part2.als
==============
```

```
28 ========== Parsing+Typechecking C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\
       ↪Benchmarks\B1\BasicProvenance\BasicProvenance.als
29 ==============
30 ========== Parsing+Typechecking C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\
       ↪Benchmarks\B1\BasicProvenance\BasicProvenance.als
31 ==============
32 Saving file C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\
       ↪BasicProvenance\BasicProvenanceforked_arrows.dot...
33 Hyper-MUST cost(seconds): 0
34 Subject,EPA (secs), BEPA (secs), SBEPA
35 BasicProvenance,0.591,0.669,0.06
36 Total time all subjects (seconds): 1
37 End.
38 PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA>
```

4.2.4. Modalidad generando una imagen de Docker

4.2.4.1. Instalación de prerrequisitos

Docker es una plataforma de virtualización ligera que permite empaquetar y ejecutar aplicaciones en contenedores. Un contenedor incluye todo lo necesario para que una aplicación se ejecute de forma consistente: código, librerías, dependencias y configuración del sistema. A diferencia de las máquinas virtuales tradicionales, los contenedores comparten el mismo núcleo del sistema operativo, lo que los hace más eficientes en recursos, portables y rápidos de iniciar.

Dado que la máquina no tenía instalado Docker, se procedió a descargar **Docker Desktop** para Windows desde la página oficial [13].

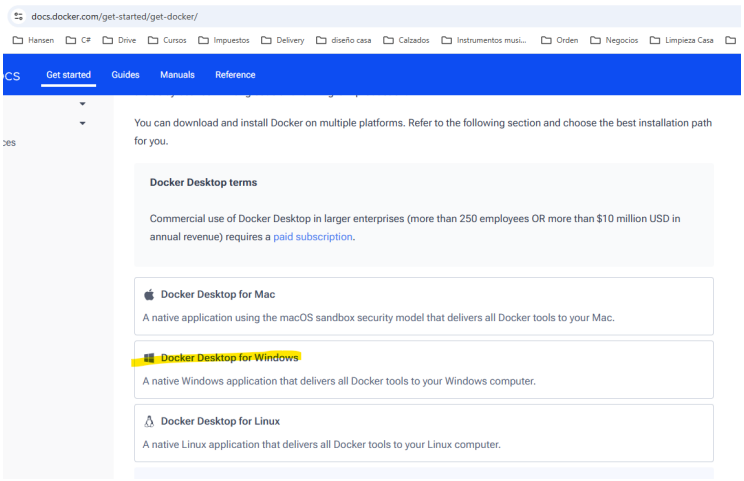


Fig. 4.10: Descarga de Docker Desktop para Windows

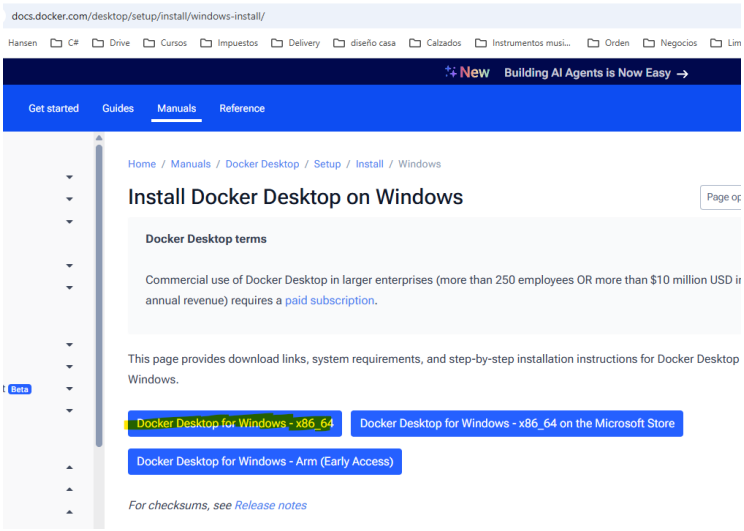


Fig. 4.11: Selección de instalador de Docker Desktop

Se ejecutó el instalador "Docker Desktop Installer.exe" y se siguieron las instrucciones, finalizando con un reinicio de la máquina.

A continuación se abrió Docker Desktop, y al abrirlo el mismo pidió actualizar la versión de WSL (Subsistema de Windows para Linux) con el comando:

bash

```
wsl --update
```

Manteniendo Docker Desktop en ejecución, se verificó la versión instalada y que estuviera correctamente instalado:

PowerShell

```
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA> docker --version
Docker version 28.4.0, build d8eb465
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:54e66cc1dd1fcb1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31
Status: Downloaded newer image for hello-world:latest
Hello from Docker! This message shows that your installation appears to be working
    ↪correctly.
To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub. (amd64)
 3. The Docker daemon created a new container from that image which runs the
     ↪executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it to your
     ↪ terminal.
To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash
Share images, automate workflows, and more with a free Docker ID: https://
```

```
    ↪hub.docker.com/
18  For more examples and ideas, visit: https://docs.docker.com/get-started/
19  PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA>
```

4.2.4.2. Creación de la imagen de Docker

En una consola, situados en el directorio principal del repositorio, se ejecutaron los siguientes comandos:

PowerShell

```
1  PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA> docker build -t alloy4pa-replication .
2  [+] Building 141.0s (17/17) FINISHED
3  ... (salida resumida)
```

Para crear la imagen, Docker descarga todas las dependencias informadas en el archivo `pom.xml` de Maven.

La imagen generada se llamará `alloy4pa-replication` (parámetro de `-t` que puede ser cambiado si se desea otro nombre). El "." indica que se tomará el directorio donde se ejecuta el comando como base para la imagen. En este directorio debe encontrarse un archivo llamado "Dockerfile" que contiene las configuraciones necesarias para la creación de la imagen, el cual ya estaba preconfigurado para arquitectura x86_64:

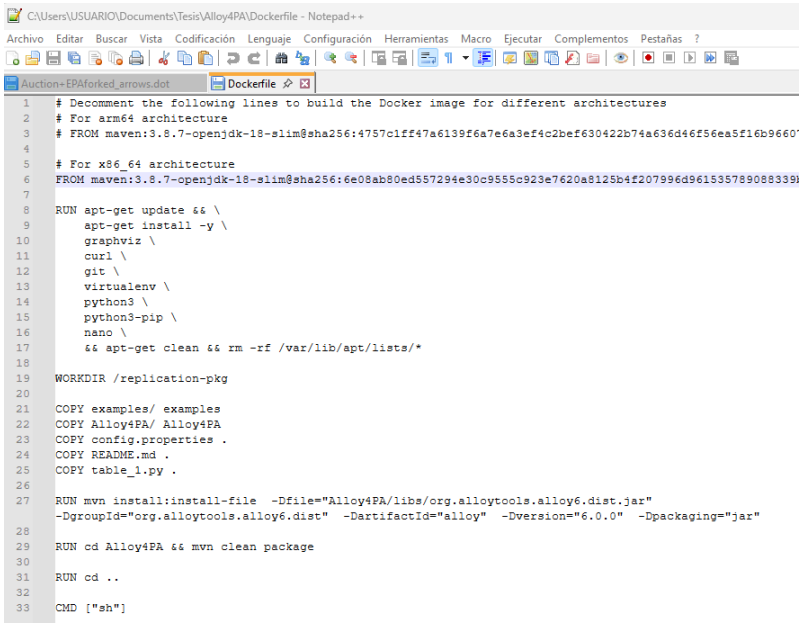


Fig. 4.12: Dockerfile preconfigurado para x86_64

4.2.4.3. Prueba de la imagen creada en Docker con un caso de B1

Como primer paso se accedió a la imagen recién creada para poder correr comandos sobre la misma, como la ejecución de la herramienta.

Fig. 4.13: Acceso a la imagen de Docker creada

Se llamó a Alloy4PA para un caso no ejecutado previamente fuera de la imagen (dado que, al crearla, ya se llevó los resultados de los mismos a la imagen), en este caso `DefectiveComponentCounter`:

bash

```
# java -jar Alloy4PA/target/Alloy4PA-1.0.0-fat-jar-with-dependencies.jar examples/Benchmarks/B1/alloy_models/DefectiveComponentCounter.als
========================================= Subjects to generate abstraction: =========================================
/replication-pkg/examples/Benchmarks/B1/alloy_models/DefectiveComponentCounter.als
========================================= STARTING WITH FILE: /replication-pkg/examples/Benchmarks/B1/alloy_models/DefectiveComponentCounter.als
=============================================================
Configuration loaded successfully from config.properties
Max Thread Number: 4
Query Timeout Limit (secs): 600
Verbose: true
Avoid Must Constructor: false
Avoid Must Tau: false
Avoid Must All: false
Allow Address 0x0: true
Output Statistics Path: results/
Overwrite Subjects: false
=============================================================
Loading config /replication-pkg/examples/Benchmarks/B1/alloy_models/DefectiveComponentCounterConfig.ini
...
========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/DefectiveComponentCounter/DefectiveComponentCounter.als
==============
========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/DefectiveComponentCounter/DefectiveComponentCounter.als
==============
Saving file /replication-pkg/examples/Benchmarks/B1/DefectiveComponentCounter/DefectiveComponentCounterforked_arrows.dot...
Hyper-MUST cost(seconds): 0
Subject,EPA (secs), BEPA (secs), SBEPA
DefectiveComponentCounter,0.583,0.358,0.071
Total time all subjects (seconds): 1
End.
#
```

Para comprobar la ejecución correcta del caso en la imagen, se accedió al directorio donde se guarda el resultado, pero dentro de la imagen (que ya va a tener AssetTransfer y BasicProvenance porque los habíamos ejecutado previamente antes de crear la imagen), para ver si tenía generado los resultados para DefectiveComponentCounter:

```bash
# ls
Alloy4PA README.md config.properties examples results table_1.py
# cd examples
# dir
Benchmarks motivation rq2
# cd Benchmarks
# ls
B1  B2
# cd B1
# ls
AssetTransfer BasicProvenance DefectiveComponentCounter alloy_models
#
```

Y se validó que fuera de la imagen los resultados para DefectiveComponentCounter, efectivamente, no estaban generados.

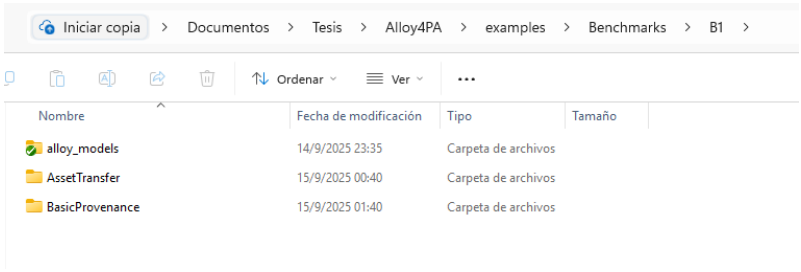


Fig. 4.14: Confirmación de resultados en Docker

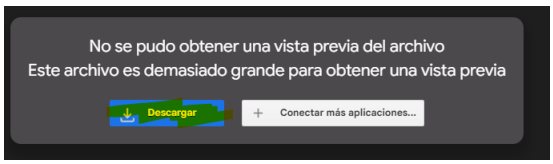
4.2.5. Modalidad con imagen de Docker prearmada

4.2.5.1. Instalación de prerrequisitos

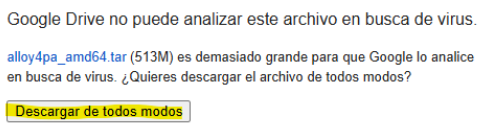
Para este paso no fue necesario instalar nada nuevo, ni comprobar versiones. Tan solo abrir Docker Desktop y mantenerlo ejecutando en segundo plano.

4.2.5.2. Descarga y ejecución de la imagen

Se descargó la imagen provista desde el link [14] informado en el repositorio.



(a) Descarga de la imagen



(b) Confirmación de descarga

Fig. 4.15: Descarga de la imagen Docker prearmada

Luego de descargarla se movió al directorio deseado y se accedió a la terminal para cargar la imagen en Docker y ejecutarla.

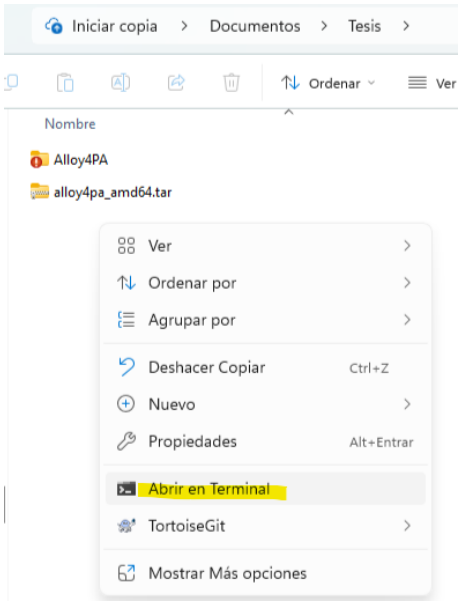


Fig. 4.16: Apertura de terminal en el directorio de la imagen

PowerShell

```
PS C:\Users\USUARIO\Documents\Tesis> docker load -i alloy4pa_amd64.tar
Loaded image: alloy4pa-amd64:latest
PS C:\Users\USUARIO\Documents\Tesis> docker run -it alloy4pa-amd64
#
```

4.2.5.3. Prueba de la imagen Docker provista con un caso de B1

Se llamó a Alloy4PA usando otra vez el caso que mencionan en el repositorio [3] como ejemplo rápido:

bash

```
# java -jar Alloy4PA/target/Alloy4PA-1.0.0-fat-jar-with-dependencies.jar examples/
    ↪Benchmarks/B1/alloy_models/BasicProvenance.als
========================================== Subjects to generate abstraction:
    ↪==========================================
/replication-pkg/examples/Benchmarks/B1/alloy_models/BasicProvenance.als
========================================== STARTING WITH FILE: /replication-pkg/
    ↪examples/Benchmarks/B1/alloy_models/BasicProvenance.als
============================================================
Configuration loaded successfully from config.properties
Max Thread Number: 4
Query Timeout Limit (secs): 600
Verbose: true
Avoid Must Constructor: false
Avoid Must Tau: false
Avoid Must All: false
Allow Address 0x0: true
Output Statistics Path: results/
Overwrite Subjects: false
```

```
16 ================================================================
17 Loading config /replication-pkg/examples/Benchmarks/B1/alloy_models/
       ↪BasicProvenanceConfig.ini
18 Total time generating alloy file with partitions and preds: 0.00 min
19 ========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/
       ↪BasicProvenance/BasicProvenance_part2.als
20 ==============
21 ========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/
       ↪BasicProvenance/BasicProvenance_part4.als
22 ==============
23 ========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/
       ↪BasicProvenance/BasicProvenance_part1.als
24 ==============
25 ========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/
       ↪BasicProvenance/BasicProvenance_part3.als
26 ==============
27 running... transition_S00_to_S01_by_constructor
28 running... transition_S01_to_S01_by_transferResponsibility
29 running... transition_S03_to_S01_by_transferResponsibility
30 running... transition_S02_to_S01_by_transferResponsibility
31 SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
32 SLF4J: Defaulting to no-operation (NOP) logger implementation
33 SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
34 transition_S01_to_S01_by_transferResponsibility: 0.212 - sat? false
35 running... transition_S01_to_S02_by_transferResponsibility
36 transition_S02_to_S01_by_transferResponsibility: 0.211 - sat? false
37 transition_S03_to_S01_by_transferResponsibility: 0.212 - sat? false
38 running... transition_S02_to_S02_by_transferResponsibility
39 running... transition_S03_to_S02_by_transferResponsibility
40 transition_S00_to_S01_by_constructor: 0.231 - sat? true
41 running... transition_S00_to_S02_by_constructor
42 transition_S03_to_S02_by_transferResponsibility: 0.046 - sat? false
43 running... transition_S03_to_S03_by_transferResponsibility
44 transition_S00_to_S02_by_constructor: 0.036 - sat? false
45 running... transition_S00_to_S03_by_constructor
46 transition_S02_to_S02_by_transferResponsibility: 0.054 - sat? true
47 running... transition_S02_to_S03_by_transferResponsibility
48 transition_S01_to_S02_by_transferResponsibility: 0.056 - sat? true
49 running... transition_S01_to_S03_by_transferResponsibility
50 transition_S00_to_S03_by_constructor: 0.013 - sat? false
51 running... transition_S03_to_S03_by_transferResponsibility: 0.017 - sat? false
52 running... transition_S03_to_S01_by_complete
53 transition_S01_to_S03_by_transferResponsibility: 0.014 - sat? false
54 transition_S02_to_S03_by_transferResponsibility: 0.015 - sat? false
55 running... transition_S01_to_S01_by_complete
56 running... transition_S02_to_S01_by_complete
57 transition_S03_to_S01_by_complete: 0.014 - sat? false
58 running... transition_S03_to_S02_by_complete
59 transition_S01_to_S01_by_complete: 0.015 - sat? false
60 running... transition_S01_to_S02_by_complete
61 transition_S02_to_S01_by_complete: 0.015 - sat? false
62 running... transition_S02_to_S02_by_complete
63 transition_S03_to_S02_by_complete: 0.012 - sat? false
64 running... transition_S03_to_S03_by_complete
65 transition_S02_to_S02_by_complete: 0.017 - sat? false
66 running... transition_S02_to_S03_by_complete
67 transition_S01_to_S02_by_complete: 0.017 - sat? false
```

```
68 running... transition_S01_to_S03_by_complete
69 transition_S03_to_S03_by_complete: 0.025 - sat? false
70 transition_S01_to_S03_by_complete: 0.021 - sat? false
71 transition_S02_to_S03_by_complete: 0.024 - sat? true
72 EPA cost(seconds): 0
73 ========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/
       ↪BasicProvenance/BasicProvenance.als
74 ==============
75 ========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/
       ↪BasicProvenance/BasicProvenance.als
76 ==============
77 running... must_transition_S01_to_S02_by_transferResponsibility
78 must_transition_S01_to_S02_by_transferResponsibility: 0.127 - sat? false
79 running... must_transition_S00_to_S01_by_constructor
80 must_transition_S00_to_S01_by_constructor: 0.376 - sat? false
81 running... must_transition_S02_to_S02_by_transferResponsibility
82 must_transition_S02_to_S02_by_transferResponsibility: 0.077 - sat? false
83 running... must_transition_S02_to_S03_by_complete
84 must_transition_S02_to_S03_by_complete: 0.02 - sat? false
85 MUST cost(seconds): 0
86 ========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/
       ↪BasicProvenance/BasicProvenance.als
87 ==============
88 ========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/
       ↪BasicProvenance/BasicProvenance.als
89 ==============
90 Saving file /replication-pkg/examples/Benchmarks/B1/BasicProvenance/
       ↪BasicProvenanceforked_arrows.dot...
91 Hyper-MUST cost(seconds): 0
92 Subject,EPA (secs), BEPA (secs), SBEPA
93 BasicProvenance,0.541,0.729,0.043
94 Total time all subjects (seconds): 1
95 End.
96 # cd examples/Benchmarks/B1
97 # dir
98 BasicProvenance alloy_models
99 #
```

4.3. Ejecución básica en macOS

En esta sección se repitió la ejecución básica de la herramienta Alloy4PA en sus distintas modalidades, en un entorno diferente.

Esta reproducción se realizó en una MacBook Pro con chip Apple M4, usando sistema operativo macOS 15.6 Sequoia.

4.3.1. Inicialización del entorno

Se accedió al repositorio de GitHub que almacena el código de la herramienta [3], y se obtuvo la URL para clonar el repositorio localmente.

Luego, se usó Git para descargar el código a la máquina. El primer paso fue comprobar si Git estaba instalado en la computadora. Abriendo una terminal, se ejecutó:

```bash
$ git --version
git version 2.39.2
```

Confirmando que Git estaba instalado.

En la terminal, desde el directorio home, se creó un directorio para proyectos y se clonó el proyecto adentro. Por último, se ingresó al directorio.

```bash
$ mkdir projects
$ cd projects
$ git clone https://github.com/j-godoy/Alloy4PA.git
$ cd Alloy4PA
```

4.3.2. Modalidad JAR precompilado

Se procedió a probar la primera opción mencionada en el `README` del proyecto: el `JAR` precompilado.

4.3.2.1. Instalación de Java

El primer paso, como describe el `README`, fue comprobar si Java estaba instalado. Se ejecutó:

```bash
$ java --version
OpenJDK Runtime Environment AdoptOpenJDK (build 14.0.2+12)
```

Java 14 es una versión más vieja que la recomendada por la herramienta, que requiere Java 18 en adelante. Se solucionó este problema usando Homebrew [15], el manejador de paquetes para Mac. Siguiendo las recomendaciones actualizadas [16], se desinstaló el Java de AdoptOpenJDK, que está deprecado en Homebrew. Se reinstaló usando `temurin`, que es el paquete actualmente recomendado para instalar Java.

Dado que el usuario principal de la computadora tenía el control de Homebrew, y esta instalación se hizo desde un usuario secundario, fue necesario pasar al usuario principal para ejecutar lo siguiente:

```bash
$ brew uninstall adoptopenjdk14
$ brew install --cask temurin
```

Al volver al usuario secundario, se comprobó nuevamente la versión de Java, y se obtuvo:

```bash
OpenJDK Runtime Environment Temurin-24.0.2+12 (build 24.0.2+12)
```

Al tener Java 24, la versión más reciente, se pudo avanzar con el proceso.

4.3.2.2. Configuración de la variable JAVA_HOME

El siguiente paso fue comprobar la variable JAVA_HOME. Al ejecutar:

```bash
$ echo $JAVA_HOME
```

La respuesta de la consola fue vacía. Esto indicaba que la variable no estaba correctamente configurada.

Se intentó establecer el JAVA_HOME de la forma estándar:

```bash
$ export JAVA_HOME=$(readlink -f $(which java) | sed 's:/bin/java::')
```

Al comprobar nuevamente el valor de la variable, la consola imprimió:

```bash
/usr
```

Teniendo un valor definido para la variable, se intentó correr el programa con la configuración de ejemplo propuesta en el README:

```bash
$ java -jar Alloy4PA/target/Alloy4PA-1.0.0-fat-jar-with-dependencies.jar examples/
    ↪Benchmarks/B1/alloy_models/BasicProvenance.als
```

El programa comenzó a ejecutar, pero no imprimió nada. Después de un tiempo, se interrumpió su ejecución usando Ctrl+C. Para entender mejor el problema, se ejecutó:

```bash
> java --version
```

Y este proceso también colgó de la misma forma. Al correr esto último con sudo, esta vez imprimió el número de versión anterior. Esto indicó que había existido un problema al colocar el JAVA_HOME, y que el usuario actual no tenía permisos suficientes.

Investigando, se descubrió que la forma estándar de manejar esta variable es desde el comienzo de la sesión de terminal [17]. Se agregó al archivo ~/.zshrc lo siguiente:

bash

```
export JAVA_HOME=$(/usr/libexec/java_home)
export PATH=$JAVA_HOME/bin:$PATH
```

Y luego se abrió una nueva terminal para que se ejecutaran esas líneas. Luego se comprobó que el comando para obtener la versión de Java estaba funcionando correctamente de nuevo, sin ser superusuario.

Se corrió nuevamente el programa con la configuración de ejemplo. El programa ejecutó exitosamente en poco tiempo, y creó la carpeta con los archivos de resultado.

bash

```
> java -jar Alloy4PA/target/Alloy4PA-1.0.0-fat-jar-with-dependencies.jar examples/
    ↪Benchmarks/B1/alloy_models/BasicProvenance.als
========================================
Subjects to generate abstraction:
========================================
/Users/main/projects/Alloy4PA/examples/Benchmarks/B1/alloy_models/BasicProvenance.als
========================================

STARTING WITH FILE: /Users/main/projects/Alloy4PA/examples/Benchmarks/B1/alloy_models/
    ↪BasicProvenance.als
============================================================
Configuration loaded successfully from config.properties
Max Thread Number: 4
Query Timeout Limit (secs): 600
Verbose: true
Avoid Must Constructor: false
Avoid Must Tau: false
Avoid Must All: false
Allow Address 0x0: true
Output Statistics Path: results/
Overwrite Subjects: false
============================================================
Loading config /Users/main/projects/Alloy4PA/examples/Benchmarks/B1/alloy_models/
    ↪BasicProvenanceConfig.ini
Total time generating alloy file with partitions and preds: 0,00 min
========== Parsing+Typechecking /Users/main/projects/Alloy4PA/examples/Benchmarks/B1/
    ↪BasicProvenance/BasicProvenance_part1.als =============
========== Parsing+Typechecking /Users/main/projects/Alloy4PA/examples/Benchmarks/B1/
    ↪BasicProvenance/BasicProvenance_part2.als =============
========== Parsing+Typechecking /Users/main/projects/Alloy4PA/examples/Benchmarks/B1/
    ↪BasicProvenance/BasicProvenance_part4.als =============
========== Parsing+Typechecking /Users/main/projects/Alloy4PA/examples/Benchmarks/B1/
    ↪BasicProvenance/BasicProvenance_part3.als =============
running... transition_S01_to_S01_by_transferResponsibility
running... transition_S02_to_S01_by_transferResponsibility
running... transition_S00_to_S01_by_constructor
running... transition_S03_to_S01_by_transferResponsibility
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
transition_S01_to_S01_by_transferResponsibility: 1.117 - sat? false
transition_S02_to_S01_by_transferResponsibility: 1.116 - sat? false
transition_S03_to_S01_by_transferResponsibility: 1.117 - sat? false
```

```
running... transition_S01_to_S02_by_transferResponsibility
running... transition_S03_to_S02_by_transferResponsibility
running... transition_S02_to_S02_by_transferResponsibility
transition_S00_to_S01_by_constructor: 1.231 - sat? true
running... transition_S00_to_S02_by_constructor
transition_S00_to_S02_by_constructor: 0.246 - sat? false
running... transition_S00_to_S03_by_constructor
transition_S03_to_S02_by_transferResponsibility: 0.313 - sat? false
running... transition_S03_to_S03_by_transferResponsibility
transition_S02_to_S02_by_transferResponsibility: 0.346 - sat? true
running... transition_S02_to_S03_by_transferResponsibility
transition_S01_to_S02_by_transferResponsibility: 0.352 - sat? true
running... transition_S01_to_S03_by_transferResponsibility
transition_S01_to_S03_by_transferResponsibility: 0.063 - sat? false
running... transition_S01_to_S01_by_complete
transition_S00_to_S03_by_constructor: 0.105 - sat? false
transition_S03_to_S03_by_transferResponsibility: 0.105 - sat? false
transition_S02_to_S03_by_transferResponsibility: 0.07 - sat? false
running... transition_S03_to_S01_by_complete
running... transition_S02_to_S01_by_complete
transition_S02_to_S01_by_complete: 0.046 - sat? false
running... transition_S02_to_S02_by_complete
transition_S03_to_S01_by_complete: 0.051 - sat? false
transition_S01_to_S01_by_complete: 0.053 - sat? false
running... transition_S03_to_S02_by_complete
running... transition_S01_to_S02_by_complete
transition_S01_to_S02_by_complete: 0.045 - sat? false
running... transition_S01_to_S03_by_complete
transition_S02_to_S02_by_complete: 0.05 - sat? false
transition_S03_to_S02_by_complete: 0.046 - sat? false
running... transition_S03_to_S03_by_complete
running... transition_S02_to_S03_by_complete
transition_S03_to_S03_by_complete: 0.035 - sat? false
transition_S01_to_S03_by_complete: 0.037 - sat? false
transition_S02_to_S03_by_complete: 0.043 - sat? true
EPA cost(seconds): 2

========== Parsing+Typechecking /Users/main/projects/Alloy4PA/examples/Benchmarks/B1/
    ↪BasicProvenance/BasicProvenance.als =============
========== Parsing+Typechecking /Users/main/projects/Alloy4PA/examples/Benchmarks/B1/
    ↪BasicProvenance/BasicProvenance.als =============
running... must_transition_S02_to_S03_by_complete
must_transition_S02_to_S03_by_complete: 0.14 - sat? false
running... must_transition_S02_to_S02_by_transferResponsibility
must_transition_S02_to_S02_by_transferResponsibility: 0.176 - sat? false
running... must_transition_S01_to_S02_by_transferResponsibility
must_transition_S01_to_S02_by_transferResponsibility: 0.163 - sat? false
running... must_transition_S00_to_S01_by_constructor
must_transition_S00_to_S01_by_constructor: 0.534 - sat? false
MUST cost(seconds): 1

========== Parsing+Typechecking /Users/main/projects/Alloy4PA/examples/Benchmarks/B1/
    ↪BasicProvenance/BasicProvenance.als =============
========== Parsing+Typechecking /Users/main/projects/Alloy4PA/examples/Benchmarks/B1/
    ↪BasicProvenance/BasicProvenance.als =============
Saving file /Users/main/projects/Alloy4PA/examples/Benchmarks/B1/BasicProvenance/
    ↪BasicProvenanceforked_arrows.dot...
```

```
89  Hyper-MUST cost(seconds): 0
90
91  Subject,EPA (secs), BEPA (secs), SBEPA
92  BasicProvenance,2.584,1.333,0.132
93  Total time all subjects (seconds): 4
94  End.
```

4.3.3. Compilar archivo JAR desde el código fuente

La compilación del programa desde el código fuente requiere Maven, por lo cual el primer paso fue verificar si estaba instalado. Se ejecutó:

bash

```
1  $ mvn --version
2  Apache Maven 3.8.4 (9b656c72d54e5bacbed989b64718c159fe39b537)
```

Indicando que Maven estaba instalado.

4.3.3.1. Instalación local de Alloy

Lo siguiente fue instalar localmente el JAR de Alloy, como indica el `README`. Se ejecutó:

bash

```
1  $ mvn install:install-file -Dfile='libs/org.alloytools.alloy6.dist.jar' -DgroupId='
       ↪org.alloytools.alloy6.dist' -DartifactId='alloy' -Dversion='6.0.0' -Dpackaging='
       ↪jar'
```

El resultado fue un error:

bash

```
1  Failed to execute goal org.apache.maven.plugins:maven-install-plugin:2.4:install-file
       ↪(default-cli) on project standalone-pom: The specified file '/Users/main/
       ↪projects/Alloy4PA/libs/org.alloytools.alloy6.dist.jar' not exists
```

Releyendo las instrucciones, el documento indicaba correr este proceso desde el directorio interno Alloy4PA/, no desde el root del proyecto. La confusión se dio porque el root y la carpeta interna habían quedado con el mismo nombre. Se ejecutó

bash

```
1  $ cd Alloy4PA/
```

Y se volvió a correr el proceso. Esta vez terminó con éxito.

4.3.3.2. Compilación del JAR

Luego, se ejecutó el comando de Maven para compilar el JAR:

bash

```
$ mvn clean package
```

Este comando corrió durante 4 minutos, descargó varios archivos de requerimientos de la herramienta, y completó con éxito:

bash

```
[INFO] Building jar: /Users/main/projects/Alloy4PA/Alloy4PA/target/Alloy4PA-1.0.0-fat-
    ↪jar-with-dependencies.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 04:04 min
```

El JAR creado se almacenó en el mismo lugar donde estaba el JAR precompilado original. Por lo tanto, para probarlo, se volvió al directorio raíz, y se ejecutó como en la opción 1:

bash

```
$ cd ..
$ java -jar Alloy4PA/target/Alloy4PA-1.0.0-fat-jar-with-dependencies.jar examples/
    ↪Benchmarks/B1/alloy_models/BasicProvenance.als
```

El proceso indicó que ya había resultados almacenados para ese benchmark, y no lo corrió nuevamente. Al terminar, el programa informó un error al crear el archivo de estadísticas.

bash

```
Error writting Times: results/13_09_2025_23_48_statistics.csv (Permission denied)
java.io.FileNotFoundException: results/13_09_2025_23_48_statistics.csv (Permission
    ↪denied)
    at java.base/java.io.FileOutputStream.open0(Native Method)
    at java.base/java.io.FileOutputStream.open(FileOutputStream.java:255)
    at java.base/java.io.FileOutputStream.<init>(FileOutputStream.java:210)
    at java.base/java.io.FileOutputStream.<init>(FileOutputStream.java:120)
    at java.base/java.io.FileWriter.<init>(FileWriter.java:67)
    at ar.uba.dc.MainMODELS.writeTimes(MainMODELS.java:168)
    at ar.uba.dc.MainMODELS.main(MainMODELS.java:78)
```

Se analizó el dueño de cada directorio en el proyecto:

bash

```
$ ls -l
```

Y fue evidente que el directorio `results/` era el único cuyo dueño era `root`. Esto se debe a que la primera ejecución exitosa se había hecho con `sudo`, y el superusuario `root` había

generado la carpeta de resultados, con permisos para sí mismo. Por lo tanto, el usuario base no tenía acceso. Este problema se solucionó cambiando el usuario de ese directorio:

bash

```
$ sudo chown -R main:staff results/
```

Al correr nuevamente el programa, se confirmó que el mensaje de error ya no aparecía, y el archivo CSV se guardó correctamente.

4.3.4. Modalidad con imagen de Docker prearmada

4.3.4.1. Descarga y ejecución de la imagen

Se comenzó abriendo el programa Orbstack, ya instalado en el equipo, para activar el servicio de Docker.

Para comprobar el estado de Docker, se ejecutó en la terminal:

bash

```
$ docker ps
CONTAINER ID  IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

La respuesta de la consola indicó que Docker estaba corriendo, y no había contenedores activos.

Luego se descargó la imagen para arquitectura `x86` desde el link [14] que aparece en el `README`.

En la consola, se navegó hasta un directorio donde se quería almacenar la imagen:

bash

```
$ cd projects
$ mkdir Alloy4PA-Docker
$ $cd Alloy4PA-Docker
$ open .
```

Desde el Finder (explorador de archivos) se movió el archivo descargado a este directorio. Luego, se lo cargó en Docker como indica el README:

bash

```
$ docker load -i alloy4pa_amd64.tar
4695cdfb426a: Loading layer [==================================================>]
    ↪31.41MB/31.41MB
81d28cca9b37: Loading layer [==================================================>]
    ↪1.582MB/1.582MB
a40ee0f44ba2: Loading layer [==================================================>]
    ↪189.1MB/189.1MB
3a3c37b590cd: Loading layer [==================================================>]
    ↪2.465MB/2.465MB
e9adeca65217: Loading layer [==================================================>]
    ↪8.351MB/8.351MB
```

```
d10a999de97f: Loading layer [==================================================>] 855B
    ↪/855B
4f468d2b5335: Loading layer [==================================================>] 359B
    ↪/359B
2c773f3df16b: Loading layer [==================================================>]
    ↪166.2MB/166.2MB
40e664e6c32a: Loading layer [==================================================>] 105B
    ↪/105B
0a4d6bf4ba96: Loading layer [==================================================>]
    ↪37.67kB/37.67kB
cd2d0c8b478f: Loading layer [==================================================>]
    ↪46.77MB/46.77MB
16b12c576a7a: Loading layer [==================================================>] 691B
    ↪/691B
6f95d397cb03: Loading layer [==================================================>]
    ↪4.199kB/4.199kB
1c2cca1bb423: Loading layer [==================================================>]
    ↪2.264kB/2.264kB
e189e3f0ecff: Loading layer [==================================================>]
    ↪20.81MB/20.81MB
ff3ff60428a2: Loading layer [==================================================>] 71MB
    ↪/71MB
5f70bf18a086: Loading layer [==================================================>] 32B
    ↪/32B
Loaded image: alloy4pa-amd64:latest

WARNING: The requested image's platform (linux/amd64) does not match the detected host
    ↪ platform (linux/arm64/v8) and no specific platform was requested
```

La advertencia indicó que la imagen era para una arquitectura diferente a la del equipo. Investigando, se encontró que el `README` provee una imagen para `amd64`, y otra distinta para `arm64`. Se validó que el equipo con chip Apple M4 tiene arquitectura `arm64` [18], lo que indica que la otra imagen era la correcta [1].

Se descargó la imagen desde el otro link [19], y se repitió el proceso de carga en Docker:

bash

```
$ docker load -i alloy4pa_arm64.tar
(...)
Loaded image: alloy4pa-arm64:latest
```

El resultado fue similar, pero sin la advertencia. Se abrió la consola dentro del contenedor de la imagen:

bash

```
$ docker run -it alloy4pa-arm64
```

4.3.4.2. Prueba de la imagen Docker provista con un caso de B1

Dentro de la consola del contenedor, se llamó a Alloy4PA usando el caso simple:

[1] La imagen para `amd64` también funciona, a través de virtualización, pero su ejecución es más lenta.

bash

```
# java -jar Alloy4PA/target/Alloy4PA-1.0.0-fat-jar-with-dependencies.jar examples/Benchmarks/B1/alloy_models/BasicProvenance.als
========================================
Subjects to generate abstraction:
========================================
/replication-pkg/examples/Benchmarks/B1/alloy_models/BasicProvenance.als
========================================

STARTING WITH FILE: /replication-pkg/examples/Benchmarks/B1/alloy_models/BasicProvenance.als
============================================================
Configuration loaded successfully from config.properties
Max Thread Number: 4
Query Timeout Limit (secs): 600
Verbose: true
Avoid Must Constructor: false
Avoid Must Tau: false
Avoid Must All: false
Allow Address 0x0: true
Output Statistics Path: results/
Overwrite Subjects: false
============================================================
Loading config /replication-pkg/examples/Benchmarks/B1/alloy_models/BasicProvenanceConfig.ini
Total time generating alloy file with partitions and preds: 0.00 min
========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/BasicProvenance/BasicProvenance_part1.als =============
========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/BasicProvenance/BasicProvenance_part3.als =============
========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/BasicProvenance/BasicProvenance_part2.als =============
========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/BasicProvenance/BasicProvenance_part4.als =============
running... transition_S02_to_S01_by_transferResponsibility
running... transition_S00_to_S01_by_constructor
running... transition_S03_to_S01_by_transferResponsibility
running... transition_S01_to_S01_by_transferResponsibility
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
transition_S01_to_S01_by_transferResponsibility: 0.153 - sat? false
running... transition_S01_to_S02_by_transferResponsibility
transition_S03_to_S01_by_transferResponsibility: 0.152 - sat? false
transition_S02_to_S01_by_transferResponsibility: 0.151 - sat? false
running... transition_S02_to_S02_by_transferResponsibility
running... transition_S03_to_S02_by_transferResponsibility
transition_S00_to_S01_by_constructor: 0.164 - sat? true
running... transition_S00_to_S02_by_constructor
transition_S02_to_S02_by_transferResponsibility: 0.026 - sat? true
running... transition_S02_to_S03_by_transferResponsibility
transition_S00_to_S02_by_constructor: 0.019 - sat? false
running... transition_S00_to_S03_by_constructor
transition_S01_to_S02_by_transferResponsibility: 0.03 - sat? true
transition_S03_to_S02_by_transferResponsibility: 0.029 - sat? false
running... transition_S01_to_S03_by_transferResponsibility
```

```
49 running... transition_S03_to_S03_by_transferResponsibility
50 transition_S02_to_S03_by_transferResponsibility: 0.011 - sat? false
51 running... transition_S02_to_S01_by_complete
52 transition_S03_to_S03_by_transferResponsibility: 0.014 - sat? false
53 running... transition_S03_to_S01_by_complete
54 transition_S00_to_S03_by_constructor: 0.019 - sat? false
55 transition_S01_to_S03_by_transferResponsibility: 0.02 - sat? false
56 running... transition_S01_to_S01_by_complete
57 transition_S02_to_S01_by_complete: 0.014 - sat? false
58 running... transition_S02_to_S02_by_complete
59 transition_S03_to_S01_by_complete: 0.011 - sat? false
60 running... transition_S03_to_S02_by_complete
61 transition_S01_to_S01_by_complete: 0.015 - sat? false
62 transition_S02_to_S02_by_complete: 0.012 - sat? false
63 running... transition_S01_to_S02_by_complete
64 running... transition_S02_to_S03_by_complete
65 transition_S03_to_S02_by_complete: 0.011 - sat? false
66 running... transition_S03_to_S03_by_complete
67 transition_S03_to_S03_by_complete: 0.019 - sat? false
68 transition_S01_to_S02_by_complete: 0.022 - sat? false
69 running... transition_S01_to_S03_by_complete
70 transition_S02_to_S03_by_complete: 0.023 - sat? true
71 transition_S01_to_S03_by_complete: 0.006 - sat? false
72 EPA cost(seconds): 0
73
74 ========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/
       ↪BasicProvenance/BasicProvenance.als =============
75 ========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/
       ↪BasicProvenance/BasicProvenance.als =============
76 running... must_transition_S01_to_S02_by_transferResponsibility
77 must_transition_S01_to_S02_by_transferResponsibility: 0.062 - sat? false
78 running... must_transition_S00_to_S01_by_constructor
79 must_transition_S00_to_S01_by_constructor: 0.199 - sat? false
80 running... must_transition_S02_to_S02_by_transferResponsibility
81 must_transition_S02_to_S02_by_transferResponsibility: 0.046 - sat? false
82 running... must_transition_S02_to_S03_by_complete
83 must_transition_S02_to_S03_by_complete: 0.016 - sat? false
84 MUST cost(seconds): 0
85
86 ========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/
       ↪BasicProvenance/BasicProvenance.als =============
87 ========== Parsing+Typechecking /replication-pkg/examples/Benchmarks/B1/
       ↪BasicProvenance/BasicProvenance.als =============
88 Saving file /replication-pkg/examples/Benchmarks/B1/BasicProvenance/
       ↪BasicProvenanceforked_arrows.dot...
89 Hyper-MUST cost(seconds): 0
90
91 Subject,EPA (secs), BEPA (secs), SBEPA
92 BasicProvenance,0.38,0.383,0.028
93 Total time all subjects (seconds): 0
94 End.
```

La ejecución fue exitosa. Para terminar de validar, se usó Orbstack para comprobar que se habían generado los resultados dentro del contenedor:

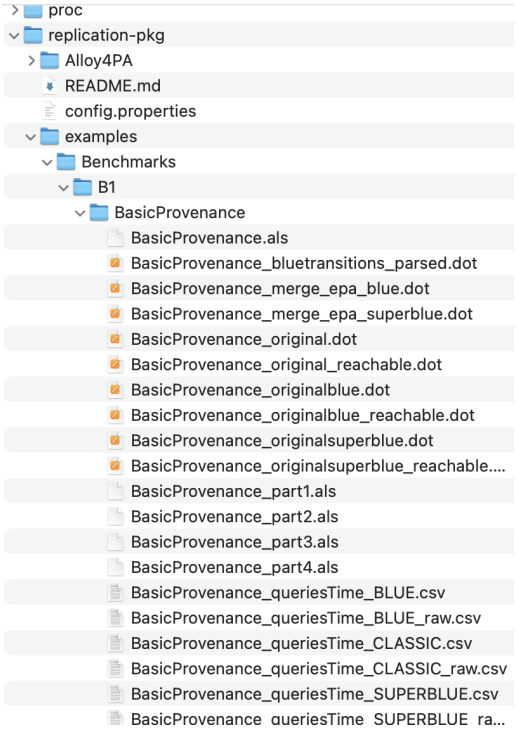


Fig. 4.17: Confirmación de resultados en Docker

4.3.5. Modalidad generando una imagen de Docker

4.3.5.1. Creación de la imagen de Docker

Se siguieron los pasos indicados en el archivo `DOCKER_INSTRUCTIONS.md` del repositorio. La primera instrucción dice que, para compilar en una arquitectura `arm64`, se debe descomentar la línea 3 del archivo `Dockerfile`, y comentar la línea 6.

Se procedió a editar el archivo `Dockerfile` según la instrucción:

bash

```
1  $ nano Dockerfile
```

Luego, se corrió el comando para crear la imagen:

bash

```
1  $ docker build -t alloy4pa-replication .
2  [+] Building 212.7s (16/16) FINISHED
       ↪
       ↪docker:orbstack
3   => [internal] load build definition from Dockerfile
        ↪                                                                    0.0s
4   => => transferring dockerfile: 967B
        ↪
        ↪0.0s
5   => [internal] load metadata for docker.io/library/maven:3.8.7-openjdk-18-slim@sha256
        ↪:4757c1ff47a6139f6a7e6a3ef4c2bef630422b74a636d46f56ea5f 3.6s
```

```
6  => [internal] load .dockerignore
       ↪
       ↪0.0s
7  => => transferring context: 2B
       ↪
       ↪0.0s
8  => [ 1/11] FROM docker.io/library/maven:3.8.7-openjdk-18-slim@sha256:4757
       ↪c1ff47a6139f6a7e6a3ef4c2bef630422b74a636d46f56ea5f16b966076a 0.2s
9  => => resolve docker.io/library/maven:3.8.7-openjdk-18-slim@sha256:4757
       ↪c1ff47a6139f6a7e6a3ef4c2bef630422b74a636d46f56ea5f16b966076a 0.0s
10 => => sha256:4757c1ff47a6139f6a7e6a3ef4c2bef630422b74a636d46f56ea5f16b966076a 1.79kB
       ↪/ 1.79kB                                          0.0s
11 => => sha256:38809959e71b90569b4bba47096881e21f1824787abd860965fdbfc329d0d403 7.73kB
       ↪/ 7.73kB                                          0.0s
12 => [internal] load build context
       ↪
       ↪0.2s
13 => => transferring context: 48.55MB
       ↪
       ↪0.2s
14 => [ 2/11] RUN apt-get update && apt-get install -y   graphviz    curl    git
       ↪virtualenv    python3    python3-pip    nano   44.9s
15 => [ 3/11] WORKDIR /replication-pkg
       ↪
       ↪0.0s
16 => [ 4/11] COPY examples/ examples
       ↪
       ↪0.0s
17 => [ 5/11] COPY Alloy4PA/ Alloy4PA
       ↪
       ↪0.1s
18 => [ 6/11] COPY config.properties .
       ↪
       ↪0.0s
19 => [ 7/11] COPY README.md .
       ↪
       ↪0.0s
20 => [ 8/11] COPY table_1.py .
       ↪
       ↪0.1s
21 => [ 9/11] RUN mvn install:install-file -Dfile="Alloy4PA/libs/
       ↪org.alloytools.alloy6.dist.jar" -DgroupId="org.alloytools.alloy6.dist" -Da 4.5s
22 => [10/11] RUN cd Alloy4PA && mvn clean package
       ↪
       ↪158.0s
23 => [11/11] RUN cd ..
       ↪
       ↪0.1s
24 => exporting to image
       ↪
       ↪1.0s
25 => => exporting layers
       ↪
       ↪1.0s
26 => => writing image sha256:18
       ↪b64735daec6e62d9dd9403ba706399aa6f46fcc2624d8ede88ccd7dd5953a0
       ↪                                        0.0s
```

```
=> => naming to docker.io/library/alloy4pa-replication
                                                                    0.0s
```

El proceso completó exitosamente, y creó la imagen `alloy4pa-replication`. Se realizó una corrida de la imagen:

bash

```bash
$ docker run -it alloy4pa-replication
```

4.3.5.2. Prueba de la imagen Docker creada con un caso de B1

Dentro de la consola del contenedor, se llamó a Alloy4PA usando el caso simple:

bash

```bash
# java -jar Alloy4PA/target/Alloy4PA-1.0.0-fat-jar-with-dependencies.jar examples/Benchmarks/B1/alloy_models/BasicProvenance.als
========================================
Subjects to generate abstraction:
========================================
/replication-pkg/examples/Benchmarks/B1/alloy_models/BasicProvenance.als
========================================

STARTING WITH FILE: /replication-pkg/examples/Benchmarks/B1/alloy_models/BasicProvenance.als
==============================================================
Configuration loaded successfully from config.properties
Max Thread Number: 4
Query Timeout Limit (secs): 600
Verbose: true
Avoid Must Constructor: false
Avoid Must Tau: false
Avoid Must All: false
Allow Address 0x0: true
Output Statistics Path: results/
Overwrite Subjects: false
==============================================================
Subject /replication-pkg/examples/Benchmarks/B1/alloy_models/BasicProvenance.als already processed. SKIPPED
Subject,EPA (secs), BEPA (secs), SBEPA
Total time all subjects (seconds): 0
End.
```

El resultado indica que el caso ya había sido procesado, dado que la imagen fue creada desde el directorio donde ya se había ejecutado esta prueba a partir del JAR fuera de Docker. La herramienta identificó correctamente la presencia de los resultados, y no lo volvió a ejecutar.

4.4. Reproducción completa de los experimentos

Habiendo verificado la correcta ejecución inicial de la herramienta en todas sus modalidades, se procedió a realizar una ejecución completa de los experimentos presentados en

el paper.

Esta reproducción se realizó en el entorno Windows detallado anteriormente, usando la modalidad de archivo JAR precompilado.

4.4.1. Ejecución de ambos Benchmarks completos

4.4.1.1. Ejecución del benchmark B1 usando el JAR provisto fuera de Docker

Se llamó para correr juntos los casos de #B1 (notar que los ya generados los detecta, al ver creada la carpeta destino, y no los vuelve a calcular, en este caso tenía ya previamente generados **AssetTransfer** y **BasicProvenance**):

bash

```
> java -jar Alloy4PA/target/Alloy4PA-1.0.0-fat-jar-with-dependencies.jar examples/Benchmarks/B1/alloy_models/
========================================= Subjects to generate abstraction: =========================================
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\AssetTransfer.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\BasicProvenance.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\DefectiveComponentCounter.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\DigitalLocker.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\FrequentFlyerRC.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\HelloBlockchain.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\RefrigeratedTransportation.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\RoomThermostat.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\SimpleMarketplace.als
========================================= STARTING WITH FILE: C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\AssetTransfer.als
============================================================
Configuration loaded successfully from config.properties
Max Thread Number: 4
Query Timeout Limit (secs): 600
Verbose: true
Avoid Must Constructor: false
Avoid Must Tau: false
Avoid Must All: false
Allow Address 0x0: true
Output Statistics Path: results/
Overwrite Subjects: false
============================================================
Subject C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\AssetTransfer.als already processed. SKIPPED
STARTING WITH FILE: C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\DefectiveComponentCounter.als
Loading config C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\DefectiveComponentCounterConfig.ini
```

```
28 Total time generating alloy file with partitions and preds: 0,00 min
29 ========== Parsing+Typechecking C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\
       ↪Benchmarks\B1\DefectiveComponentCounter\DefectiveComponentCounter_part1.als
30 ==============
31 ========== Parsing+Typechecking C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\
       ↪Benchmarks\B1\DefectiveComponentCounter\DefectiveComponentCounter_part2.als
32 ==============
33 ========== Parsing+Typechecking C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\
       ↪Benchmarks\B1\DefectiveComponentCounter\DefectiveComponentCounter_part3.als
34 ==============
35 running... transition_S02_to_S01_by_computeTotal
36 ...
37 Subject,EPA (secs), BEPA (secs), SBEPA
38 DefectiveComponentCounter,0.929,0.538,0.086
39 FrequentFlyerRC,0.177,1.841,0.066
40 HelloBlockchain,0.084,0.119,0.039
41 RefrigeratedTransportation,0.819,795.262,129.259
42 RoomThermostat,0.08,0.342,0.03
43 SimpleMarketplace,0.134,7.722,0.037
44 Total time all subjects (seconds): 937
45 >
```

Directorio con los resultados generados:

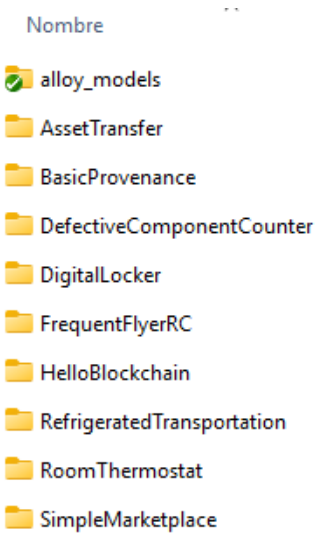


Fig. 4.18: Directorio con los resultados generados para B1

Y un ejemplo de los archivos generados para uno de los casos de B1:

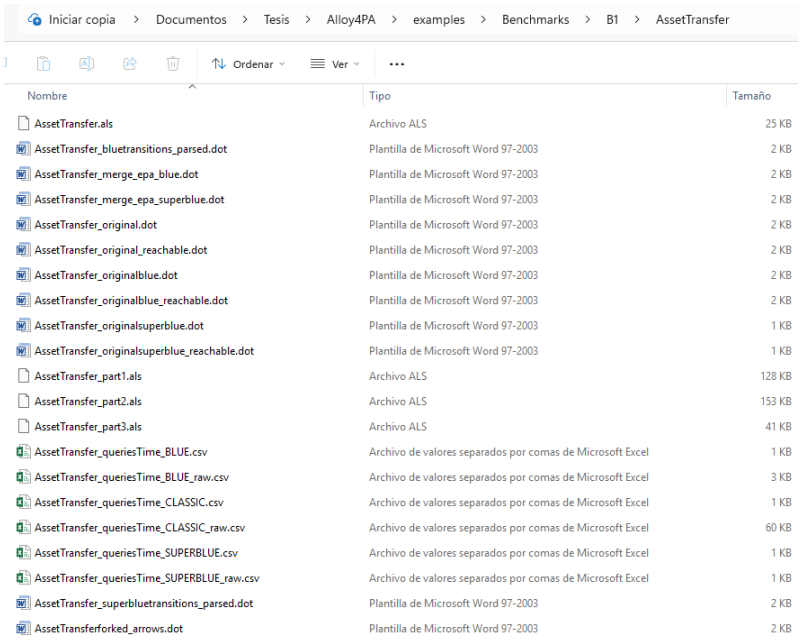


Fig. 4.19: Archivos generados para un caso de B1

4.4.1.2. Ejecución del benchmark B2 usando el JAR provisto fuera de Docker

Dado que en el paper se menciona que este Benchmark llevaría entre 10 a 15 horas, se mandó a ejecutar todos los casos juntos de B2, en horario nocturno, así al día siguiente ver los resultados:

PowerShell

```
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA> java -jar Alloy4PA/target/Alloy4PA
    ↪-1.0.0-fat-jar-with-dependencies.jar examples/Benchmarks/B2/alloy_models/
========================================= Subjects to generate abstraction:
    ↪=========================================
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\Auction+
    ↪EPA.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪AuctionFix+EPA+Ended.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪AuctionFix+EPA.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪CrowdfundingFix_EPA+Balance.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪Crowdfunding_EPA+Balance.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪EPXCrowdsale+EPA+isCrowdSaleClosed.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪EPXCrowdsale+EPA.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪EPXCrowdsale.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪EscrowVault+EPA.als
```

```
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪EscrowVault.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪RefundEscrow+EPA.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪RefundEscrow.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪RockPaperScissors+EPA.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪RockPaperScissors+OneWinner.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪SimpleAuction+Ended+HB.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪SimpleAuction+EPA+Ended.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪SimpleAuction+EPA.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪ValidatorAuction+EPA.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\
    ↪ValidatorAuction.als
========================================== STARTING WITH FILE: C:\Users\USUARIO\
    ↪Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\alloy_models\Auction+EPA.als
============================================================
Configuration loaded successfully from config.properties
Max Thread Number: 4
Query Timeout Limit (secs): 600
Verbose: true
Avoid Must Constructor: false
Avoid Must Tau: false
Avoid Must All: false
Allow Address 0x0: true
Output Statistics Path: results/
Overwrite Subjects: false
============================================================
Loading config C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B2\
    ↪alloy_models\Auction+EPAConfig.ini
=========== Parsing+Typechecking C:\Users\USUARIO\AppData\Local\Temp\
    ↪alloy_heredoc17726869834411392959.als
==============
...
Hyper-MUST cost(seconds): 36
Subject,EPA (secs), BEPA (secs), SBEPA
Auction+EPA,4.421,142.045,16.13
AuctionFix+EPA+Ended,2.058,142.702,33.667
AuctionFix+EPA,2.045,139.829,24.709
CrowdfundingFix_EPA+Balance,19.831,2720.963,307.207
Crowdfunding_EPA+Balance,23.698,2840.806,591.771
EPXCrowdsale+EPA+isCrowdSaleClosed,3.819,71.214,28.09
EPXCrowdsale+EPA,2.633,67.89,30.889
EPXCrowdsale,2.813,42.697,122.864
EscrowVault+EPA,49.793,2517.228,0.336
EscrowVault,6.749,2501.229,0.049
RefundEscrow+EPA,13.073,2340.035,1210.752
RefundEscrow,2.787,4254.562,0.05
RockPaperScissors+EPA,12.529,696.208,0.032
RockPaperScissors+OneWinner,28.058,1842.425,0.028
SimpleAuction+Ended+HB,1.307,10.002,4.969
```

```
56  SimpleAuction+EPA+Ended,5.215,99.807,35.752
57  SimpleAuction+EPA,3.464,92.249,34.998
58  ValidatorAuction+EPA,16.974,751.029,278.409
59  ValidatorAuction,5.049,686.241,36.187
60  Total time all subjects (seconds): 24922
61  End.
```

Como se verá, finalmente llevó 24922 segundos ≈ 6 horas, 55 minutos y 22 segundos. Menos de lo esperado según lo mencionado en el repositorio [3].

Se comprobó que se generaron todos los casos de B2:

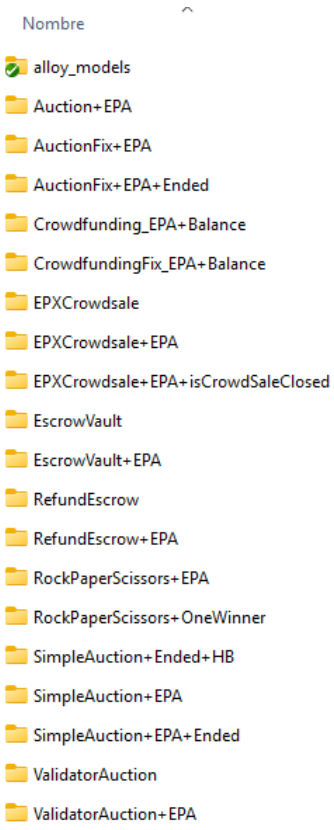
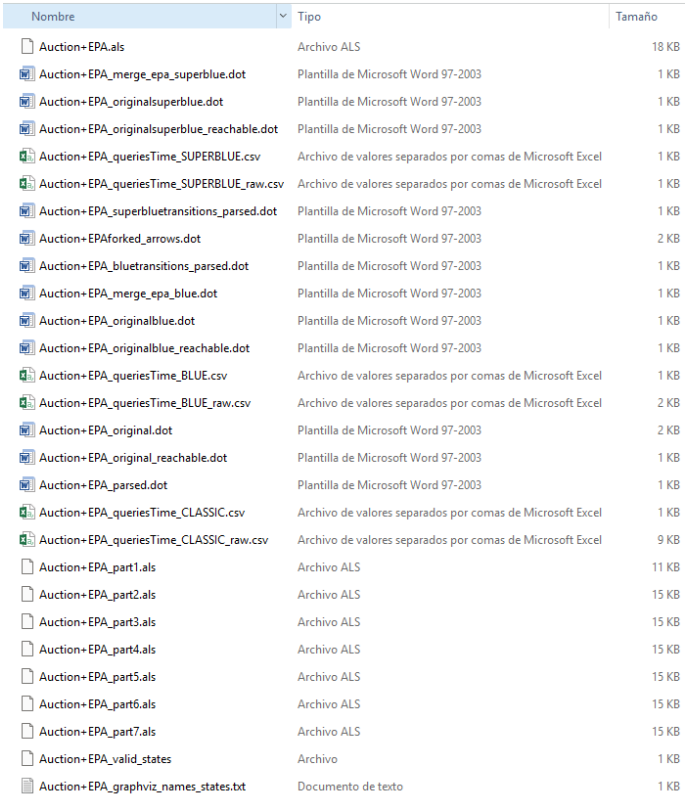


Fig. 4.20: Directorio con los resultados generados para B2

Y se accedió al caso “Auction+EPA” para revisar los archivos generados:



Nombre	Tipo	Tamaño
Auction+EPA.als	Archivo ALS	18 KB
Auction+EPA_merge_epa_superblue.dot	Plantilla de Microsoft Word 97-2003	1 KB
Auction+EPA_originalsuperblue.dot	Plantilla de Microsoft Word 97-2003	1 KB
Auction+EPA_originalsuperblue_reachable.dot	Plantilla de Microsoft Word 97-2003	1 KB
Auction+EPA_queriesTime_SUPERBLUE.csv	Archivo de valores separados por comas de Microsoft Excel	1 KB
Auction+EPA_queriesTime_SUPERBLUE_raw.csv	Archivo de valores separados por comas de Microsoft Excel	1 KB
Auction+EPA_superbluetransitions_parsed.dot	Plantilla de Microsoft Word 97-2003	1 KB
Auction+EPAforked_arrows.dot	Plantilla de Microsoft Word 97-2003	2 KB
Auction+EPA_bluetransitions_parsed.dot	Plantilla de Microsoft Word 97-2003	1 KB
Auction+EPA_merge_epa_blue.dot	Plantilla de Microsoft Word 97-2003	1 KB
Auction+EPA_originalblue.dot	Plantilla de Microsoft Word 97-2003	1 KB
Auction+EPA_originalblue_reachable.dot	Plantilla de Microsoft Word 97-2003	1 KB
Auction+EPA_queriesTime_BLUE.csv	Archivo de valores separados por comas de Microsoft Excel	1 KB
Auction+EPA_queriesTime_BLUE_raw.csv	Archivo de valores separados por comas de Microsoft Excel	2 KB
Auction+EPA_original.dot	Plantilla de Microsoft Word 97-2003	2 KB
Auction+EPA_original_reachable.dot	Plantilla de Microsoft Word 97-2003	1 KB
Auction+EPA_parsed.dot	Plantilla de Microsoft Word 97-2003	1 KB
Auction+EPA_queriesTime_CLASSIC.csv	Archivo de valores separados por comas de Microsoft Excel	1 KB
Auction+EPA_queriesTime_CLASSIC_raw.csv	Archivo de valores separados por comas de Microsoft Excel	9 KB
Auction+EPA_part1.als	Archivo ALS	11 KB
Auction+EPA_part2.als	Archivo ALS	15 KB
Auction+EPA_part3.als	Archivo ALS	15 KB
Auction+EPA_part4.als	Archivo ALS	15 KB
Auction+EPA_part5.als	Archivo ALS	15 KB
Auction+EPA_part6.als	Archivo ALS	15 KB
Auction+EPA_part7.als	Archivo ALS	15 KB
Auction+EPA_valid_states	Archivo	1 KB
Auction+EPA_graphviz_names_states.txt	Documento de texto	1 KB

Fig. 4.21: Archivos generados para el caso Auction+EPA

Se verificó la salida para con el código de `Auction+EPAforked_arrows.dot` copiando de nuevo el código del diagrama en la web [12]:

Graphviz DOT

```
digraph {
  S00 [label="Init"]
  S04 [label="bid & withdraw\n & !auctionEnd & t\n"]
  S07 [label="!bid & withdraw\n & !auctionEnd & t\n"]
  S09 [label="bid & !withdraw\n & !auctionEnd & t\n"]
  S12 [label="!bid & !withdraw\n & !auctionEnd & t\n"]

  S04_withdraw [label="", shape="point", color="blue"]
  S04_τ [label="", shape="point", color="blue"]
  S07_withdraw [label="", shape="point", color="blue"]
  S09_τ [label="", shape="point", color="blue"]

  S00->S12 [label="constructor", style="dotted", color="blue"]
  S00->S09 [label="constructor", style="dotted", color="blue"]
  S12->S12 [label="τ", style="dotted", color="blue"]
  S04->S04_withdraw [label="withdraw", style="dotted", color="blue"]
  S04_withdraw->S04 [label="", style="dotted", color="blue"]
  S04_withdraw->S09 [label="", style="dotted", color="blue"]
  S04->S04_τ [label="τ", style="dotted", color="blue"]
  S04_τ->S04 [label="", style="dotted", color="blue"]
  S04_τ->S07 [label="", style="dotted", color="blue"]
```

```
22   S04->S04 [label="bid", style="", color="black"]
23   S07->S07_withdraw [label="withdraw", style="dotted", color="blue"]
24   S07_withdraw->S07 [label="", style="dotted", color="blue"]
25   S07_withdraw->S12 [label="", style="dotted", color="blue"]
26   S07->S07 [label="τ", style="dotted", color="blue"]
27   S09->S09_τ [label="τ", style="dotted", color="blue"]
28   S09_τ->S09 [label="", style="dotted", color="blue"]
29   S09_τ->S12 [label="", style="dotted", color="blue"]
30   S09->S09 [label="bid", style="", color="black"]
31   S09->S04 [label="bid", style="", color="black"]
32  }
```

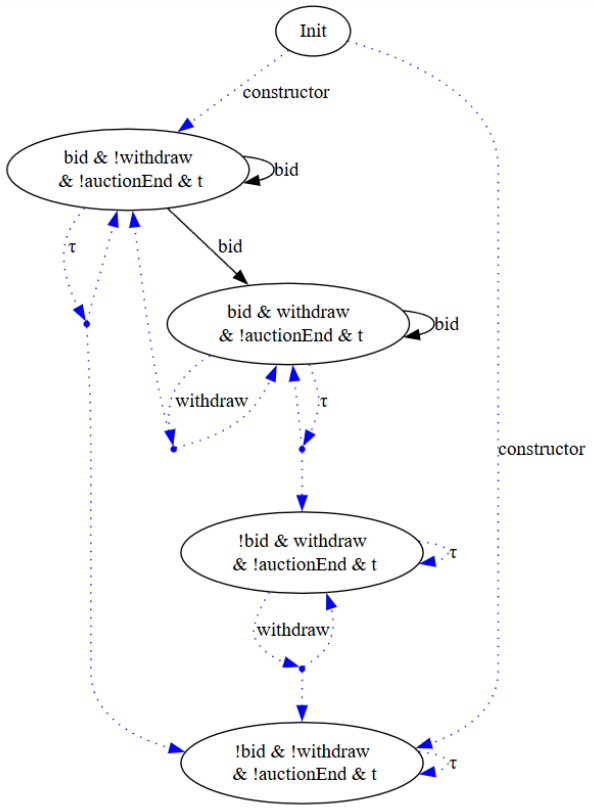


Fig. 4.22: Diagrama de transición de estados para Auction+EPA

Se puede ver que en este caso la mayoría de las transiciones son **must**, pero las transacciones "bid" son solo **may**. (Según se informa en [2] las transacciones may son representadas con líneas enteras y las must con líneas punteadas.)

4.4.2. Reproducción de los datos obtenidos para RQ#1

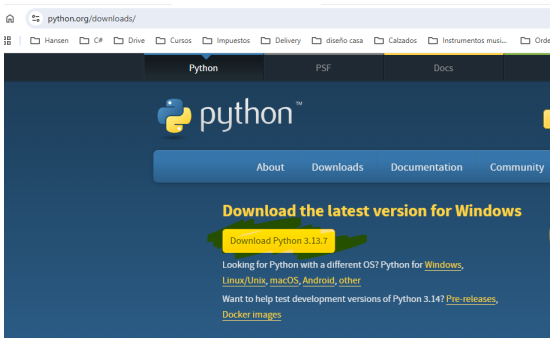
¿Qué tan prevalentes son las transiciones *must* en las abstracciones modales de contratos inteligentes?

4.4.2.1. Instalación de prerrequisitos

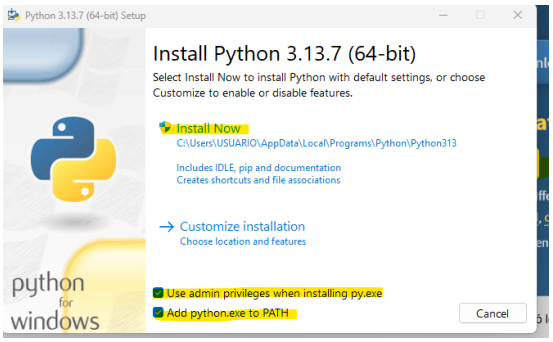
Se verificó que no estaba instalado Python en la máquina:

```bash
> py --version
py : El término 'py' no se reconoce como nombre de un cmdlet, función, archivo de
    ↪script o programa ejecutable. Compruebe si escribió correctamente el nombre o,
    ↪si incluyó una ruta de acceso, compruebe que dicha ruta es correcta e inténtelo
    ↪de nuevo.
En línea: 1 Carácter: 1
+ py --version
+ ~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (python:String) [],
        ↪CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
```

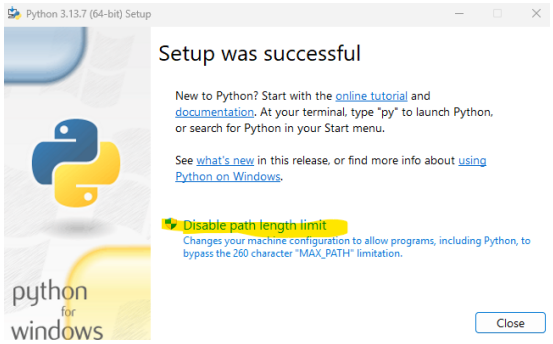
Se procedió, entonces, a descargar Python de la página oficial [20]. A continuación se ejecutó el archivo ejecutable descargado (`python-3.13.7-amd64.exe`) y se siguió los pasos.



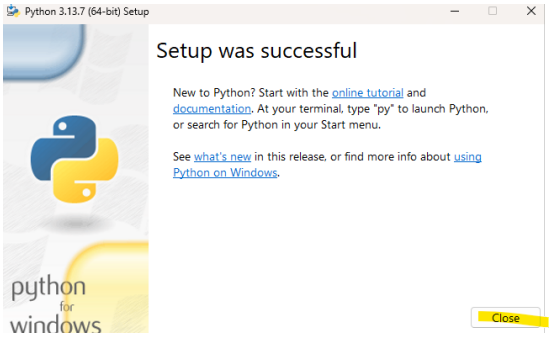
(a) Descarga del instalador



(b) Instalación: paso 1



(c) Instalación: paso 2



(d) Instalación: paso 3

Fig. 4.23: Instalación de Python en Windows

Y se comprobó la correcta instalación:

bash

```
> py --version
Python 3.13.7
```

4.4.2.2. Armado de la tabla 1 del paper a partir de los datos de los Benchmark ejecutados

Se intentó crear la tabla 1 para los ejemplos del Benchmark B1 con el comando proporcionado, pero falló:

PowerShell

```
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA> python3 table_1.py examples/Benchmarks/
    ↪B1
python3 : El término 'python3' no se reconoce como nombre de un cmdlet, función,
    ↪archivo de script o programa ejecutable. Compruebe si escribió correctamente el
    ↪nombre o, si incluyó una ruta de acceso, compruebe que dicha ruta es correcta e
    ↪inténtelo de nuevo.
En línea: 1 Carácter: 1
+ python3 table_1.py examples/Benchmarks/B1
+ ~~~~~~~
    + CategoryInfo          : ObjectNotFound: (python3:String) [],
        ↪CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA>
```

Se corrigió entonces el comando utilizando el comando `py` en vez de `python3`, dado que en Windows el comando a utilizarse es "py" no "python3", aunque si uno quiere indicar que la versión a ejecutar es la más nueva que se tenga de python 3 puede ejecutarse como "py -3" también:

PowerShell

```
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA> py table_1.py examples/Benchmarks/B1
Subject                  States |M|  #Trx  #May  #Must  #Must-t #HM  #HM-t
AssetTransfer            10     11   32    0     32     31      0    0
BasicProvenance          3      3    4     0     4      3       0    0
DefectiveComponentCounter 2     3    4     0     4      3       0    0
DigitalLocker            6      10   12    0     12     11      0    0
FrequentFlyerRC          2      2    3     0     3      2       0    0
HelloBlockchain          2      3    3     0     3      2       0    0
RefrigeratedTransportation 4    4    8     0     4      3       2    2
RoomThermostat           2      4    4     0     4      3       0    0
SimpleMarketplace        3      4    4     0     4      3       0    0
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA> py -3 table_1.py examples/Benchmarks/B1
Subject                  States |M|  #Trx  #May  #Must  #Must-t #HM  #HM-t
AssetTransfer            10     11   32    0     32     31      0    0
BasicProvenance          3      3    4     0     4      3       0    0
DefectiveComponentCounter 2     3    4     0     4      3       0    0
DigitalLocker            6      10   12    0     12     11      0    0
FrequentFlyerRC          2      2    3     0     3      2       0    0
HelloBlockchain          2      3    3     0     3      2       0    0
RefrigeratedTransportation 4    4    8     0     4      3       2    2
RoomThermostat           2      4    4     0     4      3       0    0
```

```
SimpleMarketplace        3      4    4    0    4     3       0    0
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA>
```

Se creó ahora la tabla 1 para los ejemplos del Benchmark B2:

bash

```
Subject                          States |M|  #Trx  #May  #Must  #Must-t #HM  #HM-t
Auction+EPA                      4      4    15    3     4      0       4    2
AuctionFix+EPA                   6      5    21    3     8      2       5    3
AuctionFix+EPA+Ended             6      5    21    3     8      2       5    3
CrowdfundingFix_EPA+Balance      6      5    21    0     11     4       4    2
Crowdfunding_EPA+Balance         7      5    23    0     10     4       6    2
EPXCrowdsale                     7      6    58    14    8      0       6    6
EPXCrowdsale+EPA                 4      6    17    2     7      4       4    2
EPXCrowdsale+EPA+isCrowdSaleClosed 5    6    20    2     9      5       4    2
EscrowVault                      4      10   17    1     16     11      0    0
EscrowVault+EPA                  4      10   17    1     16     11      0    0
RefundEscrow                     3      8    12    3     9      6       0    0
RefundEscrow+EPA                 4      8    17    3     10     6       2    2
RockPaperScissors+EPA            4      4    6     0     6      5       0    0
RockPaperScissors+OneWinner      6      4    12    0     12     11      0    0
SimpleAuction+Ended+HB           6      5    26    19    7      0       0    0
SimpleAuction+EPA                8      5    33    0     15     6       8    6
SimpleAuction+EPA+Ended          10     5    38    0     18     7       10   8
ValidatorAuction                 5      8    15    7     8      2       0    0
ValidatorAuction+EPA             6      8    22    2     7      2       5    3
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA>
```

Las columnas generadas tienen los mismos valores que dichas columnas en el paper [2], la única diferencia visible es que en el paper tenemos las columnas $|LOC|$, Time, Time M y Time HM, que no se generaron al ejecutar el script en Python. Donde recordemos que:

- **Subject**: nombre del contrato y predicados utilizados para cada contrato
- **States**: los estados abstractos
- $|M|$: funciones públicas que no son *view* ni *pure*
- #Trx: número de transiciones
- #May: transiciones *May* que no están subsumidas por transiciones *must*
- #Must: transiciones *must*
- #Must$-\tau$: transiciones *must* excluyendo τ y el *Constructor*
- #HM: transiciones *hyper-must*
- #HM$-\tau$: transiciones *hyper-must* excluyendo τ y el *Constructor*

4.4.3. Reproducción de los datos obtenidos para RQ#2

¿El uso de transiciones con restricciones ayuda a validar contratos inteligentes cuando se dispone del rol de usuario permitido?

4.4.3.1. Instalación de prerrequisitos

No se necesitaron nuevos prerrequisitos.

4.4.3.2. Prueba de ejecución para reproducir la obtención de los casos de prueba

Se ejecutó el comando para generar los mismos, el cual se ejecutó correctamente:

PowerShell

```
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA> java -jar Alloy4PA/target/Alloy4PA
    ↪-1.0.0-fat-jar-with-dependencies.jar examples/rq2/alloy_models/
========================================= Subjects to generate abstraction:
    ↪=========================================
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\rq2\alloy_models\
    ↪AssetTransfer_instanceAppraiser.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\rq2\alloy_models\
    ↪AssetTransfer_instanceInspector.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\rq2\alloy_models\
    ↪DigitalLocker_bankAgent.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\rq2\alloy_models\
    ↪DigitalLocker_owner.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\rq2\alloy_models\
    ↪DigitalLocker_thirdPartyRequestor.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\rq2\alloy_models\
    ↪RefrigeratedTransportation_counterparty.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\rq2\alloy_models\
    ↪RefrigeratedTransportation_initiatingCounterparty.als
========================================= STARTING WITH FILE: C:\Users\USUARIO\
    ↪Documents\Tesis\Alloy4PA\examples\rq2\alloy_models\
    ↪AssetTransfer_instanceAppraiser.als
============================================================
Configuration loaded successfully from config.properties
Max Thread Number: 4
Query Timeout Limit (secs): 600
Verbose: true
Avoid Must Constructor: false
Avoid Must Tau: false
Avoid Must All: false
Allow Address 0x0: true
Output Statistics Path: results/
Overwrite Subjects: false
============================================================
Loading config C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\rq2\alloy_models\
    ↪AssetTransfer_instanceAppraiserConfig.ini
...
Subject,EPA (secs), BEPA (secs), SBEPA
AssetTransfer_instanceAppraiser,5.779,123.086,0.263
AssetTransfer_instanceInspector,3.889,119.524,0.107
DigitalLocker_bankAgent,1.232,0.583,0.036
DigitalLocker_owner,1.197,0.556,0.044
DigitalLocker_thirdPartyRequestor,1.185,0.558,0.03
RefrigeratedTransportation_counterparty,0.406,618.136,11.223
RefrigeratedTransportation_initiatingCounterparty,0.432,623.243,12.184
Total time all subjects (seconds): 1523
End.
```

4.4.4. Reproducción de los datos obtenidos para RQ#3

¿Los auditores aprovechan la distinción entre transiciones *may* y *must* para revelar problemas en propiedades de comportamiento que no pueden ser capturados únicamente por abstracciones *may*?

4.4.4.1. Instalación de prerrequisitos

No se necesitaron nuevos prerrequisitos.

4.4.4.2. Prueba de ejecución para reproducir la obtención de los casos de prueba

Se ejecutó el comando para generar los mismos, el cual se ejecutó correctamente:

PowerShell

```
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA> java -jar Alloy4PA/target/Alloy4PA
    ↪-1.0.0-fat-jar-with-dependencies.jar examples/motivation/alloy_models/
========================================= Subjects to generate abstraction:
    ↪=========================================
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\motivation\alloy_models\
    ↪auction_buggy.als
C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\motivation\alloy_models\
    ↪auction_fixed.als
========================================= STARTING WITH FILE: C:\Users\USUARIO\
    ↪Documents\Tesis\Alloy4PA\examples\motivation\alloy_models\auction_buggy.als
============================================================
Configuration loaded successfully from config.properties
Max Thread Number: 4
Query Timeout Limit (secs): 600
Verbose: true
Avoid Must Constructor: false
Avoid Must Tau: false
Avoid Must All: false
Allow Address 0x0: true
Output Statistics Path: results/
Overwrite Subjects: false
============================================================
Loading config C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\motivation\
    ↪alloy_models\auction_buggyConfig.ini
...
Saving file C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\motivation\
    ↪auction_fixed\auction_fixedforked_arrows.dot...
Hyper-MUST cost(seconds): 212
Subject,EPA (secs), BEPA (secs), SBEPA
dauction_buggy,9.776,502.249,192.129
auction_fixed,10.044,926.48,212.484
Total time all subjects (seconds): 1853
End.
PS C:\Users\USUARIO\Documents\Tesis\Alloy4PA>
```

4.5. Problemas encontrados

A continuación se detallan los problemas encontrados con la herramienta Alloy4PA durante las reproducciones, y las formas de evitarlos o solucionarlos.

4.5.1. Problema con rutas de archivos en Windows

Durante la reproducción en Windows, se encontró que la herramienta Alloy4PA devolvía un error al ejecutar con algunos de los archivos provistos.

PowerShell

```
PS C:\Users\USUARIO\Projects\Alloy4PA> java -jar Alloy4PA/target/Alloy4PA-1.0.0-fat-jar-with-dependencies.jar .\examples\Benchmarks\B1\alloy_models\AssetTransfer.als
========================================
Subjects to generate abstraction:
========================================
C:\Users\USUARIO\Projects\Alloy4PA\.\examples\Benchmarks\B1\alloy_models\AssetTransfer.als
========================================

STARTING WITH FILE: C:\Users\USUARIO\Projects\Alloy4PA\.\examples\Benchmarks\B1\alloy_models\AssetTransfer.als
================================================================
Configuration loaded successfully from config.properties
Max Thread Number: 4
Query Timeout Limit (secs): 600
Verbose: true
Avoid Must Constructor: false
Avoid Must Tau: false
Avoid Must All: false
Allow Address 0x0: true
Output Statistics Path: results/
Overwrite Subjects: false
================================================================
Loading config C:\Users\USUARIO\Projects\Alloy4PA\.\examples\Benchmarks\B1\alloy_models\AssetTransferConfig.ini
Exception: C:\Users\USUARIO\Projects\Alloy4PA\.\examples\Benchmarks\B1\AssetTransfer\AssetTransfer.als -> C:\Users\USUARIO\Projects\Alloy4PA\_part1.\examples\Benchmarks\B1\AssetTransfer\AssetTransfer
java.nio.file.NoSuchFileException: C:\Users\USUARIO\Projects\Alloy4PA\.\examples\Benchmarks\B1\AssetTransfer\AssetTransfer.als -> C:\Users\USUARIO\Projects\Alloy4PA\_part1.\examples\Benchmarks\B1\AssetTransfer\AssetTransfer
    at java.base/sun.nio.fs.WindowsException.translateToIOException(WindowsException.java:85)
    at java.base/sun.nio.fs.WindowsException.rethrowAsIOException(WindowsException.java:103)
    at java.base/sun.nio.fs.WindowsFileCopy.copy(WindowsFileCopy.java:220)
    at java.base/sun.nio.fs.WindowsFileSystemProvider.copy(WindowsFileSystemProvider.java:282)
    at java.base/java.nio.file.Files.copy(Files.java:1305)
    at ar.uba.dc.partitioner.AlloyGenerator.run(AlloyGenerator.java:468)
    at ar.uba.dc.EPA.AEPA.makePreconditions(AEPA.java:160)
    at ar.uba.dc.EPA.AEPA.buildFullEPA(AEPA.java:117)
    at ar.uba.dc.EPA.CEPA.buildJustEPA(CEPA.java:99)
    at ar.uba.dc.EPA.SBEPA.buildFullEPA(SBEPA.java:36)
    at ar.uba.dc.MainMODELS.main(MainMODELS.java:70)
```

Como se puede observar, la herramienta terminó con una excepción, y no generó el resultado esperado. Resultó llamativo que la herramienta había funcionado previamente con `BasicProvenance` (el ejemplo provisto en el `README`), pero dejó de funcionar al elegir

otro archivo. Además, al eliminar los resultados de `BasicProvenance` y volver a correrlo, esta vez devolvió la excepción.

Luego de investigar un poco, se descubrió que el problema estaba en la forma de llamar al comando en la consola. En el primer intento, para ejecutar `BasicProvenance`, se copió y pegó la ruta del archivo directamente desde el `README`. Luego, para ejecutar `AssetTransfer`, se borró el nombre de archivo y se autocompletó el nuevo con la tecla `Tab`.

PowerShell

```
1    java -jar Alloy4PA/target/Alloy4PA-1.0.0-fat-jar-with-dependencies.jar examples/
         ↪Benchmarks/B1/alloy_models/BasicProvenance.als
2
3    java -jar Alloy4PA/target/Alloy4PA-1.0.0-fat-jar-with-dependencies.jar examples/
         ↪Benchmarks/B1/alloy_models/Asse[TAB]
4
5    java -jar Alloy4PA/target/Alloy4PA-1.0.0-fat-jar-with-dependencies.jar .\examples\
         ↪Benchmarks\B1\alloy_models\AssetTransfer.als
```

El primer ejemplo es el que se pegó directamente del `README`, y funcionó correctamente.

Para el segundo, se borró el nombre de archivo y se comenzó a escribir el otro, usando `Tab` para autocompletar el resto. Como se puede observar en la siguiente línea, el autocompletado cambió el formato de la ruta siguiendo el estilo de Windows, con barras invertidas (\).

Normalmente, ambos formatos son equivalentes, pero en el caso de la herramienta Alloy4PA, el uso de barras invertidas generó un error interno.

Para evitar este problema, se dejó de usar el autocompletado en este sistema, y la herramienta comenzó a funcionar correctamente para cualquier archivo.

5. NUEVOS EJEMPLOS

En esta sección se presentarán nuevos ejemplos de contrato inteligente, que no formaban parte de los casos de estudio originales del paper [2]. Estos ejemplos serán principalmente variaciones de los contratos estudiados en el paper, donde se harán ajustes para crear casos de bug o variantes de funcionamiento.

El objetivo es demostrar la aplicabilidad de la herramienta Alloy4PA a un rango más amplio de contratos inteligentes. Para validar su funcionamiento, se ejecutarán los nuevos ejemplos junto a los originales correspondientes, y se analizará el diagrama obtenido.

Todas las variantes con bugs de los modelos Alloy, así como los contratos Solidity correspondientes y los outputs generados para los nuevos casos de estudio desarrollados en esta tesis, se encuentran disponibles en el repositorio público de GitHub[4], de modo de facilitar su análisis, reutilización y verificación independiente.

5.1. Metodología

Para armar cada nueva variante, se partirá de un contrato existente en el repositorio. Tomando como ejemplo el contrato `BasicProvenance`, se debe empezar en el directorio `examples/Benchmarks/B1/alloy_models`.

Se puede observar que dentro del directorio `alloy_models` se encuentran:

- Un archivo `BasicProvenance.als`, que contiene el código Alloy que modela el contrato.
- Un archivo `BasicProvenanceConfig.ini`, que contiene la configuración específica para ejecutarlo con Alloy4PA.

Este mismo patrón, `<Contrato>.als` y `<Contrato>Config.ini`, se repite para todos los contratos del benchmark, y lo mismo ocurre con el benchmark B2.

Si se quiere crear un nuevo ejemplo basado en `BasicProvenance`, se debe copiar ambos archivos y renombrarlos. Por ejemplo, se puede crear una variante llamada `BasicProvenanceBug.als` junto con `BasicProvenanceBugConfig.ini`. Los nuevos archivos pueden dejarse en el mismo directorio, o moverlos a un nuevo directorio, por ejemplo, `examples/Benchmarks/B3/alloy_models`. La herramienta funcionará correctamente en ambos casos, siempre que se indique la ruta correcta. El archivo de configuración debe respetar el patrón al nombrarlo, de lo contrario la herramienta devolverá un error al no encontrarlo.

Una vez creados los nuevos archivos, se los editará según la variante que se quiera implementar, usando cualquier editor de texto o IDE.

Luego de realizar los cambios, se ejecutará Alloy4PA con el nuevo archivo `.als` como input. La herramienta se ejecutará de la misma forma que con los contratos originales, y generará un directorio de resultados con el nombre del contrato, al mismo nivel que el directorio `alloy_models`.

5.2. Simple Marketplace

En primer lugar, se presentará una variante del contrato de SimpleMarketplace del paper. Este es uno de los contratos más simples del conjunto de benchmarks, y es adecuado

para un primer acercamiento al código que modela los contratos.

5.2.1. Contrato original

El contrato muestra una lógica de marketplace donde un item puede ser puesto a la venta por el dueño (`ItemAvailable`). Otro usuario puede hacer una oferta sobre el item (transición `makeOffer`), y el dueño puede aceptar (transición `acceptOffer`) o rechazar la oferta (transición `reject`).

Si el dueño rechaza la oferta, el item vuelve a estar disponible y puede recibir una oferta diferente. Si el dueño acepta la oferta, el contrato pasa a estar en estado `Accepted`, y la venta concluye.

5.2.2. Variante con bug de aceptar sin ofertas

Un cambio trivial que se puede hacer sobre el contrato es eliminar una de las precondiciones de una transición. En este caso, se eliminará la precondición de la transición `acceptOffer` que requiere que haya una oferta activa. De esta forma, el dueño podría marcar el item como vendido aunque no haya ninguna oferta.

En este código se puede ver cómo se hace desaparecer una precondición:

Alloy

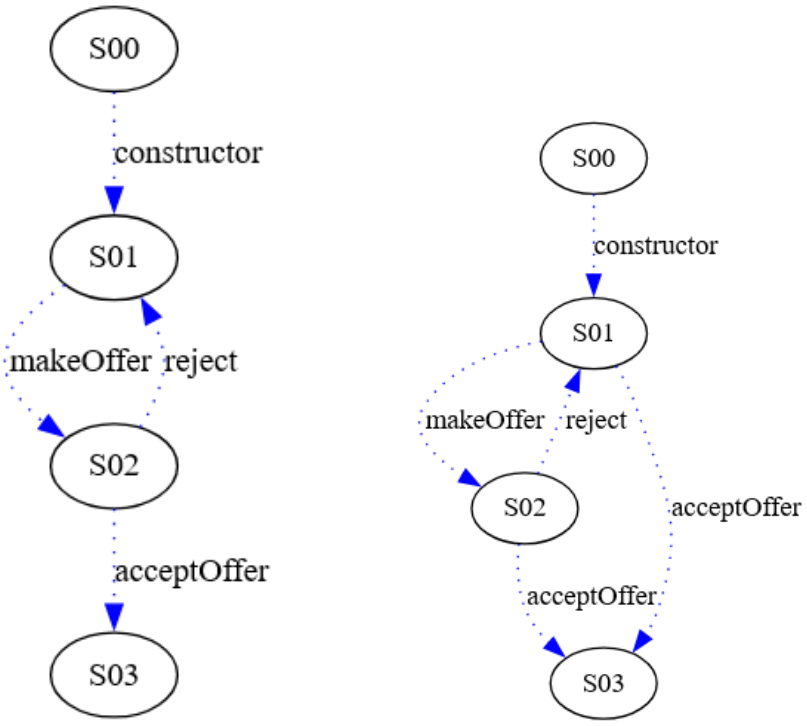
```
1     pred pre_acceptOffer[ein: EstadoConcreto] {
2         some ein._state
3         // ein._state != ItemAvailable
4         ein._state != Accepted
5     }
```

Al hacer esta modificación, se espera que el diagrama del contrato editado incluya una transición extra desde el estado `ItemAvailable` al estado `Accepted`. Esta transición no corresponde a un comportamiento válido de la transacción modelada. Al estar presente, dejaría en evidencia un bug básico que se introdujo.

5.2.3. Comparación de los resultados

Se ejecutaron ambos contratos (el original y la variante con bug) utilizando la herramienta Alloy4PA.

Estos son los diagramas de transición obtenidos:



(a) Original *(b)* Con bug

Fig. 5.1: Comparación de los diagramas de transición del contrato SimpleMarketplace: original y modificada.

Se puede observar que en el diagrama del contrato con bug, es posible hacer una transición `acceptOffer` sin haber pasado por `makeOffer` previamente.

5.3. Rock-Paper-Scissors

En este ejemplo se presentará una variante del contrato de Rock-Paper-Scissors del paper. Para este contrato, se intentará un cambio un poco más elaborado, con el objetivo de introducir un bug que la herramienta manifieste como una transición *may*.

5.3.1. Contrato original

El contrato original puede encontrarse en el archivo:
`examples/Benchmarks/B2/alloy_models/RockPaperScissors+OneWinner.als`.
Este código modela un juego de piedra, papel o tijera entre dos jugadores.

- El juego comienza permitiendo a cada jugador hacer su elección, con los métodos `choicePlayer1` y `choicePlayer2`.
- Luego, se procede a determinar el ganador con el método `determineWinner`.
- Si ambos jugadores eligen la misma opción, el dueño del contrato queda registrado como ganador por defecto.
- El juego contiene un pozo inicial, que se otorga al ganador al finalizar la partida.

5.3.2. Variante con bug de pozo no vacío

El contrato original permite jugar una partida incluso si el pozo está vacío. Además, se observa que no incluye mecanismos para llenar el pozo antes de iniciar una partida. Esto podría significar:

- Que el contrato está incompleto, y no aprovecha el mecanismo del pozo aunque lo tiene modelado.
- Que el contrato actual es una versión simplificada de otro más completo, y se eliminó el uso del pozo para facilitar la abstracción y análisis.

La propuesta de modificación es implementar el uso del pozo de forma incompleta, generando un bug.

Una implementación completa debería incluir:

- Un método para que los jugadores o el dueño puedan aportar fondos al pozo antes de iniciar la partida.
- Una precondición que impida avanzar en el juego si el pozo está vacío.

Para generar un bug, se implementará una precondición sobre el método `determineWinner` que verifique que el pozo no esté vacío. Sin embargo, se permitirá iniciar el juego tanto con pozo vacío como lleno, permitiendo que el contrato tenga algunas ejecuciones correctas y otras incorrectas.

En el contrato en Solidity, esto se haría agregando lo siguiente en la función `determineWinner`:

Solidity

```
    function determineWinner() public {
        require(address(this).balance > 0, "Pot must not be empty");
        //BUG AGREGADO: se exige balance > 0 solo al momento de determinar ganador

        if(p1Choice != -1 && p2Choice != -1) {
            uint winner = payoffMatrix[uint(p1Choice)][uint(p2Choice)];
            if(winner == 1) {
                player1.transfer(address(this).balance);
            }
            else if(winner == 2) {
                player2.transfer(address(this).balance);
            }
            else {
                owner.transfer(address(this).balance);
            }
        }
        else {
            revert();
        }
    }
```

Otra forma de lograr el mismo efecto es agregando otro `if` con un `revert()` en el `else`, como en la precondición existente en la función.

En la abstracción en Alloy, esto se traduce en agregar la siguiente precondición:

Alloy

```
pred pre_determineWinner[e:EstadoConcreto] {
  e._balance[Address0x0] > 0
  //BUG AGREGADO: se exige balance > 0 solo al momento de determinar ganador
  e._p1Choice != -1 and e._p2Choice != -1
  e._init = True
}
```

Por otro lado, la abstracción actual contiene un invariante que exige que todos los balances sean 0 en todo momento.

Este invariante es llamativo, ya que anula completamente el propósito del pozo. Una explicación posible, entendiendo que el contrato está simplificado, es que mantener el pozo en 0 achica el espacio de estados, y permite enfocar el análisis en la lógica del juego.

Dado que se planea implementar el uso del pozo, se eliminará este invariante:

Alloy

```
pred invariante[e:EstadoConcreto] {
  e._init = True
  e._p1Choice >= -1 and e._p1Choice <= 2
  e._p2Choice >= -1 and e._p2Choice <= 2

  e._payoffMatrix = 0->0->0 + 0->1->2 + 0->2->1 + 1->0->1 + 1->1->0 + 1->2->2 +
      ↪2->0->2 + 2->1->1 + 2->2->0

  some e._balance
  #e._balance = 4
  // PRECONDICIÓN ELIMINADA: no se exige que todos los balances sean 0
  // (all a:Address | e._balance[a] = 0)
  e._player1 != e._player2
  e._player1 != e._owner
  e._player2 != e._owner
  e._player1 != Address0x0
  e._player2 != Address0x0
  e._owner != Address0x0
}
```

Al hacer estas modificaciones, se espera que el diagrama del contrato editado incluya transiciones *may* al ejecutar `determineWinner`. Esto indicará que la transición puede ocurrir solo en algunos estados concretos y no en otros, dependiendo del estado del pozo.

5.3.3. Comparación de los resultados

Se ejecutaron ambos contratos (el original y la variante con bug) utilizando la herramienta Alloy4PA. Estos son los diagramas de transición obtenidos:

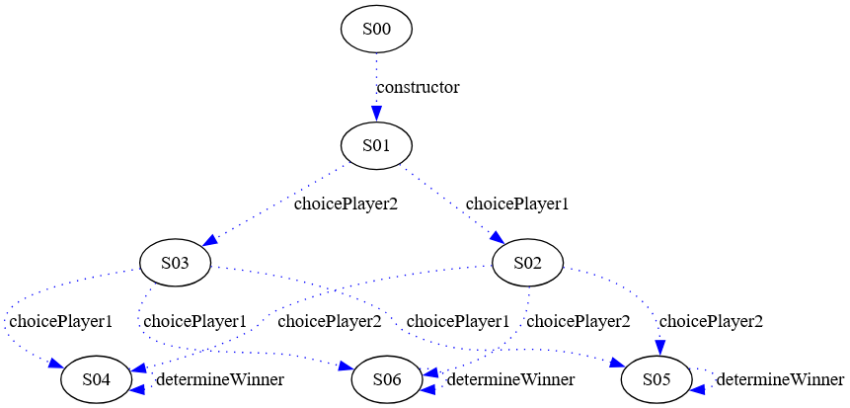


Fig. 5.2: Diagrama de transición del contrato Rock-Paper-Scissors original.

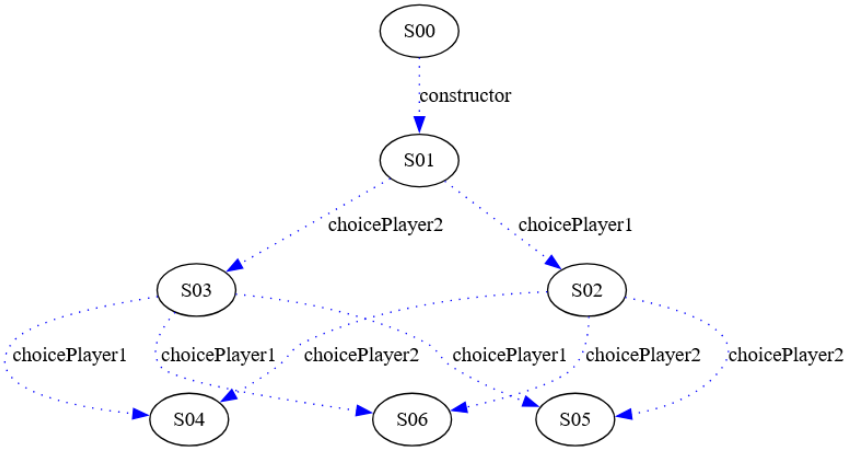


Fig. 5.3: Diagrama de transición del contrato Rock-Paper-Scissors con bug.

Se puede observar un resultado inesperado: en el diagrama del contrato con bug, las transiciones `determineWinner` desaparecen en vez de convertirse en *may*. Es decir que, en vez de tener estados donde la transición puede o no ocurrir como se esperaba, la transición directamente no es posible.

Esto indica que existe un elemento en el código que es más restrictivo de lo previsto.

5.3.4. Refinamiento de la variante

Analizando el código de la transición `determineWinner`, surge un problema en las post-condiciones existentes. El código de la abstracción exige que se cumplan simultáneamente dos condiciones contradictorias sobre los balances.

Este es el fragmento relevante del código:

Alloy

```
1   pred met_determineWinner[ein: EstadoConcreto, eout: EstadoConcreto, sender: Address]
        ↪{
2
3     (...)
```

```
4
5     //POST
6     (let winner = (ein._payoffMatrix[ein._p1Choice])[ein._p2Choice] |
7       (winner = 1) => eout._balance = ein._balance++Address1->ein._balance[Address0x0]
8       else
9         (winner = 2) => eout._balance = ein._balance++Address2->ein._balance[Address0x0
              ↪]
10        else
11        (winner = 0) => eout._balance = ein._balance++ein._owner->ein._balance[
              ↪Address0x0]
12    )
13
14    eout._balance = ein._balance++Address0x0->0
15
16    (...)
17  }
```

Tomando de ejemplo el caso `winner = 1`, y extendiendo a los otros casos:

- Dentro del condicional, hay una postcondición que dice que el balance global de salida es igual al de entrada, pero con el balance del jugador 1 incrementado con el valor del pozo.
- Después del condicional, hay otra que dice que el balance global de salida es igual al de entrada, pero con el balance del pozo puesto en 0.

Nótese que la primera postcondición afirma que todos los otros balances permanecen iguales, incluyendo el del pozo. Y la segunda afirma que todos los otros balances permanecen iguales, incluyendo el del jugador 1.

La única forma de satisfacer todas estas condiciones a la vez es empezando el juego con todos los balances en 0.

Es probable que la intención original haya sido actualizar el balance del ganador, vaciar el pozo, y mantener los otros balances iguales. En ese caso, lo correcto habría sido escribir una sola igualdad reflejando el estado completo del balance de salida.

Este problema no afectó al contrato original, ya que forzaba todos los balances a 0. Pero al eliminar esa restricción, se manifestó el bug.

Se propone una nueva versión del código de arriba, que refleje esta intención:

Alloy

```
1   // POST
2   let winner = (ein._payoffMatrix[ein._p1Choice])[ein._p2Choice] |
3     let winnerAddress =
4       (winner = 1) => ein._player1
5       else (winner = 2) => ein._player2
6       else ein._owner |
7     // actualizar ganador y vaciar el pozo en una sola igualdad
8     eout._balance = (
9       (ein._balance) ++ (winnerAddress -> ein._balance[Address0x0])
10    ) ++ (Address0x0 -> 0)
```

Al hacer esta segunda modificación, se espera que esta vez el diagrama incluya las transiciones *may* que se buscaban originalmente.

5.3.5. Resultados de la variante refinada

Se ejecutó la nueva variante utilizando la herramienta Alloy4PA. Este es el diagrama de transición obtenido:

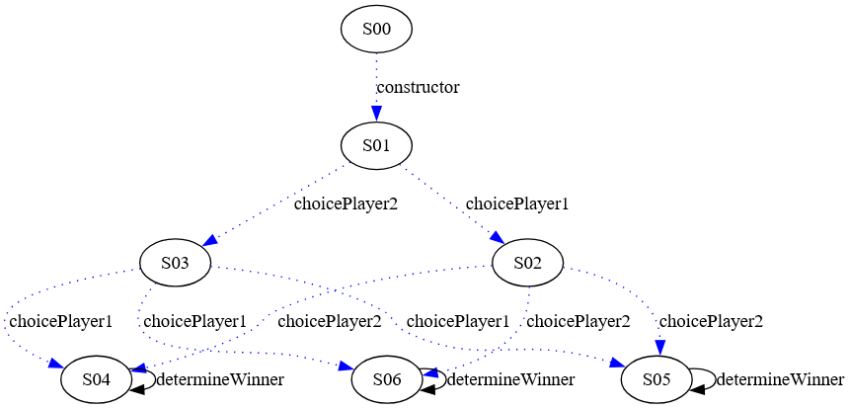


Fig. 5.4: Diagrama de transición del contrato Rock-Paper-Scissors con bug refinado.

En este caso, las transiciones `determineWinner` vuelven a estar presentes, y ahora son de tipo *may*, como se esperaba. Esto indica que la transición puede ocurrir en algunos estados concretos y no en otros.

Se pueden entender dos causas que llevan a estados concretos donde la transición no puede ejecutarse:

- Si el juego comienza con el pozo en 0, la precondición agregada en la primera modificación impide ejecutar la transición.
- Si el juego comienza con un pozo mayor a 0, la transición se puede ejecutar. Al hacerlo, el contrato permanece en el mismo estado abstracto, cambiando su estado concreto a uno con el pozo en 0. En este nuevo estado concreto, la precondición agregada impide volver a ejecutar la transición.

Cualquiera de estas dos situaciones es suficiente para que la transición pase a ser de tipo *may* en el diagrama.

En caso de continuar con la implementación del uso del pozo, sería necesario entender si tiene sentido que el método `determineWinner` pueda ejecutarse múltiples veces, ya que afecta los balances.

5.4. AssetTransfer

5.4.1. Introducción al experimento

Este ejemplo, junto con los dos siguientes, está relacionado con pruebas de introducción de distintos *bugs* al contrato *AssetTransfer* del *Benchmark#1* o a variantes reducidas del mismo, con el objetivo de analizar cómo se reflejan en los resultados de *Alloy4PA*. La idea original era lograr que el grafo resultante mostrara alguna transición *may* en lugar de que todas fueran *must*. El primer *bug* introducido se intuía que no produciría ese efecto, pero se implementó para observar un primer impacto sobre el contrato. El segundo caso se plantea como un paso intermedio hacia el tercero, en el cual finalmente se consiguió la presencia de una transición *may*.

5.4.1.1. Especificaciones del sistema donde se probarán estos escenarios

Las pruebas se ejecutaron en una máquina con las siguientes características:

- **Sistema Operativo**: Windows 11 Pro
- **Procesador**: Intel Core i7 (8 núcleos lógicos)
- **Memoria RAM**: 16 GB
- **Docker**: Docker Desktop 28.5.1
- **Java**: OpenJDK 18 (dentro del contenedor Docker)
- **Alloy4PA**: Versión 1.0.0 ejecutada en contenedor Linux

5.4.2. Contrato original

El contrato *AssetTransfer* modela la venta de un activo digital entre dos participantes (*owner*) y (*buyer*). Inicialmente, el activo pertenece a un propietario (*owner*), y el proceso comienza cuando este pone el activo en estado *Active* con una descrición y un precio esperado. En esta etapa, el vendedor ((*owner*)) puede modificar las características del activo (descripción y precio esperado) mediante la transición `modifyOffer`, siempre que el contrato continúe en *Active*.

En este estado, un posible comprador puede realizar una oferta (`placeOffer`), pasando el contrato al estado *OfferPlaced*. A partir de allí, se solicitan una inspección (`markInspected`) y una tasación (`markAppraised`) al inspector y al tasador definidos al momento de realizar la oferta. Estas acciones actualizan el estado a *Inspected* y *Appraised*, respectivamente.

Existen tres formas de no completarse la transferencia (dos que "reinician" el contrato y una que lo finaliza):

- `reject`: utilizada por el vendedor para rechazar la oferta y devolver el activo al estado *Active*, permitiendo recibir nuevas propuestas.
- `rescind`: utilizada por el comprador para retractar la oferta y devolver el activo al estado *Active*, permitiendo también al vender recibir nuevas ofertas.
- `terminate`: puede ser invocada por el *owner* cuando desea cancelar definitivamente el proceso, llevando el contrato al estado *Terminated*.

Pero si ambas partes aceptan (`buyerAccept` y `sellerAccept`), el contrato alcanza el estado *Accepted*, reflejando una compraventa exitosa.

5.4.3. Bug en la función `reject()`

5.4.3.1. Descripción del bug implementado

Se introduce un issue muy pequeño, que pareciera que casi podría no tener ningún impacto dado que el `buyer` se vuelve a setear en `makeOffer`, que consiste en **no resetear** la variable `_instanceBuyer` cuando se rechaza una oferta. Pero que se analizara si se ve reflejado en el diagrama de estados que genere la herramienta.

En el comportamiento correcto, la función `reject()` debe:

1. Cambiar el estado del contrato a `Active`
2. Resetear `_instanceBuyer` a `Address0x0` (null)

3. Resetear `_offerPrice` a 0

En la versión buggy, el segundo paso se omite, manteniendo incorrectamente la referencia al comprador anterior.

5.4.3.2. Código de la función reject() buggy

Alloy

```
pred met_reject[ein: EstadoConcreto, eout: EstadoConcreto, sender: Address] {
    //Pre
    pre_reject[ein]
    pre_params_reject[ein, sender]

    //Post
    // eout._instanceBuyer = Address0x0 // BUG: No se resetea _instanceBuyer (debería ser Address0x0)
    eout._state = Active
    eout._offerPrice = 0

    // Campos que se mantienen
    eout._instanceOwner = ein._instanceOwner
    eout._askingPrice = ein._askingPrice
    eout._description = ein._description
    eout._instanceBuyer = ein._instanceBuyer // BUG: Mantiene el buyer anterior
    eout._instanceInspector = ein._instanceInspector
    eout._instanceAppraiser = ein._instanceAppraiser
    eout._init = ein._init
}
```

5.4.3.3. Estrategia de detección: invariante adicional

Para exponer el bug, se agregó la siguiente condición al invariante del modelo (que faltaban en el modelo original):

Alloy

```
pred invariante[e:EstadoConcreto] {
    e._init = True
    e._instanceOwner != Address0x0
    e._instanceOwner != e._instanceBuyer
    e._offerPrice >= 0
    e._askingPrice >= 0
    e._instanceBuyer = Address0x0 implies (e._state = Active or e._state = Terminated)
    e._instanceInspector = Address0x0 implies (e._state = Active or e._state = Terminated)
    e._instanceAppraiser = Address0x0 implies (e._state = Active or e._state = Terminated)
    //FIXED: new condition that was missing:
    e._state = Active implies e._instanceBuyer = Address0x0
}
```

Esta condición establece que **si el estado es Active, entonces el buyer debe ser null**. Esta es una regla de negocio lógicamente correcta, ya que el estado Active indica

que el contrato está esperando ofertas.

5.4.3.4. Configuración del experimento

Se crearon dos versiones del modelo AssetTransfer:

- **AssetTransfer_original.als**: Modelo original, **sin** el invariante adicional y que **sí resetea** el vendedor en reject
- **AssetTransfer_bug.als**: Modelo con el bug, pero **con** el invariante adicional y **sin resetear** el vendedor en reject

5.4.3.5. Tiempos de ejecución

Ambos modelos se ejecutaron usando Alloy4PA con Docker:

Model	**EPA (s)**	**BEPA (s)**	**SBEPA (s)**	**Total (s)**
AssetTransfer_original	13.437	774.799	0.460	788.696
AssetTransfer_bug_reject	11.324	733.852	0.333	745.509

Tab. 5.1: Execution times of Alloy4PA for the original model and the bug-in-reject variant.

El modelo con el bug fue ligeramente más rápido (745s vs 788s), posiblemente debido a que el invariante adicional reduce el espacio de búsqueda al eliminar estados inconsistentes tempranamente en el análisis.

5.4.3.6. Grafo modal completo

A continuación, la figura 5.5 muestra el grafo de la versión original del contrato:

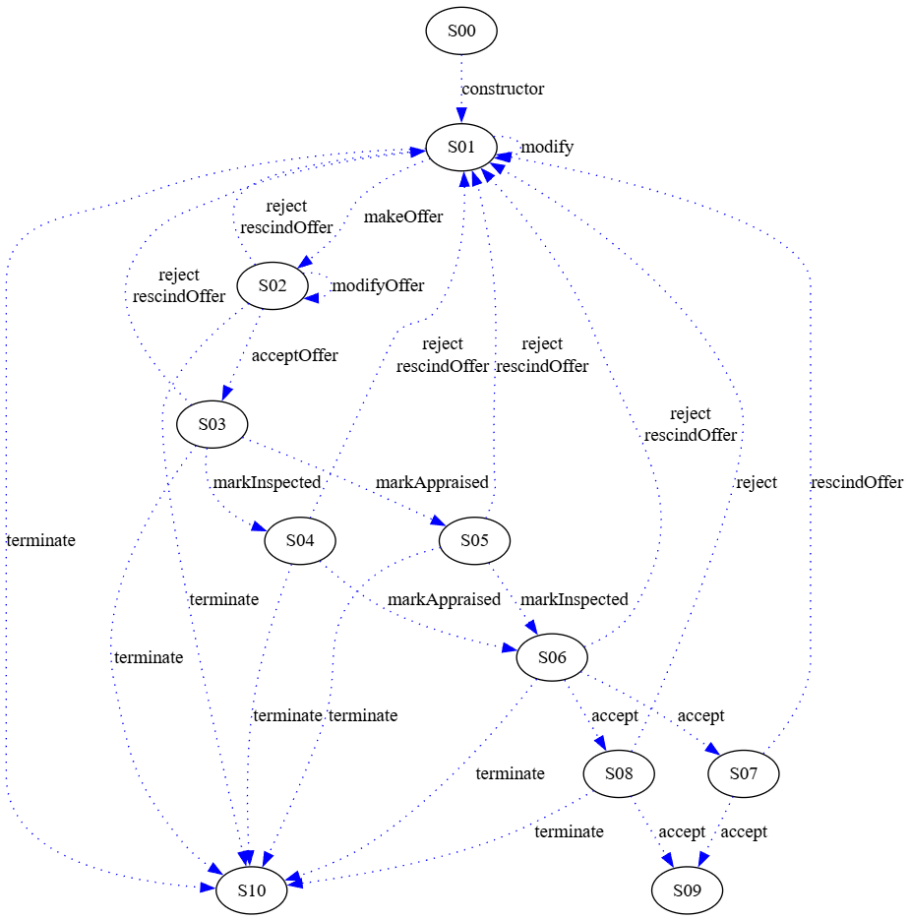


Fig. 5.5: Grafo modal completo del contrato AssetTransfer original, con todas las transacciones reject presentes.

Mientras que la siguiente figura 5.6 muestra el grafo modal completo correspondiente al modelo con el bug en `reject()`. Todas las transiciones existentes se manetiene como MUST (líneas azules punteadas), pero deja de haber posibles transacciones `reject()`.

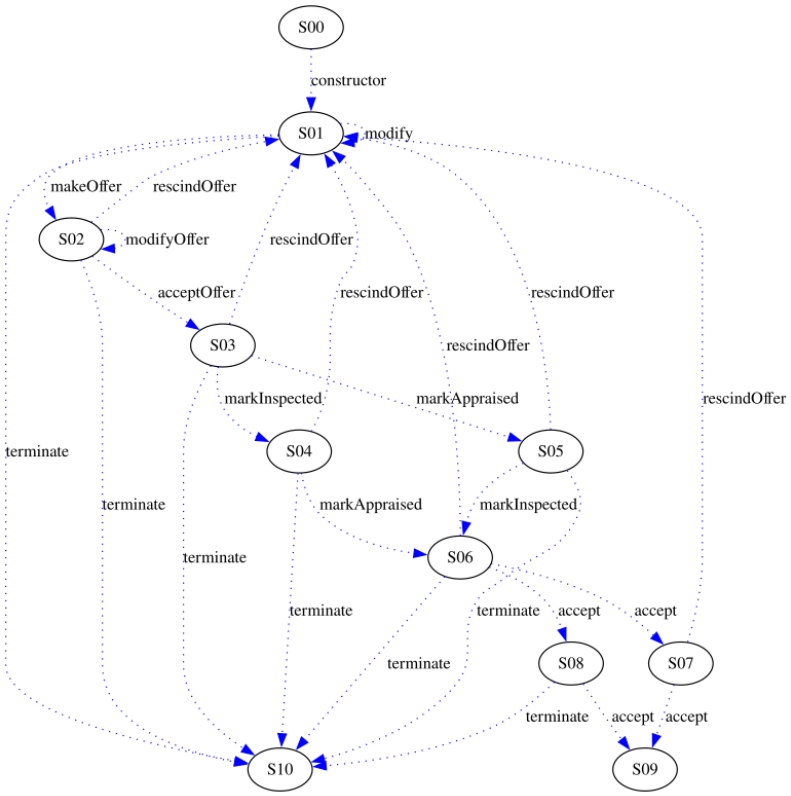


Fig. 5.6: Grafo modal completo del contrato AssetTransfer con bug en `reject()`. La eliminación de transiciones `reject` al agregar el invariante expone el defecto.

5.4.3.7. Análisis de las abstracciones modales obtenidas

Modelo original (sin invariante): El modelo original genera 27 transiciones must, incluyendo múltiples transiciones `reject` desde varios estados.

Modelo con bug: Al agregar el bug y el invariante, Alloy4PA detecta el conflicto y modifica significativamente las abstracciones:

- S02 → S01 **(sin `reject`)**
- S03 → S01 **(sin `reject`)**
- S04 → S01 **(sin `reject`)**
- S05 → S01 **(sin `reject`)**
- S06 → S01 **(sin `reject`)**
- S08 → S01 **(eliminada completamente, ya que solo era posible por `reject`)**

5.4.3.8. Interpretación de los resultados

El bug con el invariante adicional se expuso exitosamente en el grafo de la siguiente manera:

1. **Flujo normal**: makeOffer (Active → OfferPlaced, buyer = AddressX)
2. **Problema**: reject (OfferPlaced → Active, buyer sigue siendo AddressX)
3. **Conflicto**: Estado Active + buyer != null **viola el invariante**
4. **Resultado**: La transición reject se vuelve inconsistente y desaparece

5.4.3.9. Valor de las abstracciones modales

Este experimento demuestra que:

- **Con invariantes correctos**: Las abstracciones modales detectan automáticamente inconsistencias
- **Complementariedad**: Los invariantes y las abstracciones se refuerzan mutuamente

5.4.3.10. Conclusiones

Este nuevo caso de estudio confirma la efectividad de Alloy4PA para la detección de bugs, específicamente:

1. **Detección automática**: Una vez definidos los invariantes apropiados, la herramienta detecta automáticamente las inconsistencias
2. **Bugs sutiles**: Puede detectar errores que no causan fallos inmediatos pero violan la lógica de negocio
3. **Importancia de invariantes**: Los invariantes correctos son cruciales para exponer bugs latentes
4. **Validación práctica**: Confirma la utilidad de la técnica en posibles casos reales de contratos inteligentes

Este experimento aporta evidencia empírica adicional sobre la efectividad de Alloy4PA más allá de los casos reportados en el paper original, demostrando su posible aplicabilidad en escenarios de desarrollo y auditoría de contratos inteligentes.

5.4.4. Bug en `makeOffer` con EPA y particiones de estado

Esta sección documenta un caso de estudio basado en una versión reducida de las funciones del contrato original *AssetTransfer*, que conserva únicamente el constructor, el invariante y las transiciones `makeOffer`, `terminate` y `accept`. Asimismo, se redujo el número de estados a cuatro (*Active*, *OfferPlaced*, *Accepted* y *Terminated*), eliminando los atributos de descripción y el rol de inspector. Con este modelo simplificado se busca analizar cómo una falla en la lógica de negocio de la función `makeOffer` del contrato se refleja en la abstracción modal generada por *Alloy4PA*.

Para ello, en este caso vamos a utilizar EPA.

5.4.4.1. Extended Predicate Abstraction (EPA)

La *Extended Predicate Abstraction* (EPA) es la técnica que utiliza *Alloy4PA* para generar una versión simplificada del contrato, capaz de representar todos sus posibles comportamientos de manera finita y comprensible. En lugar de analizar todos los estados

concretos del contrato (lo que sería imposible por su tamaño), EPA agrupa los estados en un número reducido de categorías llamadas *particiones*, basadas en el valor de ciertos predicados lógicos relevantes.

Idea general Cada *predicado* representa una condición importante del contrato, como por ejemplo si el contrato está activo, si hay una oferta, o si el comprador ya aceptó. Al combinar estos predicados en todas sus formas posibles (verdadero o falso), se generan las distintas particiones de la abstracción.

El número de predicados seleccionados se denomina N. A partir de ellos, EPA genera automáticamente todas las combinaciones posibles, obteniendo un total de 2^N particiones o estados abstractos. Por ejemplo, con dos predicados se obtienen $2^2 = 4$ combinaciones, y con tres predicados se generan $2^3 = 8$ estados abstractos.

El rol del invariante El predicado `invariante[e]` define las condiciones que deben cumplirse en todos los estados válidos del contrato. Alloy4PA incluye automáticamente este invariante dentro de cada partición generada para asegurar que todas las configuraciones abstractas representen estados coherentes. Opcionalmente, puede agregarse un estado especial que viole el invariante (útil para detectar fallas de seguridad o de lógica), configurando:

```
Inv = INV
```

Predicados y particiones Los predicados que definen las condiciones relevantes del contrato se declaran en el modelo Alloy con la forma:

```
pred pre_makeOffer[e: EstadoConcreto] { e._state = Active }
```

En el archivo de configuración (`.ini`), se listan los predicados que Alloy4PA debe usar para generar las combinaciones, por ejemplo:

```
Predicates = pre_constructor[e], pre_makeOffer[e], pre_accept[e]
```

Cada una de esas condiciones podrá ser verdadera o falsa, y la combinación de sus valores generará las distintas particiones del modelo abstracto.

Configuración de EPA En el archivo `.ini`, la línea principal que habilita este modo es:

```
EPA = True
```

Cuando EPA está activado:

- Alloy4PA genera automáticamente todas las combinaciones posibles de los predicados listados, produciendo 2^N particiones.
- El parámetro `States` se ignora, ya que el número de estados se calcula automáticamente.
- El nombre base de las particiones se controla con `PartitionName`, por ejemplo:

  ```
  PartitionName = partition
  ```

Si en cambio se desactiva EPA:

```
EPA = False
```

entonces las particiones deben escribirse manualmente en el archivo Alloy, por ejemplo:

```
pred partitionS00[e: EstadoConcreto] { pre_constructor[e] }
pred partitionS01[e: EstadoConcreto] { not pre_constructor[e] }
```

y el número de estados debe declararse explícitamente:

```
States = 6
```

Diferencia práctica entre EPA activado y desactivado

- Con `EPA = True`, el usuario define solo los predicados principales y Alloy4PA genera automáticamente todas las combinaciones posibles. Es ideal para análisis exploratorios o cuando se busca detectar diferencias entre comportamientos posibles (*MAY*) y garantizados (*MUST*).
- Con `EPA = False`, el usuario tiene control total sobre las particiones que se evaluarán. Es más adecuado para modelos grandes o muy específicos, donde el número de combinaciones posibles sería excesivo.

Tab. 5.2: Matriz de casos EPA vs `Predicates` en Alloy4PA

Caso	Configuración	Resultado	Uso recomendado
1	`EPA = True` Lista `Predicates` definida	Genera 2^N combinaciones (una por cada subconjunto de predicados).	Análisis modal completo con pocos predicados ($N \leq 5$). Ideal para detectar diferencias entre transiciones *MUST* y *MAY*.
2	`EPA = True` `Predicates` vacío	Inconsistente: no se generan particiones útiles.	**Evitar.** Siempre incluir al menos un predicado (por ejemplo, el del constructor).
3	`EPA = False` `Predicates` vacío	Usa las particiones manuales (`partitionSXX[e]`) definidas en el modelo.	Recomendado cuando se desea control total sobre los estados o cuando N es grande.
4	`EPA = False` Lista `Predicates` definida	La lista de predicados es ignorada; solo se consideran las particiones manuales.	Evitar: puede causar confusión en el análisis.

Procedimiento resumido de uso

1. Definir los predicados principales del contrato en Alloy (por ejemplo, `pre_makeOffer`, `pre_accept`, etc.).
2. Escribir el invariante del contrato (`invariante[e]`).
3. Configurar el archivo `.ini` indicando si se usará EPA y qué predicados incluir.
4. Ejecutar Alloy4PA para generar el grafo modal y analizar las transiciones *MAY* y *MUST*.

Interpretación de resultados El análisis con EPA produce un grafo donde cada nodo representa una combinación posible de condiciones del contrato, y cada flecha representa una posible ejecución de una función.

En resumen, EPA permite pasar de un modelo complejo y de muchos estados concretos a una representación más manejable que muestra de forma visual y exhaustiva el comportamiento global del contrato.

5.4.4.2. Bug modelado

Se introduce un cambio en la transición `makeOffer`, relacionado con la comparación entre el precio ofertado por el comprador y el precio originalmente esperado por el vendedor. Si el precio ofertado es mayor, se asigna un tasador; en cambio, si el precio es menor o igual, no se realiza la asignación (se conserva el tasador previamente establecido, que al provenir del estado *Active* implica que nunca llega a asignarse ninguno).

Cambio en el contrato de Solidity:

Solidity

```
function MakeOffer(address appraiser, uint256 offerPrice) public
{
        if (appraiser == 0x0000000000000000000000000000000000000000 offerPrice == 0)
        {
                revert();
        }
        if (State != StateType.Active)
        {
                revert();
        }
        if (InstanceOwner == msg.sender)
        {
                revert();
        }

        InstanceBuyer = msg.sender;

        if (offerPrice > AskingPrice) { InstanceAppraiser = appraiser; } // BUG: solo
            ↪se setea si el precio ofrecido es mayor, caso contrario no hace nada

        OfferPrice = offerPrice;
        State = StateType.OfferPlaced;
}
```

Predicado Alloy de makeOffer:

Alloy

```
pred met_makeOffer[ein,eout:EstadoConcreto, offerPrice:Int] {
        //Pre
        pre_makeOffer[ein]
        pre_params_makeOffer[ein, sender, appraiser, offerPrice]

        //Post
```

```
7         eout._instanceBuyer = sender
8         eout._offerPrice = offerPrice
9         eout._state = OfferPlaced
10        eout._instanceOwner = ein._instanceOwner
11        eout._askingPrice = ein._askingPrice
12        eout._init = ein._init
13
14        // BUG: Solo asigna appraiser si la oferta es MAYOR que el asking price
15        (offerPrice > ein._askingPrice) implies eout._instanceAppraiser = appraiser
16        (offerPrice <= ein._askingPrice) implies eout._instanceAppraiser = ein.
              ↪_instanceAppraiser // BUG: NO asigna appraiser
17  }
```

Predicados Alloy añadidos:

Alloy

```
1   pred pre_activated[ein: EstadoConcreto] {
2     ein._init = True
3     ein._state = Active
4   }
5
6   pred pre_offerPlaced[ein: EstadoConcreto] {
7     ein._init = True
8     ein._state = OfferPlaced
9   }
10
11  pred pre_offerAccepted[ein: EstadoConcreto] {
12    ein._init = True
13    ein._state = Accepted
14  }
15
16  pred pre_terminated[ein: EstadoConcreto] {
17    ein._init = True
18    ein._state = Terminated
19  }
```

Predicados Alloy modificados para dejar más claro que se necesita tener el tasador definido:

Alloy

```
1   pred pre_accept[ein: EstadoConcreto] {
2         ein._init = True
3         ein._state = OfferPlaced
4         ein._instanceBuyer != Address0x0
5         ein._instanceAppraiser != Address0x0 // añadido para llevar a estados que
              ↪rompan el invariante
6   }
7
8   pred pre_terminate[ein: EstadoConcreto] {
9         ein._init = True
10        some ein._state
11        ein._state != Terminated
```

```
12          ein._state != Accepted
13          ein._state = OfferPlaced implies ein._instanceAppraiser != Address0x0 // añ
                ↪adido para llevar a estados que rompan el invariante
14  }
```

5.4.4.3. Configuración empleada

Archivo `AssetTransfer_bug_wEPA_pre_statesConfig.ini`:

bash

```
States = 4
Predicates = pre_activated[e],pre_offerPlaced[e],pre_offerAccepted[e]
Scope = 8 EstadoConcreto, 8 Int, 5 Address
EPA = True
Divide_states_by = 2
Inv = INV
```

Se usan tres predicados de estado genéricos para reducir combinaciones ($2^3 = 8$ particiones base) y mantener tiempos manejables. El invariante `INV` deliberadamente *no* descarta las ramas defectuosas: así aparece como estado alcanzable. No se consideró pre_terminated solamente para reducir la cantidad de particiones y porque pre_offerAccepted iba a tener el mismo comporatamiento.

5.4.4.4. Objetivo del experimento

Mostrar que la función `makeOffer` conduce a dos destinos *must* distintos: uno en el que luego puede ejecutarse `accept` (flujo sano) y otro que se convierte en un **estado sin salida** donde la oferta queda "a medias" y no puede jamás aceptarse ni terminarse. La coexistencia de ambos destinos como *must* revelaría no sólo pérdida de garantía de progreso, sino la presencia explícita de un *estado corrupto estancado*.

5.4.4.5. Grafo modal del resultado

Graphviz DOT

```
digraph {

S00 [label="Init"]
S04 [label="makeOffer & terminate\n & !accept & pre_activated[e]"]
S10 [label="!makeOffer & terminate\n & accept & pre_offerPlaced[e]"]
S16 [label="!makeOffer & !terminate\n & !accept & pre_offerPlaced[e]"]
S24 [label="!makeOffer & !terminate\n & !accept & pre_offerAccepted[e]"]


S00->S04 [label="constructor", style="dotted", color="blue"]
S10->S24 [label="accept", style="dotted", color="blue"]
S04->S10 [label="makeOffer", style="dotted", color="blue"]
S04->S16 [label="makeOffer", style="dotted", color="blue"]
}
```

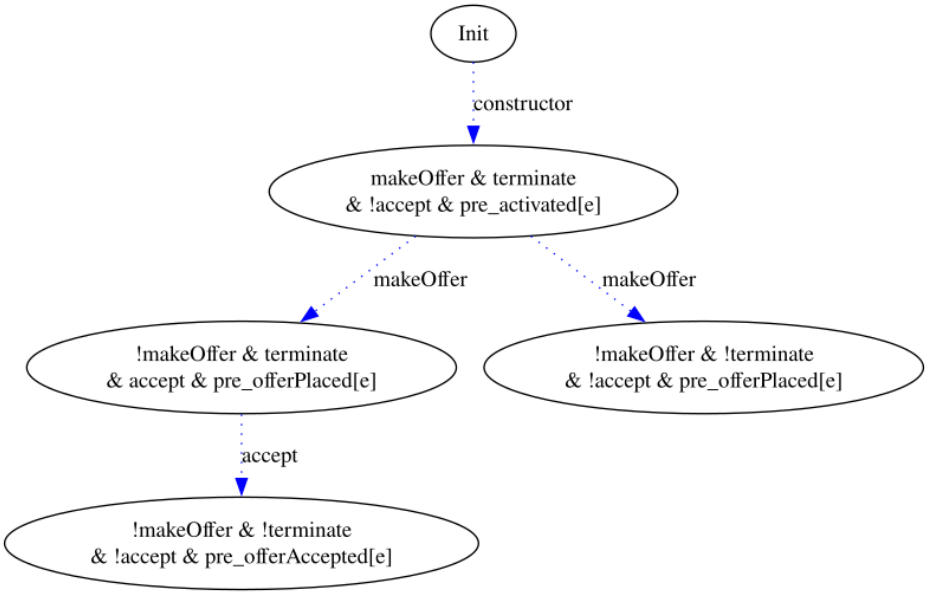


Fig. 5.7: Abstracción modal del ejemplo simplificado: bifurcación MUST de `makeOffer` hacia estado aceptable (S10) y estado estancado (S16).

5.4.4.6. Estados abstractos relevantes

La salida modal (archivo `forked_arrows.dot`) contiene estos nodos principales:

- S00: Init
- S04: Partición activa previa a ofertas (`pre_activated`) donde `makeOffer` es ejecutable
- S10: Oferta colocada .ªceptable"(`accept` habilitado)
- S16: Oferta colocada defectuosa (`accept` imposible)
- S24: Oferta aceptada (estado posterior al `accept`)

5.4.4.7. Transiciones MUST observadas

Todas las transiciones listadas aparecen como MUST (estilo punteado azul):

- constructor: S00 → S04
- makeOffer: S04 → S10 (rama sana)
- makeOffer: S04 → S16 (rama defectuosa)
- accept: S10 → S24

Esto significa que **desde la abstracción** se considera garantizado poder caer en la rama defectuosa. Allí se materializa un **estado sin salida**: desde S16 no existe ninguna transición (ni `accept`, ni `terminate`, ni otra `makeOffer`) que permita recuperar el flujo. El síntoma del bug deja de ser sólo pérdida de universalidad; es la aparición de un punto de estancamiento permanente.

5.4.4.8. Costos de cómputo

Resumen de tiempos (archivos `queriesTime_CLASSIC.csv` y `queriesTime_BLUE.csv`):

Phase	**Total Queries**	**Total Time (s)**
CLASSIC (EPA + partitions)	133	80.095
BLUE (MUST)	4	820.508
SUPERBLUE (Hyper-MUST)	0	0.000

Tab. 5.3: Number of queries and total execution time per analysis phase.

El mayor costo proviene de intentar confirmar o refutar universalidad en transiciones MUST (BLUE). Reducir predicados de estado fue clave para mantener el experimento viable.

5.4.4.9. Conclusión del ejemplo: Estado atascado y corrupción de la oferta

El nodo S16 representa una oferta colocada que nunca llega a condiciones de aceptación y que tampoco puede cerrarse/terminarse. En el grafo modal:

- S16 no posee aristas salientes: evidencia formal de que el modelo permite un ciclo de vida incompleto.
- La etiqueta interna ("`!makeOffer & !terminate & !accept & pre_offerPlaced[e]`") confirma ausencia de habilitación de funciones críticas.
- La abstracción lo marca como MUST porque la partición no discrimina el subcaso con tasador inválido; por eso el estado degradado se integra como garantizado.

Este patrón expone directamente una **violación de progreso (liveness)** y una **corrupción de la semántica de la oferta**, ya que queda persistentemente en un limbo operativo.

5.4.5. Variante con transición `makeOffer` *must* y `accept` *may*

Esta variante se basa en el mismo caso reducido anterior de `AssetTransfer`, manteniendo también el *bug* en `makeOffer`, consistente en la omisión de la asignación correcta del tasador cuando el precio ofertado es menor o igual al esperado por el vendedor. Sin embargo, a diferencia del caso previo, aquí se adopta un particionado manual (`EPA = False`) con el fin de generar una mezcla de estados concretos válidos y defectuosos dentro de una misma partición genérica **OfferPlaced**. El refuerzo del invariante mantiene a `makeOffer` como transición *must*, mientras que `accept` y `terminate` desde **OfferPlaced** quedan solo como *may*.

5.4.5.1. Objetivo del experimento

Mostrar el contraste entre:

- Caso bifurcado (`EPA = True`): una misma transición *must* (`makeOffer`) conduce a dos destinos distintos, uno de los cuales corresponde a un estado sin salida.

- Caso mezclado (`EPA = False`): la universalidad (*must*) de `makeOffer` se mantiene, pero se pierde la garantía en `accept`, que pasa a ser *may*.

5.4.5.2. Configuración empleada

Archivo `AssetTransfer_bug_wMAY_transitionConfig.ini`:

bash

```
Inv =      #(vacío: la herramienta no agrega estado de quiebre de invariante)
States = 4
Scope = 8 EstadoConcreto, 8 Int, 5 Address
EPA = False
Predicates =
Divide_states_by = 2
```

El invariante reforzado es:

Alloy

```
pred invariante[e:EstadoConcreto] {
    e._init = True
    e._instanceOwner != Address0x0
    e._instanceOwner != e._instanceBuyer
    e._offerPrice >= 0
    e._askingPrice >= 100

    e._instanceBuyer = Address0x0 implies (e._state = Active or e._state = Terminated)
    //FIXED: nueva condición que faltaba en el contrato original
    e._state = Active implies e._instanceBuyer = Address0x0
}
```

Esto hace que todos los estados Active en la partición satisfagan `pre_makeOffer`, manteniendo dicha transición en **must**.

Particiones definidas para este caso EPA=False:

Alloy

```
// Initial partition for EPA=True: only the constructor precondition state
pred partitionS00[e: EstadoConcreto]{ //
       e._init = False
}

pred partitionS01[e: EstadoConcreto]{ //
       (invariante[e])
       e._state = Active
}

pred partitionS02[e: EstadoConcreto]{ //
       (invariante[e])
       e._state = OfferPlaced
}

pred partitionS03[e: EstadoConcreto]{ //
```

```
17          (invariante[e])
18          e._state = Accepted
19  }
20
21  pred partitionS04[e: EstadoConcreto]{ //
22          (invariante[e])
23          e._state = Terminated
24  }
```

5.4.5.3. Transiciones observadas

Grafo modal generado (archivo `AssetTransfer_bug_wMAY_transitionforked_arrows.dot`):

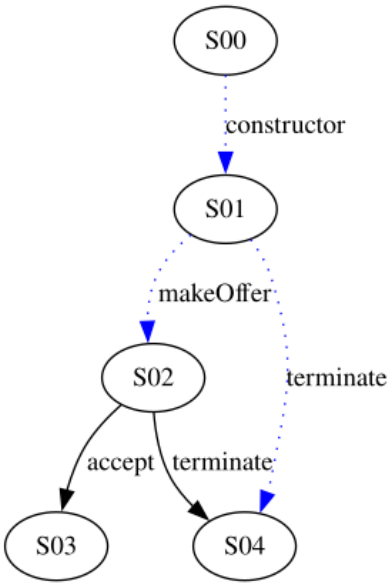


Fig. 5.8: Abstracción modal de la variante mezclada: `makeOffer` recupera **must**; `accept` y `terminate` (OfferPlaced) quedan como **may** por la mezcla de casos sanos y buggy.

- **must**: `constructor`, `makeOffer`, `terminate` (Active → Terminated).
- **may**: `accept` y `terminate` desde **OfferPlaced** (mezcla de estados concretos con tasador válido e inválido).

5.4.5.4. Interpretación modal

La universalidad de `makeOffer` se mantiene porque la partición **Active** solo contiene estados concretos con `_instanceBuyer = Address0x0`. Sin embargo, `accept` se degrada a *may*, ya que en la partición **OfferPlaced** conviven estados donde el tasador quedó sin asignar (bug) junto con estados válidos. Por este motivo, las precondiciones de `accept` y `terminate` dejan de ser universales.

5.4.5.5. Comparativa con variante EPA

- **EPA + bifurcación**: el bug aparecía como un segundo destino *must* estancado (estado sin salida), exponiendo una violación de progreso de forma directa.

- **Partición mezclada**: el estado atascado deja de mostrarse como nodo separado, pero se evidencia una pérdida de garantía, ya que `accept` deja de ser *must* (riesgo de oferta que no progresa hacia la aceptación).

En una auditoría, disponer de ambas vistas puede ser útil: la bifurcación muestra de forma explícita la naturaleza del estancamiento, mientras que la mezcla cuantifica su impacto al degradar la universalidad de dos funciones críticas.

5.4.5.6. Costos de cómputo

El análisis con particionado manual (`EPA = False`) presentó un costo computacional sensiblemente menor que el del caso con generación automática de particiones. Al no requerir la expansión combinatoria de 2^N estados derivados de los predicados, Alloy4PA ejecutó las fases de abstracción en tiempos mucho más reducidos.

Los valores registrados fueron los siguientes:

- **EPA**: 6.99 segundos
- **BEPA**: 387.11 segundos
- **SBEPA**: 0.09 segundos

En esta configuración, la mayor parte del tiempo de cómputo se concentró nuevamente en la fase `BEPA`, responsable de la verificación de transiciones *must*. No obstante, los tiempos totales resultaron considerablemente menores respecto del caso con `EPA = True`, donde la generación automática de todas las combinaciones de predicados multiplica el número de consultas ejecutadas por Alloy.

En términos cualitativos, el modo manual reduce el crecimiento exponencial del análisis y ofrece tiempos de respuesta mucho más estables, lo que lo convierte en una alternativa adecuada para etapas exploratorias o de depuración del modelo.

5.4.5.7. Conclusión

Esta variante ilustra una segunda forma de evidenciar el bug: no como un estado estancado explícito, sino como una degradación de las garantías sobre una función de progreso. La posibilidad de alternar entre una partición que separa y otra que mezcla los estados concretos brinda flexibilidad para resaltar distintos síntomas de una misma falla lógica.

Además, se observa una diferencia significativa en los tiempos de ejecución entre ambas configuraciones. En la versión con `EPA = True`, la generación automática de particiones (2^N combinaciones) incrementa de manera exponencial el costo de análisis, especialmente en la etapa `BEPA`, donde deben evaluarse todas las transiciones *must*. En cambio, en esta variante con particionado manual (`EPA = False`), los tiempos registrados fueron de mucho menores, reflejando una reducción considerable frente al caso automático.

Este resultado confirma que el enfoque manual permite mantener una cobertura analítica suficiente con un costo computacional mucho menor, a costa de una menor granularidad en la abstracción. Así, EPA =False resulta ventajoso para iteraciones rápidas y exploración de hipótesis sobre la lógica del contrato, mientras que `EPA = True` sigue siendo preferible cuando se busca una representación exhaustiva del espacio de estados.

6. DISCUSIÓN

En esta sección se discutirá la reproducción y ampliación de los experimentos de *Modal Abstractions for Smart Contract Validation* [2], utilizando la herramienta *Alloy4PA* [3] en múltiples entornos. Se discuten la reproducibilidad, utilidad práctica, rendimiento, usabilidad y el análisis de las preguntas de investigación (RQ).

6.1. Resultados de los experimentos

Los experimentos originales del paper fueron completados exitosamente, lo que confirma la reproducibilidad del trabajo original. Se logró replicar los resultados tanto en Windows como en MacOS, y en cada uno se probaron las distintas modalidades de ejecución de *Alloy4PA*. Además, se propusieron y ejecutaron nuevos ejemplos, basados en variaciones de los contratos presentados en el paper, los cuáles incluyeron *SimpleMarket*, *RockPaperScissors* y *AssetTransfer* con bugs intencionales. En estos últimos, la eliminación o degradación de garantías se reflejó en el grafo modal (p. ej., desaparición de transiciones o paso de *must* a *may*), siendo estos resultados finales acorde a lo previsto según la teoría.

La comparación entre *EPA=True* y *EPA=False* mostró un compromiso claro entre exhaustividad y costo computacional: la mayor parte del tiempo se concentra en *BEPA* (verificación de transiciones *must*); con *EPA=False* los tiempos totales se reducen sensiblemente, útil para iteraciones exploratorias, manteniendo una cobertura analítica suficiente con menor granularidad.

6.2. Errores encontrados

Durante la reproducción y ejecución se observaron inconvenientes al ejecutar la herramienta:

- **Gestión de rutas y nombres.** En Windows, rutas con espacios o separadores mixtos provocaron errores. También hubo inconsistencias de nombres esperados (p. ej., sufijo `Config` y extensión `.ini`), resueltas ajustando manualmente.
- **Reejecución tras fallas.** La herramienta tiene una respuesta poco adecuada ante errores. Por ejemplo, si la ejecución falla, pero alcanza a crear el directorio de output, no volverá a intentar correr ese contrato, ya que asume que se completó exitosamente. Para solucionar esto, el usuario tiene dos alternativas prácticas: (i) eliminar manualmente la carpeta antes de relanzar; (ii) ajustar en la configuración global el parámetro que permite *sobrescribir* resultados previos (recordando restaurar el valor luego para evitar efectos no deseados sobre otras ejecuciones).
- **Organización del output.** Los archivos de output también tienen nombres poco consistentes en formato. Por ejemplo, el diagrama de transición de `AssetTransfer` se guarda como `AssetTransferforked_arrows.dot`, que mezcla convenciones de nombre, y no separa claramente el nombre del contrato del tipo de archivo.

6.3. Limitaciones de usabilidad

Si bien se logró validar el funcionamiento de la herramienta, se identificaron algunas limitaciones en cuanto a su usabilidad que impactan la adopción:

- **Costo computacional.** En primer lugar, la herramienta no está optimizada para lograr una buena performance, tal como se menciona en el paper. Algunos experimentos tardaron un tiempo considerable en completarse, lo que dificulta su uso en la práctica. Por ejemplo, el contrato `Rock-Paper-Scissors` tardó alrededor de media hora en completar cada ejecución. (Algunos resultados exactos: 1834s, 1847s.) Dado que este contrato debió ejecutarse varias veces para probar distintas variantes, el tiempo total necesario fue significativo. En este punto es clave mencionar que *BEPA* concentra el tiempo (especialmente con *EPA=True*); *EPA=False* reduce drásticamente tiempos a costa de granularidad, lo que lo vuelve preferible en etapas exploratorias.
- **Interfaz.** Su uso está limitado a línea de comandos, y no tiene ningún tipo de interfaz gráfica. Esto puede dificultar su manejo para usuarios más casuales, y requiere un trabajo más manual para su ejecución.
- **Curva de aprendizaje y documentación.** *Alloy4PA* carece de una guía integral de instalación/uso. Conceptos internos (fases *EPA*, *BEPA*, *SBEPA*) y su relación con los resultados no están explicados en un único recurso, lo que obliga a invertir tiempo en comprender tanto el marco teórico como la configuración/ejecución antes de focalizarse plenamente en el análisis del contrato.
- **Configuración.** Los archivos de input tienen nombres incómodos, por ejemplo, la convención requiere que el archivo de configuración tenga el mismo nombre que el archivo `.als`, con el sufijo `Config`. Siendo que la propia extensión `.ini` ya indica que es un archivo de configuración, por lo que podría omitirse el sufijo. Esto puede ser molesto si se necesita variar el formato de los nombres, por ejemplo separando las palabras con guiones bajos. En este caso se puede destacar que la herramienta informa de manera clara cuál es el archivo esperado cuando no lo encuentra.
- **Organización de archivos de salida.** La organización de carpetas es muy poco intuitiva. El output final de la herramienta se genera en la misma carpeta con los archivos intermedios generados durante la ejecución, lo cual impide encontrar rápidamente los archivos relevantes para el usuario. Así como es poco intuitivo donde se va a generar la propia carpeta de salida de la ejeción (un directorio superior al path), resultando confuso.

6.4. Análisis de las preguntas de investigación

RQ1 – ¿Qué tan frecuentes son las transiciones *must*? En los contratos reproducidos y en los nuevos, las *must* tienden a ser predominantes incluso excluyendo *constructor* y τ, lo que concuerda con el enfoque y con lo observado localmente en *AssetTransfer* y casos simples; las *may* se concentran donde hay condiciones alternativas o incompletas y orientan el diagnóstico.

RQ2 – ¿Las *constraint transitions* ayudan a validar contratos cuando se dispone de roles de usuario permitidos? Si bien en esta tesis no se modelaron roles diferenciados como en el trabajo original, las *constraint transitions* se aplicaron a condicio-

nes internas del contrato (tasador no cargado, comprador no reseteado, pozo vacío). Estas condiciones funcionan como restricciones análogas a los roles en términos de limitar cuándo una operación puede ejecutarse. En los experimentos, dichas variaciones paramétricas modificaron la clasificación modal (*must*/*may*), evidenciando que las *constraint transitions* también aportan información diagnóstica aun en ausencia de roles explícitos.

RQ3 – ¿Los auditores aprovechan la distinción *may*/*must* para exponer problemas que no revela una abstracción solo-*may*? Aunque este trabajo no incluyó auditores profesionales, la distinción *may*/*must* permitió identificar situaciones que serían relevantes en un proceso de auditoría. En las variantes con bug de *AssetTransfer*, transiciones esperadas como *reject* desaparecen o se degradan de *must* a *may*, señalando una pérdida de garantía. En *RockPaperScissors*, modificaciones en precondiciones e invariantes asociadas al pozo generaron transiciones *may* dependientes del balance. Estos patrones son precisamente el tipo de señales que un auditor podría utilizar para detectar comportamientos anómalos que una abstracción solo-*may* no captura.

6.5. Síntesis general

En conjunto, *Alloy4PA* [3] permitió reproducir y extender el enfoque modal con resultados consistentes. *EPA=True* ofrece exhaustividad a mayor costo; *EPA=False*, iteración ágil con menor granularidad. Los errores y las limitaciones detectadas orientan el camino hacia mejoras prácticas y motivan el material de guía propuesto en trabajo futuro. Por ejemplo si se busca que la herramienta sea adoptada más ampliamente, es importante mejorar su rendimiento, como dicen los autores del paper, o hacer mayor énfasis en las diferentes formas de ejecución de la herramienta según objetivos (por ejemplo, con o sin EPA).

7. CONCLUSIONES

Esta tesis reprodujo y extendió el enfoque de *Modal Abstractions for Smart Contract Validation* [2] sobre contratos en Solidity, utilizando *Alloy4PA* [3] en distintos entornos. La reproducción validó la estabilidad del método; las extensiones (*SimpleMarket*, *RockPaperScissors*, *AssetTransfer* con bugs) mostraron cómo la abstracción modal expone inconsistencias de lógica y pérdida de garantías de ejecución.

7.1. Síntesis del trabajo realizado

El trabajo comprendió: (i) estudio del marco de abstracciones modales y su relación con *predicate abstractions*; (ii) reproducción de experimentos del paper base; (iii) nuevos casos y variantes con y sin *EPA*, para evaluar costos y señalización de defectos. Se verificó que la herramienta produce resultados coherentes y que los grafos modales capturan degradaciones de garantía cuando se introducen condiciones problemáticas.

7.2. Conclusiones respecto de las RQ

RQ1. Las transiciones *must* resultaron ser mayoritarias en todos los casos reproducidos y extendidos, coincidiendo con lo esperado por el enfoque modal.

RQ2. Aunque no se utilizaron roles de usuario explícitos, las *constraint transitions* asociadas a condiciones internas demostraron ser útiles para analizar variaciones semánticas equivalentes al uso de roles permitidos.

RQ3. La distinción *may*/*must* permitió identificar problemas de comportamiento que no serían visibles mediante abstracciones solo-*may*, alineándose con el tipo de señalamiento que los auditores podrían explotar.

7.3. Observaciones sobre desempeño y usabilidad

La comparación *EPA=True* vs *EPA=False* confirmó el compromiso entre exhaustividad y costo: *BEPA* domina el tiempo, y el particionado manual acelera iteraciones con menor granularidad. En usabilidad, se identificaron oportunidades claras: documentación integrada, separación de outputs intermedios/finales, mensajes de error más informativos y una superficie de uso menos dependiente de consola.

7.4. Aportes y proyección futura

- **Generación automática de modelos.** Desarrollar herramientas que automaticen desde Solidity la generación de modelos Alloy, incluyendo extracción de predicados, invariantes y relaciones relevantes del código fuente.
- **Interfaz de usuario.** Incorporar una interfaz gráfica simple para configurar/ejecutar *Alloy4PA* [3] sin editar archivos de texto.
- **Guía y material introductorio.** Elaborar documentación que explique el funcionamiento general de *Alloy4PA*, sus fases internas (*EPA*, *BEPA*, *SBEPA*) y pautas

de interpretación de resultados, reduciendo la curva de aprendizaje observada en esta tesis.

- **Organización y trazabilidad de resultados.** Unificar nomenclatura y separar outputs intermedios/finales para facilitar interpretación y comparación entre ejecuciones.

En suma, el enfoque modal es reproducible, ampliable y útil para destacar comportamientos no garantizados en contratos inteligentes. Las mejoras propuestas priorizan accesibilidad y productividad, acercando la técnica tanto a perfiles técnicos como a analistas sin experiencia previa en verificación formal.

Bibliografía

[1] J. Godoy, J. P. Galeotti, D. Garbervetsky, and S. Uchitel. Predicate abstractions for smart contract validation. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*, pages 289–299, October 2022.

[2] J. Godoy, M. Capretto, M. Ceresa, J. P. Galeotti, D. Garbervetsky, C. Sánchez, and S. Uchitel. Modal abstractions for smart contract validation. *To appear*, 2025.

[3] J. Godoy. Alloy4PA: Alloy for predicate abstractions of smart contracts. GitHub repository, 2025. `https://github.com/j-godoy/Alloy4PA`.

[4] Matías Nicolás Incem and Alejandra Alicia Rodríguez. tesis-incem-rodriguez-2025: Modelos y artefactos experimentales de la tesis "validación experimental de abstracciones modales para contratos inteligentes". `https://github.com/alu-rodriguez/tesis-incem-rodriguez-2025`, 2025. Repositorio GitHub.

[5] D. Jackson. Alloy: A language and tool for exploring software designs. *Communications of the ACM*, 62(9):66–76, 2019. doi: 10.1145/3338843.

[6] A. Biere, M. Heule, and H. van Maaren, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, 2009. ISBN 978-1-58603-929-5.

[7] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.

[8] R. M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, 1976.

[9] K. G. Larsen and B. Thomsen. A modal process logic. In *LICS '88*, pages 203–210. IEEE, 1988.

[10] M. Huth, R. Jagadeesan, and D. Schmidt. Modal transition systems: A foundation for three-valued program analysis. In David Sands, editor, *Programming Languages and Systems*, pages 155–169. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[11] Java SE Development Kit 18 - Downloads. `https://www.oracle.com/java/technologies/javase/jdk18-archive-downloads.html`, 2025.

[12] Graphviz Online. `https://dreampuf.github.io/GraphvizOnline`, 2025.

[13] Docker - Downloads. `https://docs.docker.com/get-started/get-docker/`, 2025.

[14] Alloy4PA Docker image on Google Drive (amd64). `https://drive.google.com/file/d/1pangZGQJCH1aD1z7V6dLXoUQhuy4jsmj/view?usp=sharing`, 2025.

[15] Homebrew. `https://brew.sh`, 2025.

[16] How to change JDK version on macOS - Stack Overflow. https://stackoverflow.com/a/68105964, 2025.

[17] How to set JAVA_HOME variable on macOS - Stack Overflow. https://stackoverflow.com/q/22842743, 2025.

[18] Apple silicon platform architecture. https://developer.apple.com/documentation/apple-silicon, 2025.

[19] Alloy4PA Docker image on Google Drive (arm64). https://drive.google.com/file/d/1G0Cy-NVLz4zY_ogy_kA0glbbZds3_o4M/view?usp=sharing, 2025.

[20] Python - Downloads. https://www.python.org/downloads/, 2025.