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

Una caracterización operacional para modelos de consistencia transaccional

Tesis de Licenciatura en Ciencias de la Computación

Emanuel Lamela

Director: Christian Roldán

Buenos Aires, 2020

UNA CARACTERIZACIÓN OPERACIONAL PARA MODELOS DE CONSISTENCIA TRANSACCIONAL

Hoy en día se construyen aplicaciones sobre bases de datos replicadas que garantizan escalabilidad y alta disponibilidad a expensas de sacrificar la consistencia, es decir, los usuarios pueden observar temporalmente diferencias sobre el estado del sistema. Un modelo de consistencia caracteriza este tipo de inconsistencias o anomalías. Por lo tanto, razonar sobre la semántica de programas está directamente ligado a los modelos de consistencia que este tipo de bases de datos ofrecen.

La literatura define a los modelos de consistencia en términos de axiomas que restringen las posibles ejecuciones que suceden en una base de datos replicada. En particular, en esta tesis nos concentramos en modelos de consistencia para *transacciones*. Una traza de ejecución es reconstruida a través de un grafo de eventos conocido como ejecución abstracta. La misma se define sobre un conjunto de transacciones y dos relaciones: (i) visibilidad, que define cuando una transacción es conocida por otra, y (ii) arbitración, que especifica un orden relativo entre las transacciones ejecutadas por el sistema. Estos mecanismos, si bien son declarativos, no inducen una operatoria clara que pueda ser sencillamente traducida a una implementación.

En esta tesis proponemos un modelo operacional basado en un sistema de transición etiquetado que captura de forma general los modelos de consistencia para transacciones. Para esto definimos el estado del sistema como un orden parcial sobre una secuencia de transacciones, una regla de equivalencia estructural, y un conjunto de reglas de derivación descriptas en términos de dos predicados paramétricos: *commit consistency* y *arbitration consitency*. En este trabajo, presentamos instanciaciones particulares de cada predicado que servirán como definiciones alternativas de los modelos de consistencia como *Snapshot Isolation* y *Causal Consistency*.

Probaremos la equivalencia entre el enfoque clásico, definido axiomáticamente, y nuestro modelo operacional. Concretamente, demostramos la correspondencia presentando pruebas de *soundness* y *completeness* para cada modelo de consistencia transaccional.

Palabras claves: Replicación, Modelos de consistencia transaccional, Semántica operacional, Soundness, Completeness

Índice general

1. INTRODUCCIÓN

1.1. Motivación

Replicación en bases de datos distribuidas refiere al hecho de almacenar múltiples copias de un mismo dato en diferentes nodos (o servidores) de una red, con el fin de mejorar la latencia y la tolerancia a fallas. Latencia es una característica vinculada a la experiencia de usuario, y tiene que ver con la velocidad de respuesta que tiene ejecutar una operación. Tolerancia a fallas significa que el sistema puede seguir operando aunque haya caída de nodos o haya perdida de mensajes. En contraposición, el precio a pagar es la consistencia (fuerte) que ofrece el sistema, es decir, la seguridad que cada usuario del sistema observe un mismo estado. Esta paradoja, enunciada formalmente como el Teorema CAP [9] (por sus siglas en inglés: *Consistency*, *Availability* y *Partition Tolerance*), ocurre ante la imposibilidad de garantizar simultáneamente consistencia fuerte, alta disponibilidad y tolerancia a fallas. Dependiendo del problema y los atributos de calidad a alcanzar, es la elección de cuáles de estas propiedades satisfacer. Lógicamente, la tolerancia a fallas es una elección inevitable, por ende la decisión recae en consistencia versus disponibilidad. Hoy en día, las bases de datos distribuídas más populares, como Cassandra [10], Dynamo [8] o MongoDB [12], aseguran disponibilidad pero ofrecen una noción débil de consistencia, conocida como *Consistencia Eventual* [2]. La consistencia eventual garantiza que cualquier escritura tarde o temprano (*eventually*) será entregada a cada réplica del sistema logrando que el sistema, es decir el conjunto de réplicas, converjan a un mismo estado.

La consistencia se puede pensar tanto a nivel *operación individual* (escritura o lectura) o a nivel *transacción* (secuenciado de operaciones individuales). La agrupación transaccional responde a una cuestión semántica y operativa. Por ejemplo, considere el escenario donde un usuario, cuya cuenta bancaria es identificada por `o`, desea realiza una transferencia bancaria por medio de la operación `transferir(Double m, Cuenta d)`. La operación transferir es responsable de debitar un monto `m` a la cuenta origen `o`, y depositar el monto `m` a la cuenta destino `d`. La expectativa luego de ejecutar la operación `transferir` será: (i) o bien se ejecute el débito y depósito correspondiente o (ii) no se produjo ningún efecto en el estado de la cuenta origen y destino. En consecuencia, las transacciones son introducidas al sistema de manera atómica: el sistema aplica el 0 % o el 100 % de la misma. Este trabajo pone el foco en sistemas transaccionales distribuidos.

La definición de consistencia eventual habilita un amplio espectro de niveles de consistencia que una base de datos puede proveer [15]. Los *modelos de consistencia* formalizan ciertas garantías las cuales limitan los tipos de anomalías o inconsistencia que un sistema admite. Entre las garantías de consistencia para sistemas no-transaccionales nos encontramos por ejemplo con *read-your-writes* [14], que garantiza que cuando un cliente lea una base de datos siempre verá los efectos de sus escrituras precedentes. No obstante, este modelo asegura que las operaciones de lectura que ejecuta un cliente se corresponden con escrituras previamente realizadas en la base de datos. Para sistemas transaccionales existen *causal consistency* [11, 5] y *prefix consistency* [6], entre otras. En este trabajo estudiamos *modelos de consistencia transaccional*, es decir, garantías de consistencia que predican sobre transacciones. La literatura define los modelos de consistencia en base a axiomas de consistencia [4, 15]: predicados lógicos que caracterizan las ejecuciones permitidas por el sistema, evitando de

esta manera, anomalías tales como *Lost update* o *Write Skew* [7].

1.2. Objetivo

El trabajo de tesis tiene como objetivo crear una caracterización operacional que sea equivalente a las definiciones axiomáticas enunciadas en [7]. Concretamente, vamos a mostrar que aquellas ejecuciones que respetan los axiomas de consistencia transaccionales: TransVis, NoConflicts, Prefix y TotalVis (presentados formalmente en [7]), pueden combinarse y describir modelos de consistencia en un único modelo operacional, instanciado adecuadamente de acuerdo a la garantía de consistencia a alcanzar.

Las ejecuciones se modelan a través de una estructura de eventos conocida como *ejecución abstracta* [5] que indica como se vinculan los eventos en término de dos relaciones concretas: *visibilidad*, indicando que una transacción conoce los efectos de la otra, y *arbitración*, indicando un orden relativo entre las transacciones ejecutadas en el sistema. Ambas relacionan *atómicamente* las transacciones como células indivisibles, sin permitir conocimiento y/u ordenamiento de alguna subsecuencia de operaciones dentro de una transacción. Con esta estructura como sustento, enunciaremos formalmente los axiomas de consistencia ya mencionados.

Luego, presentamos un modelo operacional en términos de un sistema de transiciones etiquetados para describir el comportamiento del sistema. El estado, se describe como un orden parcial, que permite describir el orden en el que las transacciones fueron impactadas en cada réplica. Cada transacción es decorada con información sobre la primera réplica donde fue ejecutada, y el nombre de las réplicas que la conocen. El comportamiento está dado por reglas de transición paramétricas en base a los predicados *commit consistency* y *arbitration consistency*, instanciables de acuerdo al modelo de consistencia que se intenta garantizar. Remarcamos que este *framework* operacional es lo suficientemente general para no dar detalles concretos de implementación como los mecanismos de propagación entre réplicas.

Finalmente, probamos que el modelo operacional instanciado según cada modelo de consistencia transaccional `C`, es *sound* y *complete*, es decir, es equivalente a ejecuciones abstractas que satisfacen el modelo `C`. De esta manera, proporcionamos un marco de trabajo equivalente, donde los usuarios pueden moverse entre ambos mundos sin perder expresividad.

1.3. Organización

En el siguiente capítulo se resumen nociones básicas y conceptos que utilizaremos a lo largo de este trabajo. Principalmente, nos detenemos en ejecución abstracta y los axiomas de consistencia a nivel transacción. En el capítulo 3 desarrollamos un modelo operacional para caracterizar bases de datos con consistencia eventual, formulando las definiciones alternativas para los modelos de consistencia transaccional de interés. Luego, en capítulo 4, mostramos la correspondencia de nuestro framework con ejecuciones abstracta que ofrecen distintos niveles de consistencia. Finalmente en el capítulo 5 se presentan consideraciones finales y, se discuten aspectos futuros a explorar y trabajos relacionados.

2. PRELIMINARES

2.1. Conjuntos

Un *conjunto* A es una colección de objetos distintos tales que, dado un objeto cualquiera x, se puede determinar si x pertenece a A o no. Este mecanismo está representado por el operador $\in$ cuya aplicación se denota $x \in A$ y su negación $x \notin A$. El literal de los conjuntos es $A = \{x_1, \ldots, x_n\}$ y expresa $(x_1 \in A \land \ldots \land x_n \in A)$. Utilizamos la notación $\{x \mid \textit{condición}\}$ para indicar que el conjunto se construye tomando todos los x que satisfacen *condición*.

▶ **Ejemplo 2.1.1.** *Sea* $\mathbb{N}$ *el conjunto de números naturales. Escribimos el conjunto de los naturales pares como* $\mathbb{P} = \{x \mid x \in \mathbb{N} \land x \ mod\ 2 = 0\}$.

Cuando ocurre que $x \in A \implies x \in B$ decimos que A es *subconjunto* de B (o que B es *superconjunto* de A). El operador $A \subseteq B$ (o $A \supseteq B$ respectivamente) encapsula dicha propiedad. Un corolario que se desprende es $A \subseteq B \land B \subseteq A \implies A = B$. Un conjunto particular es el *vacío*, cuyo literal es $\emptyset$, que verifica $\forall\, x.\, x \notin \emptyset \land \forall A.\, \emptyset \subseteq A$. El *tamaño* o la *cardinalidad* de un conjunto se define como la cantidad de elementos que posee y se escribe como $|A|$, por lo tanto $|\emptyset| = 0$ y $|\{1, 2, 3\}| = 3$. Algunas operaciones comunes sobre conjuntos que vamos a utilizar son:

- *Unión de Conjuntos*: $A \cup B = \{x \mid x \in A \lor x \in B\}$
- *Intersección de conjuntos*: $A \cap B = \{x \mid x \in A \land x \in B\}$
- *Diferencia de conjuntos*: $A \setminus B = \{x \mid x \in A \land x \notin B\}$

Decimos que dos (o más) conjuntos son *disjuntos* si $A_1 \cap \ldots \cap A_n = \emptyset$. Una construcción de conjuntos que será esencial es el *producto cartesiano*. Se define como $A_1 \times \ldots \times A_n = \{(x_1, \ldots, x_n) \mid x_1 \in A_1 \land \ldots \land x_n \in A_n\}$, siendo $(x_1, \ldots, x_n)$ una *tupla n-aria*. El producto cartesiano de un conjunto A cero o más veces consigo mismo es formalizado por la notación: $A^* = \{(x_1, \ldots, x_n) \mid x_i \in A \land i \in \{1, \ldots, n\} \land n \geq 0\}$.

2.2. Relaciones y órdenes

Habiendo definido conjuntos, podemos definir una *relación* como un subconjunto del producto cartesiano entre dos conjuntos. En particular, nos enfocaremos en *relaciones binarias*. R es una relación binaria de A con B si R es un subconjunto de $A \times B$. Al ser un conjunto, también se puede definir como $\{\ (a, b) \mid a \in A \land b \in B \land \textit{condición}\ \}$. Por esta razón, las relaciones binarias suelen representar una correspondencia entre 2 conjuntos sujeta a *condición*. Notamos $a \xrightarrow{R} b$ cuando ocurre $(a, b) \in R$.

▶ **Ejemplo 2.2.1.** *Sea* $A = \{1, 2, 3\}$ *y* $B = \mathbb{N}$*, entonces la siguiente relación*

$$\{(a, b) \mid a \in A \land b \in B \land b = 2 * a\ \} = \{(1, 2), (2, 4), (4, 6)\}$$

es la que asocia los elementos de A *a sus dobles en* B.

Las relaciones también pueden ser *compuestas*. Esta operación se caracteriza como $R_1; R_2 = \{(a, c) \mid \exists\, b.\, (a, b) \in R_1 \wedge (b, c) \in R_2\}$. Otra operación que nos será de utilidad es la *restricción de relaciones*. Decimos que restringimos una relación binaria R a un conjunto E cuando $R|_E = \{(a, b) \mid (a, b) \in R \wedge a \in E \wedge b \in E\}$.

Las relaciones suelen categorizarse en base a la satisfacción de propiedades de interés. A continuación, enunciamos algunas de las que utilizaremos posteriormente. Siendo R una relación binaria:

- Antisimetria: $(a, b) \in R \wedge (b, a) \in R \implies a = b$
- Transitividad: $(a, b) \in R \wedge (b, c) \in R \implies (a, c) \in R$
- Aciclicidad: $\nexists\, a_1, \ldots, a_k.\, (a_1, a_2) \in R \wedge \cdots \wedge (a_k, a_1) \in R$
- Downward Totality: $(a, c) \in R \wedge (b, c) \in R \implies (a, b) \in R \vee (b, a) \in R$
- Totalidad: $(a, b) \in R \vee (b, a) \in R$

La conjunción de algunas de estas (y otras) propiedades se cataloga como *órden*. Los órdenes que emplearemos son: strict prefix order y total order.

Strict prefix order es un orden sobre una relación $R \subseteq A \times A$ tal que R es antisimétrica, transitiva y downward total.

► **Ejemplo 2.2.2.** *Considere el conjunto $A = \{1, 2, 3, 4, 5\}$, la relación*

$$R_1 = \{(1, 2), (2, 3), (1, 3), (3, 4), (1, 4), (2, 4), (2, 5), (1, 5)\}$$

es un strict prefix order. Fig. 2.1a es una representación gráfica de la relación R_1.

Una característica de utilidad es que induce ramificaciones. Partiendo de Ej. 2.2.2, tenemos que $(2, 4) \in R_1 \wedge (3, 4) \in R_1$ lo cuál obliga, por downward totality, a que $(2, 3) \in R_1 \vee (3, 2) \in R_1$. En este caso particularmente $(2, 3) \in R_1$. Por lo tanto, por transitividad, se debe considerar el *prefijo* (o *historia*) del elemento 2, lo que da origen al nombre del orden. Estas particularidades hacen que este tipo de orden sea útil para modelar ejecuciones. En el marco de una ejecución, los prefijos pueden considerarse estado compartido y las bifurcaciones como eventos que ocurren concurrentemente, el cuál es un caso común en bases de datos.

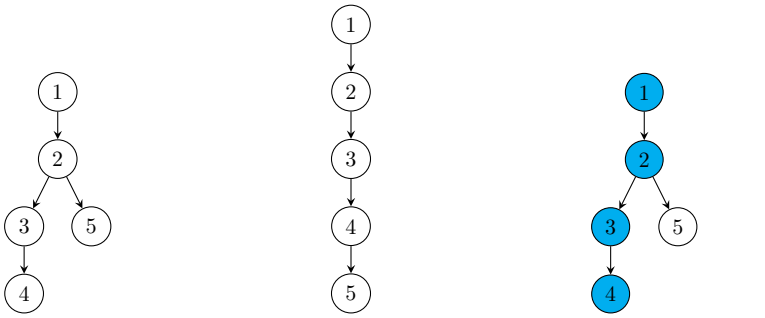
Total order es un orden sobre una relación $R \subseteq A \times A$ tal que R es antisimétrica, transitiva y total.

► **Ejemplo 2.2.3.** *Considerando el mismo conjunto A de Ej. 2.2.2, la relación*

$$R_2 = \{(1, 2), (1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (2, 5), (3, 4), (3, 5), (4, 5)\}$$

es un total order. La Fig. 2.1b representa gráficamente la relación R_2.

Este tipo de relaciones, al demandar totalidad, caracterizan relaciones lineales. Este fenómeno se aprecia en la segunda cadena de Fig. 2.1b. Es por ello que suelen utilizarse para modelar secuencias de eventos totalmente ordenados, como puede ser el caso de una secuencia de transacciones ejecutadas en una misma réplica. El *timestamp* puede determinar un orden total sobre las mismas.



(a) Relación R1 *(b)* Relación R2 *(c)* Conjunto $E \subseteq A$

Fig. 2.1: Órdenes y conjuntos, sin ejes transitivos

Por último, una propiedad que predica sobre un conjunto y un órden: la noción de conjunto downward closed. Decimos que un conjunto $E \subseteq A$ es Downward Closed con respecto a un orden $R \subseteq A \times A$ si verifica:

$$e \in E \land a \in A \land (a, e) \in R \implies a \in E$$

El subconjunto E extrae elementos de A tales que están relacionados en R *a izquierda* con $e \in E$. Reutilizando A y R_1 del Ej. 2.2.2, en Fig. 2.1c se puede observar el conjunto downward closed $E = \{1, 2, 3, 4\}$. La presencia del elemento 4 obliga incluir a $\{1, 2, 3\}$ ya que están relacionados a izquierda. De quitarlo, $E' = \{1, 2, 3\}$ continuaría siendo downward closed. Por el contrario, $E'' = \{2, 3, 4\}$ pierde la propiedad debido a que 1 está relacionado a izquierda con el resto. En definitiva, la presencia de un elemento en un conjunto downward closed garantiza la inclusión de lo ordenado a izquierda transitivamente.

2.3. Transacciones

Intuitivamente, una transacción se puede ver como un programa en ejecución que incluye uno o más accesos a una base de datos. Por lo tanto, una transacción es una secuencia de acciones (u operaciones) que se impactan de manera atómica: se aplican todas las operaciones en el orden en el que aparecen, y en caso de error se hace un *rollback* de todas las operaciones que fueron realizadas hasta el momento, cancelando entonces las operaciones posteriores. Dicho eso, este trabajo operara bajo ciertas simplificaciones. No consideramos el caso de error, es decir, las transacciones siempre son ejecutadas de forma satisfactoria.

▶ **Ejemplo 2.3.1.** *La siguiente tabla describe el contenido de dos transacciones:* T_1 *y* T_2*. Ambas manipulan objetos (o variables) a través de operaciones de lectura y escritura. Concretamente, la transacción* T_1 *acumula* 50 *sobre el objeto* x*, mientras que* T_2 *acumula* 25*, sobre el mismo objeto.*

T_1	T_2
`x := x + 50`	`x := x + 25`

Entonces, T_1 *y* T_2*, primero obtienen el valor almacenado en el objeto* x *(es decir, a través de una operación de lectura) y luego suman un valor constante, actualizando el valor del*

objeto x con el resultado de la operación de adición. Considere un escenario donde el sistema se compone de dos réplicas: $\mathtt{i}_1$ *y* $\mathtt{i}_2$*, donde el objeto x vale* 100 *en la réplica* $\mathtt{i}_1$ *y* 200 *en la réplica* $\mathtt{i}_2$*. Entonces, si ejecutamos* T_1 *y* T_2 *en* $\mathtt{i}_1$ *y* $\mathtt{i}_2$ *respectivamente, entonces sería equivalente ejecutar las siguientes transacciones:*

T'_1	T'_2
`x := 150`	`x := 225`

Por lo tanto, el contenido de las transacciones T'_1 *y* T'_2 *se obtiene sustituyendo los valores relativos,* $(x + 50)$ *y* $(x + 25)$*, de* T_1 *y* T_2 *con los valores* 100 *y* 200 *respectivamente. Esto es, reemplazamos los valores relativos del objeto x por los valores absolutos, resultado de leer los valores* 100 *y* 200 *de x en las réplicas respectivas.*

Cómo se resuelven estos valores relativos depende del estado de cada réplica que es *visible* a cada transacción al momento de impactarse. El conjunto de las transacciones es representado por $\mathbb{T}$. El modelado formal será presentado en el capítulo 3.

2.4. Ejecución abstracta

Presentamos una construcción que nos permite describir de manera declarativa la ejecución de un sistema. El recurso es conocido en la literatura como *ejecución abstracta* [5]. Es una estructura que caracteriza un conjunto de *eventos* $\mathcal{E}$, los cuales serán etiquetados/asignados a una transacción, con respecto a las relaciones de *arbitración* $\mathtt{AR}$ y *visibilidad* $\mathtt{VIS}$.

Visibilidad indica cuando un evento conoce los efectos de otro. $\mathtt{e}_1 \xrightarrow{\mathtt{VIS}} \mathtt{e}_2$ nos dice que la operación representada por $\mathtt{e}_2$ conoce los efectos de la representada por $\mathtt{e}_1$. Si ocurre que $\neg(\mathtt{e} \xrightarrow{\mathtt{VIS}} \mathtt{e}' \vee \mathtt{e}' \xrightarrow{\mathtt{VIS}} \mathtt{e})$, decimos que los dos eventos son *concurrentes*.

Arbitración describe un orden temporal entre eventos. $\mathtt{e}_1 \xrightarrow{\mathtt{AR}} \mathtt{e}_2$ indica que el evento $\mathtt{e}_2$ ocurre posteriormente a $\mathtt{e}_1$. Cuando ocurre que $(\mathtt{e} \xrightarrow{\mathtt{AR}} \mathtt{e}' \vee \mathtt{e}' \xrightarrow{\mathtt{AR}} \mathtt{e})$, se dice que los eventos se encuentran *arbitrados*.

Definición 1 (**Ejecución abstracta**)**.** *Una ejecución abstracta es una tupla* $\mathcal{A} = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR} \rangle$ *siendo:*

1. $\mathtt{OP} : \mathcal{E} \mapsto \mathbb{T}$ *una función de eventos a transacciones*
2. $\mathtt{VIS} \subseteq \mathcal{E} \times \mathcal{E}$ *una relación acíclica*
3. $\mathtt{AR} \subseteq \mathcal{E} \times \mathcal{E}$ *un strict prefix order tal que* $\mathtt{VIS} \subseteq \mathtt{AR}$

La función $\mathtt{OP}$ asigna a cada evento $\mathtt{e}$ una transacción $\delta \in \mathbb{T}$. La restricción $\mathtt{VIS} \subseteq \mathtt{AR}$ prohíbe que un evento pueda quedar arbitrado antes de otro visible. Notamos $\emptyset$ como la ejecución abstracta vacía, es decir tal que $\mathcal{E} = \emptyset$.

2.5. Modelos de consistencia transaccionales

Los *modelos de consistencia transaccionales* definen qué tipo de interacciones son permitidas por el sistema, acotando los tipos de anomalías permitidos. En rigor, se formalizan como una conjunción de *axiomas de consistencia*: predicados que indican cómo pueden relacionarse

C	Modelo	Axiomas	Causality violation	Lost update	Long fork	Write skew
CC	Causal consistency [11, 5]	TransVis	✗	✓	✓	✓
PSI	Parallel snapshot isolation [1]	TransVis, NoConflict	✗	✗	✓	✓
PC	Prefix consistency [6]	Prefix	✗	✓	✗	✓
SI	Snapshot isolation [3]	Prefix, NoConflict	✗	✗	✗	✓
SER	Serialisability [13]	TotalVis	✗	✗	✗	✗

Tab. 2.1: Definiciones de modelos de consistencia y anomalías que prohíben

las transacciones dentro del sistema, representado por una ejecución abstracta. A continuación presentamos los siguientes axiomas de consistencia que nos servirán para definir modelos de consistencia, los mismos se encuentran en el diagrama Tab. 2.1:

- TransVis (TV): $\texttt{VIS}$ es transitiva
- NoConflict (NC):
  $\forall \mathsf{e}, \mathsf{e}' \in \mathcal{E}.\, \mathsf{e} \neq \mathsf{e}' \wedge \texttt{VAR}(\mathsf{e}) \cap \texttt{VAR}(\mathsf{e}') \neq \emptyset \implies (\mathsf{e}, \mathsf{e}') \in \texttt{VIS} \vee (\mathsf{e}', \mathsf{e}) \in \texttt{VIS}$
- Prefix (PP): $\texttt{AR}; \texttt{VIS} \subseteq \texttt{VIS}$
- TotalVis (TOTV): $\texttt{VIS}$ es total

El primer predicado es TransVis (TV). Como indica el nombre, fuerza el siguiente escenario: $\mathsf{e} \xrightarrow{\texttt{VIS}} \mathsf{e}' \xrightarrow{\texttt{VIS}} \mathsf{e}''$ implica $\mathsf{e} \xrightarrow{\texttt{VIS}} \mathsf{e}''$. Informalmente, los eventos heredan la historia visible de aquello que observan de manera directa. Esto permite mitigar violaciones del estilo *Causality violation*, ilustrada en Ej. 2.5.1.

▶ **Ejemplo 2.5.1.** *Violación de Visibilidad Transitiva - Causality violation*

T_1	T_2	T_3
`x := 25` `y := 100`	`x := x + 50`	`y := y + 100`

$$\begin{bmatrix} \mathcal{A} = \langle \{\mathsf{e}_1, \mathsf{e}_2, \mathsf{e}_3\}, \texttt{OP}, \texttt{VIS}, \texttt{AR} \rangle \\ \texttt{OP} = \{\mathsf{e}_1 \mapsto \mathsf{T}_1, \mathsf{e}_2 \mapsto \mathsf{T}_2, \mathsf{e}_3 \mapsto \mathsf{T}_3\} \\ \texttt{VIS} = \{(\mathsf{e}_1, \mathsf{e}_2), (\mathsf{e}_2, \mathsf{e}_3)\} \\ \texttt{AR} = \texttt{VIS} \end{bmatrix}$$

Supongamos entonces que la transacción T_1 sobrescribe el estado de los objetos x e y con los valores 25 y 100 respectivamente. Como ocurre que $\mathsf{e}_1 \xrightarrow{\texttt{VIS}} \mathsf{e}_2$, entonces T_2 actualiza x con el valor 75. Por otra parte, suponga que T_3 se ejecuta en una réplica distinta, donde x e y valen 0. Además, sabiendo que $\mathsf{e}_2 \xrightarrow{\texttt{VIS}} \mathsf{e}_3$, pero $\neg(\mathsf{e}_1 \xrightarrow{\texttt{VIS}} \mathsf{e}_3)$. Podemos concluir que la réplica actualiza y, colocando 100 en ella en lugar de 200. Esto es un ejemplo de una violación de visibilidad transitiva ya que T_3 no conoce las dependencias causales. Obligar la ocurrencia de $\mathsf{T}_1 \xrightarrow{\texttt{VIS}} \mathsf{T}_3$ resolvería esta anomalía.

► **Ejemplo 2.5.2.** *Violación de No Conflictos - Lost update*

T_1	T_2
`x := x + 50`	`x := x + 25`

$$\left[\begin{array}{r} \mathcal{A} = \langle\{\mathtt{e}_1, \mathtt{e}_2\}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR}\rangle \\ \mathtt{OP} = \{\mathtt{e}_1 \mapsto \mathsf{T}_1, \mathtt{e}_2 \mapsto \mathsf{T}_2\} \\ \mathtt{VIS} = \emptyset \\ \mathtt{AR} = \emptyset \end{array}\right]$$

Dado que $\neg(\mathtt{e}_1 \xrightarrow{\mathtt{VIS}} \mathtt{e}_2 \vee \mathtt{e}_2 \xrightarrow{\mathtt{VIS}} \mathtt{e}_1)$, T_1 *y* T_2 *ingresan al sistema concurrentemente buscando escribir el mismo objeto* x*. Suponiendo un escenario inicial* $x = 0$*, cada escritura ocurre de manera ajena a la otra, obteniendo como valor final* 25 *o* 50*, es decir, ignorando la otra escritura. Esta se denomina* Lost update *y es mitigada por* NoConflict *(*`NC`*), que demanda que operaciones que trabajen sobre un mismo subconjunto de variables deban conocerse entre sí.*

En este trabajo introducimos el operador `VAR(e)` para proyectar el conjunto de variables sobre las cuáles escribe una transacción asociada al evento `e`. La formalización de este operador será introducida posterior al modelado de las transacciones.

Considere ahora el axioma Prefix (`PP`), proponiendo garantías sobre los prefijos de las transacciones: $\mathtt{e} \xrightarrow{\mathtt{AR}} \mathtt{e}' \xrightarrow{\mathtt{VIS}} \mathtt{e}''$ implica $\mathtt{e} \xrightarrow{\mathtt{VIS}} \mathtt{e}''$. Prefix entonces implica TransVis ya que $\mathtt{VIS} \subseteq \mathtt{AR}$, por lo tanto se verifica $\mathtt{PP} \implies \mathtt{TV}$. El axioma garantiza heredar la historia de arbitración. El objetivo de este modelo es evitar la anomalía *Long fork*, que provienen de potencialmente observar transacciones concurrentes en distintos órdenes, permitidas por los predicados previos.

► **Ejemplo 2.5.3.** *Violación de Prefijos - Long fork*

T_1	T_2	T_3	T_4
`x := 10`	`y := 10`	`z1 := y + 30`	`z2 := x + 40`

$$\left[\begin{array}{r} \mathcal{A} = \langle\{\mathtt{e}_1, \mathtt{e}_2, \mathtt{e}_3, \mathtt{e}_4\}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR}\rangle \\ \mathtt{OP} = \{\mathtt{e}_1 \mapsto \mathsf{T}_1, \mathtt{e}_2 \mapsto \mathsf{T}_2, \\ \qquad \mathtt{e}_3 \mapsto \mathsf{T}_3, \mathtt{e}_4 \mapsto \mathsf{T}_4\} \\ \mathtt{VIS} = \{(\mathtt{e}_1, \mathtt{e}_3), (\mathtt{e}_2, \mathtt{e}_4)\} \\ \mathtt{AR} = \mathtt{VIS} \end{array}\right]$$

Como ocurre $\mathtt{e}_1 \xrightarrow{\mathtt{VIS}} \mathtt{e}_3$ *y* $\mathtt{e}_2 \xrightarrow{\mathtt{VIS}} \mathtt{e}_4$*, desde las perspectivas de* T_3 *y* T_4*, las transacciones* T_2 *y* T_1 *no ocurrieron respectivamente. Sin embargo, si luego el sistema decide arbitrar* T_1 *antes que* T_2*, es decir,* $\mathsf{T}_1 \xrightarrow{\mathtt{AR}} \mathsf{T}_2$*, esto significa una reescritura de la historia de* T_4*. Por ejemplo, si los valores iniciales para* x *e* y *es* 0*, al ejecutar* T_4 *luego de* T_2 *el valor para* $z2$ *será* 40*, mientras que luego de arbitrar* T_1 *antes que* T_2 *el valor esperable para esa traza sería que* $z2$ *valga* 50*. Entonces, el predicado* Prefix *responde a la intuición de que si un evento ocurre antes que otro conocido, debería observar los efectos en el mismo orden.*

Por último, el axioma más restrictivo, TotalVis. Concretamente, obliga a ordenar totalmente por visibilidad, lo que también fuerza arbitración por $\mathtt{VIS} \subseteq \mathtt{AR}$, efectivamente transformando la ejecución abstracta en una historia serial. Prevee todo tipo de anomalías, incluyendo bifurcales como *Write skew* que no es alcanzada por los otros modelos. El Ej. 2.5.4 exhibe la situación.

► **Ejemplo 2.5.4.** *Violación de Visibilidad Total - Write skew*

T_1	T_2	T_3

`x := 60`
`y := 60`
	`if (x + y > 100) {`
	`  x := x - 100`
	`}`
		`if (x + y > 100) {`
		`  y := y - 100`
		`}`

$$\begin{bmatrix} \mathcal{A} = \langle \{\mathsf{e}_1, \mathsf{e}_2, \mathsf{e}_3\}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR} \rangle \\ \mathtt{OP} = \{\mathsf{e}_1 \mapsto \mathsf{T}_1, \mathsf{e}_2 \mapsto \mathsf{T}_2, \\ \qquad \mathsf{e}_3 \mapsto \mathsf{T}_3\} \\ \mathtt{VIS} = \{(\mathsf{e}_1, \mathsf{e}_2), (\mathsf{e}_1, \mathsf{e}_3)\} \\ \mathtt{AR} = \mathtt{VIS} \end{bmatrix}$$

Inicialmente, T_1 *impacta* $x = y = 60$*, lo que es contemplado por* T_2 *y* T_3 *debido a que* $\mathsf{e}_1 \xrightarrow{\mathtt{VIS}} \mathsf{e}_2 \wedge \mathsf{e}_1 \xrightarrow{\mathtt{VIS}} \mathsf{e}_3$*. El hecho* $\neg(\mathsf{e}_2 \xrightarrow{\mathtt{VIS}} \mathsf{e}_3) \wedge \neg(\mathsf{e}_3 \xrightarrow{\mathtt{VIS}} \mathsf{e}_2)$ *hace que* T_2 *y* T_3 *sean concurrentes. La evaluación de la condición* $(x + y > 100)$ *resulta* `true` *en ambos casos, lo que desemboca en la ejecución de* $x = -40$ *y* $y = -40$ *respectivamente. Lo que no hubiese ocurrido en un contexto de ejecución serial.*

3. MODELO OPERACIONAL

Introducimos un framework operacional que considera los predicados *commit consistency* y *arbitration consistency*, los cuáles instanciados adecuadamente permiten dar garantías equivalentes a los modelos de consistencia transaccional introducidos en Tab. 2.1. Primero presentaremos el modelo en su versión simplificada que luego decoraremos para mostrar las equivalencias entre este y las ejecuciones abstractas. Una vez presentada la gramática y la semántica, procedemos a mostrar cómo modelar escenarios concretos a partir del modelo desarrollado.

3.1. Sintaxis

Asumimos $\mathcal{O}$ el conjunto de *objetos* $x, y, z, \ldots$, $\mathcal{V}$ el conjunto de *valores de respuesta* $v, v'', v'', \ldots$, siendo $\perp$ el valor indefinido, e $\mathcal{I}$ el conjunto de *réplicas* $\mathtt{i}, \mathtt{j}, \ldots$. Representamos los estados de un sistema replicado como secuencias σ de cero o más transacciones, lo que conlleva a $\sigma \in \mathbb{T}^*$. Las transacciones son caracterizadas como secuencias de *escrituras*, cuyo conjunto es $\mathbb{U}$. Escribimos $\mathtt{u} = (x \hookleftarrow v)$ en lugar de (x, v) para denotar la actualización del objeto x con el valor v. Por ende $\mathbb{U} = \mathcal{O} \times \mathcal{V}$. También $\mathsf{T} = [\mathtt{u}_1 \cdot \ldots \cdot \mathtt{u}_n]^{\mathtt{I}}_{\mathtt{i}}$ para representar $([\mathtt{u}_1 \cdot \ldots \cdot \mathtt{u}_n], \mathtt{i}, \mathtt{I})$ la transacción que aplica la secuencia de escrituras $[\mathtt{u}_1 \cdot \ldots \cdot \mathtt{u}_n]$ realizadas en la réplica $\mathtt{i}$ conocida por conjunto de réplicas $\mathtt{I}$. En consecuencia $\mathbb{T} = \mathbb{U}^* \times \mathcal{I} \times 2^{\mathcal{I}}$. Utilizamos la agrupación $[x_1 \ldots x_n]$ para determinar que el conjunto de elementos $\{x_1, \ldots, x_n\}$ se encuentra ordenado totalmente por orden de aparición en la secuencia.

Tanto las escrituras como las transacciones proveen proyecciones de sus propiedades. Por ejemplo, denotamos $\mathsf{T}.\mathtt{i}$ como la segunda componente de la tupla de la transacción T. De manera particular, la proyección de $\mathsf{T}.\vec{\mathtt{u}}$ representa $[\mathtt{u}_1 \cdot \ldots \cdot \mathtt{u}_n]$, la primer componente de la tupla, y $\mathsf{T}.\mathtt{X}$ es el conjunto de todos los objetos sobre los cuáles T escribe. Adicionalmente, los estados proyectan sus réplicas a través de $\sigma.\mathtt{I} = \mathsf{T}_1.\mathtt{I} \cup \ldots \cup \mathsf{T}_n.\mathtt{I}$. Dependiendo de la ocasión, tomamos la libertad de describir una transacción como $\mathsf{T}^{\mathtt{I}}_{\mathtt{i}}$, con $\mathsf{T} = [\mathtt{u}_1 \cdot \ldots \cdot \mathtt{u}_n]$. El operador $\_[\_ \mapsto \_]$ redefine el valor de una componente de una tupla. A modo de ejemplo, $\mathtt{u}[v \mapsto v']$ es el resultado de tomar la escritura $\mathtt{u}$ y reemplazar v por v'. En cuánto a la construcción de las secuencias, ϵ es la secuencia vacía para los estados y $[]$ la correspondiente para transacciones, y pueden ampliarse a través del operador de concatenación $- \cdot -$. Utilizamos $\mathtt{u} \in \mathsf{T}$ para indicar que $\mathsf{T} = [\ldots \cdot \mathtt{u} \cdot \ldots]$ y $\mathsf{T} \in \sigma$ para denotar $\sigma = \sigma_0 \cdot \mathsf{T} \cdot \sigma_1$. Consideremos los siguientes ejemplos de uso.

▶ **Ejemplo 3.1.1.** *Sea* $\sigma = [(x \hookleftarrow 68) \cdot (y \hookleftarrow 34)]^{\{\mathtt{i}_1, \mathtt{i}_2\}}_{\mathtt{i}_1}$ *un estado con una única transacción, la llamamos* T*, que contiene dos operaciones: la escritura del valor* 68 *en el objeto* x *seguida de la escritura de* 34 *en el objeto* y*. El subíndice* $\mathtt{i}_1$ *indica que* T *arribó a dicha réplica. El superíndice* $\{\mathtt{i}_1, \mathtt{i}_2\}$ *indica que naturalmente la transacción es conocida por su réplica de llegada, pero que además fue propagada a una nueva réplica llamada* $\mathtt{i}_2$*.*

▶ **Ejemplo 3.1.2.** *Sea* $\sigma = [(x \hookleftarrow 25)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(x \hookleftarrow 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2}$*. En este caso, el sistema posee dos transacciones, que denominamos* T_1 *y* T_2 *respectivamente. Cada una impacta a* x *con un valor distinto, pero también lo hacen en réplicas distintas,* $\mathtt{i}_1$ *y* $\mathtt{i}_2$ *respectivamente. Ninguna de las transacciones fue propagada, por ende uno podría describir el sistema*

equivalentemente como $\sigma' = [(x \leftarrow\!\shortmid 10)]_{\mathtt{i}_2}^{\{\mathtt{i}_2\}} \cdot [(x \leftarrow\!\shortmid 25)]_{\mathtt{i}_1}^{\{\mathtt{i}_1\}}$. *Es decir, intercambiamos transacciones concurrentes. Esto motiva la siguiente definición de equivalencia de estados.*

Definición 2 (**Equivalencia de estados**). *La equivalencia de estados* $\equiv$ *es una relación de congruencia sobre* $\mathbb{T}^*$ *tal que* $\mathsf{T} \cdot \mathsf{T}' \equiv \mathsf{T}' \cdot \mathsf{T} \iff \mathsf{T}.\mathtt{X} \cap \mathsf{T}'.\mathtt{X} = \emptyset \ \vee \ \mathsf{T}.\mathtt{I} \cap \mathsf{T}'.\mathtt{I} = \emptyset$.

Concretamente, el orden de las transacciones en un sistema no es determinante en la medida que las mismas actúen sobre distintos objetos o sean conocidas por conjuntos de réplicas distintos. Cuando ese caso ocurre, las operaciones son totalmente independientes entre sí, y cuál ocurrió primero no afecta el resultado final. Caso contrario, las transacciones deben ser *arbitradas*. Es decir, se debe definir un orden relativo entre ellas que no puede ser alterado en el futuro.

Definición 3 (**Estado arbitrado**). *Un estado está arbitrado, escrito como* $\mathtt{arb}(\sigma)$*, si se cumple* $\mathtt{arb}(\sigma)$ *iff.* $\forall \sigma'.\sigma \equiv \sigma' \implies \sigma = \sigma'$

3.2. Semántica Operacional

La semántica operacional estará definida a partir de: (i) un sistema de transiciones etiquetadas, ilustrado en Fig. 3.1, y (ii) la relación de equivalencia de estados introducida en Def. 2. El sistema de transición se aplica sobre $\mathbb{T}^*$ y las siguientes etiquetas:

$$\beta \ ::= \ \alpha \mid \tau \qquad\qquad \alpha \ ::= \ \mathsf{T}_{\mathtt{i}}$$

Las transiciones τ se utilizan cuando el sistema realiza una operación interna, mientras que α ocurre cuando una transacción es commiteada al sistema. Aquí hacemos uso del abuso de notación previamente mencionado para utilizar como label $\mathsf{T}_{\mathtt{i}}$, una transacción T que llega a la réplica $\mathtt{i}$ sin la información de $\mathtt{I}$. Como es usual, utilizamos $\implies$ para indicar $(\longrightarrow)^*$, es decir la aplicación de 0 o más transiciones.

Para moldear el comportamiento de los estados definimos los siguientes predicados abstractos:

- $\Subset \subseteq \mathbb{T}^* \times \mathbb{T}$ (*commit consistency*): $\sigma \Subset \mathsf{T}$ se cumple cuando el sistema σ admite T.
- $\dagger \subseteq \mathbb{T}^* \times \mathbb{T} \times \mathbb{T}^*$ (*arbitration consistency*): $\dagger(\sigma_0, \mathsf{T}, \sigma_1)$ se cumple cuando un sistema σ puede ser escrito como $\sigma_0 \cdot \mathsf{T} \cdot \sigma_1$, admitiendo que T sea arbitrada después de σ_0 y antes de σ_1.

A través de la parametrización de los mismos, se pueden definir invariantes que el sistema debe respetar, permitendo especificar distintas garantías de consistencia. Ambos serán evaluados dentro de las reglas definidas en Fig. 3.1, que describen la transformación de estados para sistemas transaccionales replicados.

La regla COMMIT explica cómo se agrega una nueva transacción al sistema, exigiendo que el siguiente predicado sea verdadero $\sigma \Subset \mathsf{T}$. Este es el único mecanismo por el cuál pueden hacerse efectivas nuevas transacciones, que son inmutables en cuánto a la réplica inicial y los cambios que impactan en el estado. Inicialmente, sólo es conocida por la réplica a la cuál llega y por lo tanto $\mathsf{T}.\mathtt{I} = \{\mathsf{T}.\mathtt{i}\}$.

La regla PROPAGATION describe la propagación asincrónica capturando cuando una transacción conoce y se ordena sobre otras transacciones. Cuando una transacción T se propaga a una réplica $\mathtt{i}$, entonces se agrega la réplica $\mathtt{i}$ al conjunto de réplicas que conocen a T,

$$\frac{\sigma \circledS \mathsf{T}_{\mathtt{i}}^{\{\mathtt{i}\}}}{\sigma \xrightarrow{\mathsf{T}_{\mathtt{i}}} \sigma \cdot \mathsf{T}_{\mathtt{i}}^{\{\mathtt{i}\}}}\ (\textsc{commit}) \qquad \frac{\sigma \equiv \sigma_0 \quad \sigma_0 \xrightarrow{\tau} \sigma_0' \quad \sigma_0' \equiv \sigma'}{\sigma \xrightarrow{\tau} \sigma'}\ (\textsc{str})$$

$$\frac{\mathsf{T}' := \mathsf{T}[\mathtt{I} \mapsto \mathsf{T}.\mathtt{I} \cup \{\mathtt{i}\}] \qquad \dagger(\sigma_0, \mathsf{T}', \sigma_1) \qquad \mathtt{arb}(\sigma_0 \cdot \mathsf{T}' \cdot \sigma_1) \qquad \sigma_1 \cdot \sigma_2 \equiv \sigma_2 \cdot \sigma_1 \qquad \forall \mathsf{S} \in \sigma_2.\mathtt{i} \notin \mathsf{S}.\mathtt{I}}{\sigma_0 \cdot \mathsf{T} \cdot \sigma_1 \cdot \sigma_2 \xrightarrow{\tau} \sigma_0 \cdot \mathsf{T}' \cdot \sigma_1 \cdot \sigma_2}\ (\textsc{propagation})$$

Fig. 3.1: Semántica operacional para un sistema

TransVis (TV)	**NoConflict** (NC)
$\sigma \circledS \mathsf{T}$ iff $\forall \mathsf{S} \in \sigma.\ \mathsf{T}.\mathtt{i} \in \mathsf{S}.\mathtt{I} \implies \mathsf{T}.\mathtt{I} \subseteq \mathsf{S}.\mathtt{I}$ (3.1) $\dagger(\sigma_0, \mathsf{T}, \sigma_1)$ iff $\sigma_0 \circledS \mathsf{T}$ (3.2)	$\sigma \circledS \mathsf{T}$ iff $\forall \mathsf{S} \in \sigma.\ \mathsf{T}.\mathtt{X} \cap \mathsf{S}.\mathtt{X} \neq \emptyset \implies \mathsf{T}.\mathtt{i} \in \mathsf{S}.\mathtt{I}$ (3.3) $\dagger(\sigma_0, \mathsf{T}, \sigma_1)$ iff $\sigma_0 \circledS \mathsf{T}$ (3.4)
Prefix (PP)	**TotalVis** (TOTV)
$\sigma \circledS \mathsf{T}$ iff $\sigma \neq \epsilon \implies \sigma = \sigma_0 \cdot \mathsf{S} \cdot \sigma_1 \wedge \sigma_0 \cdot \mathsf{S} \bowtie \sigma_1 \wedge (\mathsf{T}.\mathtt{I} \subseteq \mathsf{S}.\mathtt{I} \vee \sigma.\mathtt{I} \cap \mathsf{T}.\mathtt{I} = \emptyset)$ (3.5) $\dagger(\sigma_0, \mathsf{T}, \sigma_1)$ iff $\sigma_0 \circledS \mathsf{T} \wedge (\forall \mathsf{S} \in \sigma_1.\ \mathsf{S}.\mathtt{I} \subseteq \mathsf{T}.\mathtt{I})$ (3.6)	$\sigma \circledS \mathsf{T}$ iff $\forall \mathsf{S} \in \sigma.\ \mathsf{T}.\mathtt{I} \subseteq \mathsf{S}.\mathtt{I}$ (3.7) $\dagger(\sigma_0, \mathsf{T}, \sigma_1)$ iff $\sigma_0 \circledS \mathsf{T}$ (3.8)

Fig. 3.2: Axiomas de consistencia

Causal Consistency (CC)	**Parallel Snapshot Isolation** (PSI)
$\sigma \circledS \mathsf{T}$ iff $\forall \mathsf{S} \in \sigma.\ \mathsf{T}.\mathtt{i} \in \mathsf{S}.\mathtt{I} \implies \mathsf{T}.\mathtt{I} \subseteq \mathsf{S}.\mathtt{I}$ (3.9) $\dagger(\sigma_0, \mathsf{T}, \sigma_1)$ iff $\sigma_0 \circledS \mathsf{T}$ (3.10)	$\sigma \circledS \mathsf{T}$ iff $\forall \mathsf{S} \in \sigma.\ (\mathsf{T}.\mathtt{X} \cap \mathsf{S}.\mathtt{X} \neq \emptyset \vee \mathsf{T}.\mathtt{i} \in \mathsf{S}.\mathtt{I}) \implies \mathsf{T}.\mathtt{I} \subseteq \mathsf{S}.\mathtt{I}$ (3.11) $\dagger(\sigma_0, \mathsf{T}, \sigma_1)$ iff $\sigma_0 \circledS \mathsf{T}$ (3.12)
Prefix Consistency (PC)	**Serialisability** (SER)
$\sigma \circledS \mathsf{T}$ iff $\sigma \neq \epsilon \implies \sigma = \sigma_0 \cdot \mathsf{S} \cdot \sigma_1 \wedge \sigma_0 \cdot \mathsf{S} \bowtie \sigma_1 \wedge (\mathsf{T}.\mathtt{I} \subseteq \mathsf{S}.\mathtt{I} \vee \sigma.\mathtt{I} \cap \mathsf{T}.\mathtt{I} = \emptyset)$ (3.13) $\dagger(\sigma_0, \mathsf{T}, \sigma_1)$ iff $\sigma_0 \circledS \mathsf{T} \wedge (\forall \mathsf{S} \in \sigma_1.\ \mathsf{S}.\mathtt{I} \subseteq \mathsf{T}.\mathtt{I})$ (3.14)	$\sigma \circledS \mathsf{T}$ iff $\forall \mathsf{S} \in \sigma.\ \mathsf{T}.\mathtt{I} \subseteq \mathsf{S}.\mathtt{I}$ (3.15) $\dagger(\sigma_0, \mathsf{T}, \sigma_1)$ iff $\sigma_0 \circledS \mathsf{T}$ (3.16)
Snapshot Isolation (SI)	
$\sigma \circledS \mathsf{T}$ iff $(\sigma \neq \epsilon \implies \sigma = \sigma_0 \cdot \mathsf{S} \cdot \sigma_1 \wedge \sigma_0 \cdot \mathsf{S} \bowtie \sigma_1 \wedge (\mathsf{T}.\mathtt{I} \subseteq \mathsf{S}.\mathtt{I} \vee \sigma.\mathtt{I} \cap \mathsf{T}.\mathtt{I} = \emptyset)) \wedge (\forall \mathsf{S} \in \sigma.\ \mathsf{T}.\mathtt{X} \cap \mathsf{S}.\mathtt{X} \neq \emptyset \implies \mathsf{T}.\mathtt{i} \in \mathsf{S}.\mathtt{I})$ (3.17) $\dagger(\sigma_0, \mathsf{T}, \sigma_1)$ iff $\sigma_0 \circledS \mathsf{T} \wedge (\forall \mathsf{S} \in \sigma_1.\ \mathsf{S}.\mathtt{I} \subseteq \mathsf{T}.\mathtt{I})$ (3.18)	

Fig. 3.3: Modelos de consistencia

es decir, $\mathsf{T}[\mathtt{I} \mapsto \mathsf{T}.\mathtt{I} \cup \{\mathtt{i}\}]$. La operación exige $\mathtt{arb}(\sigma_0 \cdot \mathsf{T}' \cdot \sigma_1)$, lo que expresa que σ_0 y σ_1, los subestados *a priori* y *a posteriori*, quedaron ordenados tal que ocurren antes y después respectivamente. Esta arbitración debe ser admitida por σ_0 y σ_1, por ende se exige $\dagger(\sigma_0, \mathsf{T}', \sigma_1)$. La evaluación del predicado podría forzar a que las propagaciones se deban hacer en un orden u otro para poder ser válidas. Finalmente, la condición $\forall \mathsf{T}' \in \sigma_2.\mathtt{i} \notin \mathsf{T}'.\mathtt{I}$ obliga a arbitrar T relativo al resto de las transacciones que se encuentran en la réplica $\mathtt{i}$.

Regla STR permite el libre reordenamiento de transacciones que no presentan orden entre sí. Es decir, el estado resultante posterior al reordenamiento es equivalente.

3.3. Consideraciones sobre el framework

Luego de dar la semántica operacional remarcamos las decisiones de diseño tomadas con el fin de obtener un modelo simple y compacto que brinde lo necesario para generar las trazas que se explican con un modelo axiomático.

- Existe un *gap* entre ésta semántica y una que refleje la semántica de programas con transacciones. Estos últimos consideran primitivas para comenzar una transacción, realizando un *lock* de la réplica hasta que se commitea. Otra característica es que las transacciones no necesariamente son secuencias de escrituras sobre objetos que son registros. En el final de este trabajo discutiremos sobre agregar estas reglas, sin embargo remarcamos que agregarlas no cambian nuestros resultados, ya que lo importante ocurre cuando se commitea y se propaga una transacción, el resto de las reglas no cambian el estado del sistema.
- No hay rollback, lo que posibilita que las reglas que modifican el sistema, COMMIT y PROPAGATION, operen de manera *straightforward* sin considerar la posibilidad de errorores.

3.4. Modelos de consistencia transaccionales sobre el modelo operacional

Estamos en condiciones de explicar según nuestro modelo operacional como capturar los modelos de consistencia transaccionales enunciados y descriptos sobre ejecuciones abstractas. Las definiciones son descriptas en Fig. 3.2 y Fig. 3.3. Mapeamos los modelos y los axiomas a una instanciación de los operadores *commit consistency* y *arbitration consistency*. Algunos modelos de consistencia son enunciados como la conjunción de axiomas, detallados en Tab. 2.1. Por ejemplo, $\texttt{SI} = \texttt{PP} \wedge \texttt{NC}$ y $\texttt{SER} = \texttt{TOTV}$.

Nos referimos a la *transacción entrante* como la transacción T que aparece en la especificación de los predicados abstractos, dentro de cada modelo. Análogamente, llamamos *transacciones examinadas* a las transacciones S y S' que son utilizadas para iterar al resto de las transacciones preexistentes del sistema. Finalmente, *el sistema* o *el estado* son sinónimos para σ.

La especificación de TRANSVIS (TV) exige que la transacción entrante, si ocurre en la misma réplica que conoce a una transacción examinada, debe ser *visible* a réplicas que la examinada también lo es. Este invariante busca preservar la historia de transacciones visibles entre sí. El caso en el que T está siendo commiteada es trivial puesto que $\mathsf{T}.\texttt{I} = \{\texttt{i}\}$, pero no así cuando es propagada. En dicho caso, se fuerza a que una transacción no pueda "dejar de ver" a otra transacción en el futuro. Como la condición $\mathsf{T}.\texttt{i} \in \mathsf{S}.\texttt{I}$ se cumple a lo largo de la ejecución, ya que no hay reglas que permitan borrar información, esto obliga a que la historia de visibilidad se construya en orden en cada réplica. Si uno quisiera propagar T a una réplica que no conocía a S, previamente debe informar de S a la misma. Para hacer eso, también se debe propagar la historia de S. Es decir, la historia visible se difunde de manera transitiva.

El especificación de NOCONFLICT (NC) opera sobre transacciones que escriben un mismo subconjunto de objetos. La transacción entrante T, de compartir variables con una transacción examinada S, debe realizarse en una réplica que conozca a la examinada. De esta forma, imposibilita las escrituras *concurrentes* a un mismo objeto. Como depende de propiedades, $\mathsf{T}.\texttt{i}$ y $\mathsf{T}.\texttt{X}$, que no pueden ser alteradas una vez que una transacción ingresa al sistema, alcanza con corroborar este axioma al momento de COMMIT.

La especificación de Prefix (PP) garantiza que a medida que las réplicas comienzan a decidir un orden sobre las transacciones, todas las réplicas deben respetar el mismo orden. De esta manera, si se entiende al estado como una secuencia de transacciones, todas las réplicas comparten un prefijo del estado. Comenzamos analizando la instanciación de *commit consistency* y para esto presentamos el siguiente predicado infijo que da las condiciones que tiene que cumplir un estado $\sigma = \sigma_0 \cdot \sigma_1$ con al menos un elemento para poder agregar una transacción:

$$\sigma_0 \bowtie \sigma_1 = \mathtt{arb}(\sigma_0) \ \wedge \ \sigma_0.\mathtt{I} \cap \sigma_1.\mathtt{I} = \emptyset \ \wedge \ \forall \mathsf{S}, \mathsf{S}' \in \sigma_1.\mathsf{S}.\mathtt{I} \cap \mathsf{S}'.\mathtt{I} = \emptyset$$

El predicado requiere que: (i) las transacciones de σ_0 no se puedan reordenar, (ii) describe que ninguna transacción en σ_1 es conocida por las réplicas que conocen las transacciones en σ_0. Finalmente, la condición (iii) significa que las transacciones en σ_1 no se conocen. Por lo tanto, este $\bowtie$ describe las condiciones necesarias sobre la estructura de σ para contener un prefijo que fue decidido.
Entonces, la definición la podemos descomponer como una disyunción de dos proposiciones. La primera proposición:

$$\sigma_0 \cdot \mathsf{S} \bowtie \sigma_1 \ \wedge \ \mathsf{T}.\mathtt{I} \subseteq \mathsf{S}.\mathtt{I}$$

La misma expresa que cuando arriba T el estado $\sigma = \sigma_0 \cdot \mathsf{S} \cdot \sigma_1$, debe ser correcto y T sólo podrá ser entregado a una réplica que conozca al prefijo de σ. Ls segunda parte de la diyunción, es si no se entrega en una réplica que conozca al prefijo de σ. La única posibilidad es que se entregue sobre una réplica fresca, a espera que el sistema decida arbitrarla. Por lo tanto, la segunda diyunción es:

$$\sigma_0 \cdot \mathsf{S} \bowtie \sigma_1 \ \wedge \ \sigma.\mathtt{I} \cap \mathsf{T}.\mathtt{I} = \emptyset$$

Por último, procedemos al análisis de *arbitration consistency*. La intuición para esta regla es que como ya se decidió un prefijo del estado del sistema, no se puede reordenar. Por lo tanto, si $\sigma_0 \cdot \mathsf{T} \cdot \sigma_1$ ya fue arbitrado, no es posible que se reordene este prefijo aceptando una transacción nueva. Cualquier transacción que no pertenecía al prefijo, solo puede agregarse al final. Por lo tanto, vamos a exigir que (i) $\sigma_0 \circledS \mathsf{T}'$, explicado anteriormente, y además (ii) $\forall \mathsf{S} \in \sigma_1.\mathsf{S}.\mathtt{I} \subseteq \mathsf{T}.\mathtt{I}$. Esta última condición obliga a propagar las transacciones de un prefijo, de izquierda a derecha. La conjunción de estas condiciones garantiza que las transacciones entrantes observen las historias (o prefijos) preexistentes en el orden en el que ocurrieron.

La especificación de TotalVis (TOTV) obliga a toda transacción entrante a tener visibilidad de todo el sistema. En todo momento, las réplicas en las cuáles T es visible, no pueden desconocer a ninguna S del estado. Para extender el conjunto $\mathsf{T}.\mathtt{I}$ con una réplica fresca uno debe propagar *todo* el estado previo a T, en el orden correspondiente, a la misma. Por lo tanto, fuerza una historia serializable.

3.5. Ejemplos

A continuación, mostramos cómo modelar escenarios concretos a partir del modelo operacional desarrollado. Asumimos en cada uno de los ejemplos que inicialmente los objetos son inicializados en cero, es decir, cada réplica posee una copia de cada objeto, inicializados con el valor cero.

Observación 1. *Usamos la notación* $\xrightarrow{\texttt{COMM}}$, $\xrightarrow{\texttt{PROP}}$ *o* $\xrightarrow{\texttt{STR}}$ *para indicar que regla aplicada para esa reducción es* COMMIT, PROPAGATION *o* STR *respectivamente.*

Observación 2. *En cada ejemplo, inicialmente no son aplicadas restricciones de consistencia. Es decir,*

$$\sigma \Subset \mathsf{T} \textit{ iff } \texttt{true} \ \wedge \ \dagger(\sigma_0, \mathsf{T}, \sigma_1) \textit{ iff } \texttt{true}$$

▶ **Ejemplo 3.5.1.** *Considere la descripción de la ejecución presentada en Ej. 2.5.1.*

T_1	T_2	T_3
`x := 25` `y := 100`	`x := x + 50`	`y := y + 100`

Mostramos paso a paso una traza en la que, inicialmente, T_1 y T_2 son conocidas por la misma réplica $\mathtt{i}_1$. Más adelante, ocurre que únicamente la transacción T_2 es propagada a la réplica $\mathtt{i}_2$, haciendo que la futura transacción T_3, cuya réplica inicial es también $\mathtt{i}_2$, tenga solamente conocimiento de T_2.

Comenzamos por la ejecución de T_1 en la réplica $\mathtt{i}_1$

$$\epsilon \xrightarrow{\texttt{COMM}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\mathtt{i}_1}^{\{\mathtt{i}_1\}}$$

Luego, ocurre lo mismo con T_2 en $\mathtt{i}_1$

$$\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\mathtt{i}_1}^{\{\mathtt{i}_1\}} \cdot [(x \hookleftarrow 75)]_{\mathtt{i}_1}^{\{\mathtt{i}_1\}}$$

Próximo paso, la propagación de T_2 a $\mathtt{i}_2$

$$\xrightarrow{\texttt{PROP}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\mathtt{i}_1}^{\{\mathtt{i}_1\}} \cdot [(x \hookleftarrow 75)]_{\mathtt{i}_1}^{\{\mathtt{i}_1,\mathtt{i}_2\}}$$

Por último, como T_3 no es consciente de T_1, se produce la actualización de y a 100

$$\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\mathtt{i}_1}^{\{\mathtt{i}_1\}} \cdot [(x \hookleftarrow 75)]_{\mathtt{i}_1}^{\{\mathtt{i}_1,\mathtt{i}_2\}} \cdot [(y \hookleftarrow 100)]_{\mathtt{i}_2}^{\{\mathtt{i}_2\}}$$

La traza anómala resultante

$$\begin{aligned}
\epsilon &\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\mathtt{i}_1}^{\{\mathtt{i}_1\}} \\
&\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\mathtt{i}_1}^{\{\mathtt{i}_1\}} \cdot [(x \hookleftarrow 75)]_{\mathtt{i}_1}^{\{\mathtt{i}_1\}} \\
&\xrightarrow{\texttt{PROP}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\mathtt{i}_1}^{\{\mathtt{i}_1\}} \cdot [(x \hookleftarrow 75)]_{\mathtt{i}_1}^{\{\mathtt{i}_1,\mathtt{i}_2\}} \\
&\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\mathtt{i}_1}^{\{\mathtt{i}_1\}} \cdot [(x \hookleftarrow 75)]_{\mathtt{i}_1}^{\{\mathtt{i}_1,\mathtt{i}_2\}} \cdot [(y \hookleftarrow 100)]_{\mathtt{i}_2}^{\{\mathtt{i}_2\}}
\end{aligned}$$

Veamos qué ocurre bajo el modelo de consistencia Causal Consistency (`CC`). *Para ello, debemos ver cómo se modificaría la traza cuando el predicado commit consistency evalúa* `CC`. *El paso de propagación de T_2 a $\mathtt{i}_2$ es imposibilitado por la evaluación de $\dagger$ en* propagation, *tomando $\sigma_0 = \mathsf{T}_1$. Se verifica $\mathsf{T}_2.\mathtt{i} \in \mathsf{T}_1.\mathtt{I}$, pero no el consecuente $\mathsf{T}_2.\mathtt{I} \subseteq \mathsf{T}_1.\mathtt{I}$. Por lo tanto, para continuar con el escenario, es requisito previo propagar T_1 a $\mathtt{i}_2$.*

$$\begin{aligned}
\epsilon &\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\mathtt{i}_1}^{\{\mathtt{i}_1\}} \\
&\xrightarrow{\texttt{PROP}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\mathtt{i}_1}^{\{\mathtt{i}_1,\mathtt{i}_2\}}
\end{aligned}$$

Bajo este contexto, es lícito propagar T_2 a $\mathtt{i}_2$

$$\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\texttt{i}_1}^{\{\texttt{i}_1,\texttt{i}_2\}} \cdot [(x \hookleftarrow 75)]_{\texttt{i}_1}^{\{\texttt{i}_1\}}$$
$$\xrightarrow{\texttt{PROP}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\texttt{i}_1}^{\{\texttt{i}_1,\texttt{i}_2\}} \cdot [(x \hookleftarrow 75)]_{\texttt{i}_1}^{\{\texttt{i}_1,\texttt{i}_2\}}$$

Finalmente, la anomalía se previene logrando que T_3 *sea impactada correctamente actualizando el objeto* y *con el valor* 200

$$\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\texttt{i}_1}^{\{\texttt{i}_1,\texttt{i}_2\}} \cdot [(x \hookleftarrow 75)]_{\texttt{i}_1}^{\{\texttt{i}_1,\texttt{i}_2\}} \cdot [(y \hookleftarrow 200)]_{\texttt{i}_2}^{\{\texttt{i}_2\}}$$

La traza completa sin la presencia de la anomalía es

$$\begin{aligned}
\epsilon &\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\texttt{i}_1}^{\{\texttt{i}_1\}} \\
&\xrightarrow{\texttt{PROP}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\texttt{i}_1}^{\{\texttt{i}_1,\texttt{i}_2\}} \\
&\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\texttt{i}_1}^{\{\texttt{i}_1,\texttt{i}_2\}} \cdot [(x \hookleftarrow 75)]_{\texttt{i}_1}^{\{\texttt{i}_1\}} \\
&\xrightarrow{\texttt{PROP}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\texttt{i}_1}^{\{\texttt{i}_1,\texttt{i}_2\}} \cdot [(x \hookleftarrow 75)]_{\texttt{i}_1}^{\{\texttt{i}_1,\texttt{i}_2\}} \\
&\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 25) \cdot (y \hookleftarrow 100)]_{\texttt{i}_1}^{\{\texttt{i}_1,\texttt{i}_2\}} \cdot [(x \hookleftarrow 75)]_{\texttt{i}_1}^{\{\texttt{i}_1,\texttt{i}_2\}} \cdot [(y \hookleftarrow 200)]_{\texttt{i}_2}^{\{\texttt{i}_2\}}
\end{aligned}$$

Notar que no es la única posible. Alcanza con que la propagación de T_1 *a* $\texttt{i}_2$ *preceda la de* T_2 *a* $\texttt{i}_2$ *para que la traza sea consistente con el modelo. Tampoco es único el modelo de consistencia que previene esta anomalía. También lo hace* Parallel Snapshot Isolation (PSI)*. Basta ver que* PSI $\implies$ CC*, lo que es simple de corroborar puesto que el antecedente de* PSI *implica al de* CC*, y poseen el mismo consecuente.*

▶ **Ejemplo 3.5.2.** *Analizamos la ejecución introducida Ej. 2.5.2 con la siguiente descripción.*

T_1	T_2
`x := x + 50`	`x := x + 25`

Detallamos una traza que en primera instancia impacta T_1 *en* $\texttt{i}_1$*, cuyas modificaciones luego son dejadas sin consideración por* T_2*, que tiene como réplica inicial a* $\texttt{i}_2$ *y posteriormente arriba a* $\texttt{i}_1$ *vía propagación.*

Inicialmente, el sistema recibe T_1 *y decide impactarla en* $\texttt{i}_1$

$$\epsilon \xrightarrow{\texttt{COMM}} [(x \hookleftarrow 50)]_{\texttt{i}_1}^{\{\texttt{i}_1\}}$$

Seguido, el mismo proceso ocurre con T_2 *en la réplica* $\texttt{i}_2$*. Al no conocer a* T_1 *actualiza con el valor* 25 *al objeto* x

$$\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 50)]_{\texttt{i}_1}^{\{\texttt{i}_1\}} \cdot [(x \hookleftarrow 25)]_{\texttt{i}_2}^{\{\texttt{i}_2\}}$$

Esto conlleva a que, al propagar T_2 *a* $\texttt{i}_1$*, el último valor observable de* x *sea* 25

$$\xrightarrow{\texttt{PROP}} [(x \hookleftarrow 50)]_{\texttt{i}_1}^{\{\texttt{i}_1\}} \cdot [(x \hookleftarrow 25)]_{\texttt{i}_2}^{\{\texttt{i}_2,\texttt{i}_1\}}$$

Recopilando todas las transiciones, obtenemos la traza completa

$$\begin{aligned}\epsilon &\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 50)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \\ &\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 50)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(x \hookleftarrow 25)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \\ &\xrightarrow{\texttt{PROP}} [(x \hookleftarrow 50)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(x \hookleftarrow 25)]^{\{\mathtt{i}_2,\mathtt{i}_1\}}_{\mathtt{i}_2}\end{aligned}$$

La inconsistencia se presenta a raíz de que tanto T_1 *como* T_2 *operan sobre un mismo conjunto de variables y al poner en común los cambios, en este caso al propagarse* T_2 *a* $\mathtt{i}_1$*, el estado resulta inconsistente con la intención original. Analizamos la traza bajo el modelo de consistencia* Parallel Snapshot Isolation (`PSI`)*.*

El primer paso ocurre sin alteraciones

$$\epsilon \xrightarrow{\texttt{COMM}} [(x \hookleftarrow 50)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1}$$

Pero el escenario cambia al querer impactar T_2 *en* $\mathtt{i}_2$*. La evaluación de* $\Subset$ *en* commit *prohíbe la transición original puesto que* $\mathsf{T}_2.\mathtt{X} \cap \mathsf{T}_1.\mathtt{X} = \{x\} \neq \emptyset$ *pero no se cumpliría* $\mathsf{T}_2.\mathtt{I} = \{\mathtt{i}_2\} \not\subseteq \mathsf{T}_1.\mathtt{I}$*. La única forma en la que la ejecución puede continuar es garantizando que* T_1 *sea conocida por* $\mathtt{i}_2$

$$\xrightarrow{\texttt{PROP}} [(x \hookleftarrow 50)]^{\{\mathtt{i}_1,\mathtt{i}_2\}}_{\mathtt{i}_1}$$

Esto posibilita la entrada de T_2 *al sistema sobre la réplica* $\mathtt{i}_2$*, pero en esta ocasión la escritura de* x *se lleva adelante con el valor* 75*, debido al conocimiento de* T_2 *sobre* T_1

$$\begin{aligned}&\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 50)]^{\{\mathtt{i}_1,\mathtt{i}_2\}}_{\mathtt{i}_1} \cdot [(x \hookleftarrow 75)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \\ &\xrightarrow{\texttt{PROP}} [(x \hookleftarrow 50)]^{\{\mathtt{i}_1,\mathtt{i}_2\}}_{\mathtt{i}_1,} \cdot [(x \hookleftarrow 75)]^{\{\mathtt{i}_2,\mathtt{i}_1\}}_{\mathtt{i}_2}\end{aligned}$$

En conclusión, la anomalía se resuelve por las restricciones que propone el modelo en cuestión, con la traza resultante siendo

$$\begin{aligned}\epsilon &\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 50)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \\ &\xrightarrow{\texttt{PROP}} [(x \hookleftarrow 50)]^{\{\mathtt{i}_1,\mathtt{i}_2\}}_{\mathtt{i}_1} \\ &\xrightarrow{\texttt{COMM}} [(x \hookleftarrow 50)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(x \hookleftarrow 75)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \\ &\xrightarrow{\texttt{PROP}} [(x \hookleftarrow 50)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(x \hookleftarrow 75)]^{\{\mathtt{i}_2,\mathtt{i}_1\}}_{\mathtt{i}_2}\end{aligned}$$

Análogo a lo que ocurre en Ej. 3.5.1, el predicado Snapshot Isolation (`SI`) *también previene esta irregularidad. Nuevamente, ocurre que* $\texttt{SI} \implies \texttt{PSI}$*.*

▶ **Ejemplo 3.5.3.** *Consideramos la ejecución, previamente presentada en Ej. 2.5.3, cuya descripción es*

T_1	T_2	T_3	T_4
`x := 10`	`y := 10`	`z1 := y + 30`	`z2 := x + 40`

En base a ésta descripción, presentaremos una traza en la cuál T_1 *es conocida por* T_3 *y* T_2 *es conocida por* T_4*. Luego, se propaga* T_1 *hacia* $\mathtt{i}_2$*, decidiendo que* T_1 *sea ordenado antes que* T_2*. Esto provoca una reescritura de la historia conocida por* T_4*, cuyo comportamiento hubiese cambiado en caso de haber tenido conocimiento sobre* T_1*.*

La traza inicia con el ingreso de T_1 *y* T_2 *a las réplicas* $\mathtt{i}_1$ *e* $\mathtt{i}_2$*, respectivamente*

$$\begin{aligned}\epsilon &\xrightarrow{\text{COMM}} [(x \hookleftarrow 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \\ &\xrightarrow{\text{COMM}} [(x \hookleftarrow 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \hookleftarrow 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2}\end{aligned}$$

Los pasos siguientes son simétricos sobre las transacciones T_3 *y* T_4

$$\begin{aligned}&\xrightarrow{\text{COMM}} [(x \hookleftarrow 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \hookleftarrow 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \cdot [(z_1 \hookleftarrow 30)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \\ &\xrightarrow{\text{COMM}} [(x \hookleftarrow 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \hookleftarrow 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \cdot [(z_1 \hookleftarrow 30)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(z_2 \hookleftarrow 40)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2}\end{aligned}$$

En este último paso ocurre la reescritura: se propaga T_1 *a* $\mathtt{i}_2$*, arbitrando* T_1 *antes de* T_2

$$\xrightarrow{\text{PROP}} [(x \hookleftarrow 10)]^{\{\mathtt{i}_1,\mathtt{i}_2\}}_{\mathtt{i}_1} \cdot [(y \hookleftarrow 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \cdot [(z_1 \hookleftarrow 30)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(z_2 \hookleftarrow 40)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2}$$

Así, obtenemos como resultado la traza

$$\begin{aligned}\epsilon &\xrightarrow{\text{COMM}} [(x \hookleftarrow 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \\ &\xrightarrow{\text{COMM}} [(x \hookleftarrow 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \hookleftarrow 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \\ &\xrightarrow{\text{COMM}} [(x \hookleftarrow 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \hookleftarrow 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \cdot [(z_1 \hookleftarrow 30)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \\ &\xrightarrow{\text{COMM}} [(x \hookleftarrow 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \hookleftarrow 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \cdot [(z_1 \hookleftarrow 30)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(z_2 \hookleftarrow 40)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \\ &\xrightarrow{\text{PROP}} [(x \hookleftarrow 10)]^{\{\mathtt{i}_1,\mathtt{i}_2\}}_{\mathtt{i}_1} \cdot [(y \hookleftarrow 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \cdot [(z_1 \hookleftarrow 30)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(z_2 \hookleftarrow 40)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2}\end{aligned}$$

Para resolver la inconsistencia, debemos impedir que historias paralelas converjan de manera tardía alterando la historia previa de alguna de ellas. El modelo de consistencia Prefix Consistency (PC) *es el indicado para este tipo de escenarios. Revisitamos la traza aplicando dicho modelo.*

Análogo a otros ejemplos, el modelo permite que los primeros casos ocurran sin alteraciones. Es trivial chequear que el commit *de* T_1 *lo cumple ya que el estado en ese momento es vacío. Asimismo el caso de* T_2 *es sencillo: basta tomar* $\mathsf{S} = \mathsf{T}_2$ *y observar que el resto de las condiciones se cumplen trivialmente puesto que* $\sigma_0 = \sigma_1 = \epsilon$*. No tan trivial es el caso de* T_3*, para verlo tomamos* $\sigma_0 = \epsilon$*,* $\mathsf{S} = \mathsf{T}_1$ *y* $\sigma_1 = \mathsf{T}_2$*. Así, el predicado* $\bowtie$ *se verifica puesto que las condiciones de que dependen de* σ_0 *se vuelven triviales y la restante también, que depende únicamente de* σ_1*, debido a que posee una única transacción. Finalmente, como* $\mathsf{T}_3 = \mathsf{T}$*, se cumple* $\mathsf{T}.\mathtt{I} = \{\mathtt{i}_1\} \subseteq \mathsf{S}.\mathtt{I}$*. Por lo tanto, los commits de dichas transacciones ocurren sin problemas*

$$\begin{aligned}\epsilon &\xrightarrow{\text{COMM}} [(x \hookleftarrow 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \\ &\xrightarrow{\text{COMM}} [(x \hookleftarrow 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \hookleftarrow 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \\ &\xrightarrow{\text{COMM}} [(x \hookleftarrow 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \hookleftarrow 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \cdot [(z_1 \hookleftarrow 30)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1}\end{aligned}$$

En este punto, el modelo prohíbe el commit de T_4 *a* $\mathtt{i}_2$*. Si* $\mathsf{S} = \mathsf{T}_1$ *o* $\mathsf{S} = \mathsf{T}_3$*, nunca podrá cumplir* $\mathsf{T}_4.\mathtt{I} \subseteq \mathsf{S}.\mathtt{I}$*. Caso contrario,* $\mathsf{S} = \mathsf{T}_2$*, pero ninguna combinación de* σ_0 *y* σ_1 *posible logra satisfacer el resto de las condiciones. Las vías por las cuáles sí se habilitaría el ingreso de* T_4 *al sistema son (i) que* T_4 *arribe a una réplica fresca y (ii) que arribe a* $\mathtt{i}_1$*. Exploramos la opción (ii) dado que la (i) es trivial.*

Previo a poder impactar T_4 *en* $\mathtt{i}_1$*, debemos reordernar las transacciones en* $\mathtt{i}_1$ *para poder satisfacer las condiciones de* $\bowtie$*. Básicamente, agrupamos el prefijo formado por* $\mathsf{T}_1 \cdot \mathsf{T}_3$ *a la izquierda*

$$\xrightarrow{\text{STR}} [(x \leftarrowtail 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(z_1 \leftarrowtail 30)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \leftarrowtail 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2}$$

El estado resultante permite llevar adelante la operación. Basta tomar $\mathsf{T}_1 \cdot \mathsf{T}_3 = \sigma_0 \cdot \mathsf{S}$ *y* $\sigma_1 = \mathsf{T}_2$ *para verificar las condiciones de* $\circledcirc$*. Ahora bien, el hecho de que* T_4 *arribe a* $\mathtt{i}_1$ *le permite observar el impacto de* T_1*, por lo tanto* $z_2 = x + 40 = 50$

$$\begin{aligned}
\epsilon \quad &\xrightarrow{\text{COMM}} [(x \leftarrowtail 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \\
&\xrightarrow{\text{COMM}} [(x \leftarrowtail 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \leftarrowtail 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \\
&\xrightarrow{\text{COMM}} [(x \leftarrowtail 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \leftarrowtail 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \cdot [(z_1 \leftarrowtail 30)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \\
&\xrightarrow{\text{STR}} [(x \leftarrowtail 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(z_1 \leftarrowtail 30)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \leftarrowtail 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \\
&\xrightarrow{\text{COMM}} [(x \leftarrowtail 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(z_1 \leftarrowtail 30)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \leftarrowtail 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \cdot [(z_2 \leftarrowtail 50)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1}
\end{aligned}$$

De esta manera, el model prohíbe la unión de estas historias de manera tardía. Así, se previene la anomalía, arrojando como traza final

$$\begin{aligned}
\epsilon \quad &\xrightarrow{\text{COMM}} [(x \leftarrowtail 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \\
&\xrightarrow{\text{COMM}} [(x \leftarrowtail 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \leftarrowtail 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \\
&\xrightarrow{\text{COMM}} [(x \leftarrowtail 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \leftarrowtail 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \cdot [(z_1 \leftarrowtail 30)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \\
&\xrightarrow{\text{STR}} [(x \leftarrowtail 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(z_1 \leftarrowtail 30)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \leftarrowtail 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \\
&\xrightarrow{\text{COMM}} [(x \leftarrowtail 10)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(z_1 \leftarrowtail 30)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \leftarrowtail 10)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2} \cdot [(z_2 \leftarrowtail 50)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1}
\end{aligned}$$

Comparte con el ejemplo anterior que el predicado Snapshot Isolation (SI) *prevee esta inconsistencia. De modo análogo, la propiedad* $\mathtt{SI} \implies \mathtt{PC}$ *se sostiene.*

▶ **Ejemplo 3.5.4.** *Examinamos la siguiente ejecución, descripta originalmente en Ej. 2.5.4.*

T_1	T_2	T_3

`x := 60`
`y := 60`
	`if (x + y > 100) {`
	`  x := x - 100`
	`}`
		`if (x + y > 100) {`
		`  y := y - 100`
		`}`

Proponemos, en base a la descripción, una traza bifurcal. En primera instancia, la transacción T_1 *altera el estado de los objetos* x *e* y*. La bifurcación de la historia desencadena en que las transacciones* T_2 *y* T_3 *ocurran de manera concurrente, con ambas observando únicamente las modificaciones hechas por* T_1*. En última instancia, desemboca en la errónea actualización a raíz de* T_3*.*

La traza comienza con el arribo de T_1 *a las réplicas* $\mathtt{i}_1$ *y* $\mathtt{i}_2$

$$\begin{aligned}
\epsilon \quad &\xrightarrow{\text{COMM}} [(x \leftarrowtail 60) \cdot (y \leftarrowtail 60)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \\
&\xrightarrow{\text{PROP}} [(x \leftarrowtail 60) \cdot (y \leftarrowtail 60)]^{\{\mathtt{i}_1, \mathtt{i}_2\}}_{\mathtt{i}_1}
\end{aligned}$$

Posteriormente, T_2 *altera el estado, previamente leyendo los objetos* x *e* y*, determinando que debe restar* 100 *a* x*, lo que es lo mismo que colocar* -40 *en dicho objeto*

$$\xrightarrow{\texttt{COMM}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]^{\{\mathtt{i}_1,\mathtt{i}_2\}}_{\mathtt{i}_1} \cdot [(x \leftarrow\!\shortmid -40)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1}$$

Como última transición, se presenta la anomalía proveniente de que T_3 *no conoce el impacto de* T_2*, proponiendo también una actualización a* -40 *del objeto* y

$$\xrightarrow{\texttt{COMM}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]^{\{\mathtt{i}_1,\mathtt{i}_2\}}_{\mathtt{i}_1} \cdot [(x \leftarrow\!\shortmid -40)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \leftarrow\!\shortmid -40)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2}$$

La cadena resultante final de la ejecución bajo el modelo operacional

$$\begin{aligned}
\epsilon &\xrightarrow{\texttt{COMM}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \\
&\xrightarrow{\texttt{PROP}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]^{\{\mathtt{i}_1,\mathtt{i}_2\}}_{\mathtt{i}_1} \\
&\xrightarrow{\texttt{COMM}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]^{\{\mathtt{i}_1,\mathtt{i}_2\}}_{\mathtt{i}_1} \cdot [(x \leftarrow\!\shortmid -40)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \\
&\xrightarrow{\texttt{COMM}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]^{\{\mathtt{i}_1,\mathtt{i}_2\}}_{\mathtt{i}_1} \cdot [(x \leftarrow\!\shortmid -40)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \cdot [(y \leftarrow\!\shortmid -40)]^{\{\mathtt{i}_2\}}_{\mathtt{i}_2}
\end{aligned}$$

Este tipo de ejecuciones no son contempladas por los anteriores modelos de consistencia puesto que las transacciones T_2 *y* T_3 *no comparten subconjunto de objetos ni réplicas conocidas. Ante este escenario, una forma posible de resolver la anomalía es a través de la serialización de la historia, garantía que provee el modelo* Serialisability (`SER`)*. El modelo prohibiría, en la última ejecución de* commit*, que la transacción* T_3 *se ejecute en la réplica* $\mathtt{i}_2$*. De esa manera* $\mathsf{T}_3.\mathtt{I} \nsubseteq \mathsf{T}_2.\mathtt{I}$*, llevando a la cadena a ser inconsistente con el modelo.*

Los primeros pasos ocurren como en la cadena original

$$\begin{aligned}
\epsilon &\xrightarrow{\texttt{COMM}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1} \\
&\xrightarrow{\texttt{PROP}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]^{\{\mathtt{i}_1,\mathtt{i}_2\}}_{\mathtt{i}_1} \\
&\xrightarrow{\texttt{COMM}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]^{\{\mathtt{i}_1,\mathtt{i}_2\}}_{\mathtt{i}_1} \cdot [(x \leftarrow\!\shortmid -40)]^{\{\mathtt{i}_1\}}_{\mathtt{i}_1}
\end{aligned}$$

Pero esta tendencia no se mantiene. Para que T_3 *pueda ser impactada en* $\mathtt{i}_2$*, debe ocurrir que* $\mathsf{T}_3.\mathtt{I} \subseteq \mathsf{T}_1.\mathtt{I} \wedge \mathsf{T}_2.\mathtt{I} \subseteq \mathsf{T}_1.\mathtt{I}$*. Esta proposición no se cumpliría, puesto que* $\mathsf{T}_3.\mathtt{I} = \{\mathtt{i}_2\} \nsubseteq \mathsf{T}_2.\mathtt{I}$*. Para poder llevar adelante esta operación, primero se debe propagar* T_2 *a* $\mathtt{i}_2$

$$\xrightarrow{\texttt{PROP}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]^{\{\mathtt{i}_1,\mathtt{i}_2\}}_{\mathtt{i}_1} \cdot [(x \leftarrow\!\shortmid -40)]^{\{\mathtt{i}_1\,\mathtt{i}_2\}}_{\mathtt{i}_1}$$

Así están dadas las condiciones para impactar T_3*, que bajo este contexto es una transacción vacía ya que la única escritura está atada a una condición que no se cumple. Es decir, al conocer los efectos de* T_2*, ahora la transacción* T_3 *evalúa la condición* $x+y = (-40)+60 = 20 <= 100$*, por lo tanto no ejecuta su potencial única escritura*

$$\xrightarrow{\texttt{COMM}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]^{\{\mathtt{i}_1,\mathtt{i}_2\}}_{\mathtt{i}_1} \cdot [(x \leftarrow\!\shortmid -40)]^{\{\mathtt{i}_1\,\mathtt{i}_2\}}_{\mathtt{i}_1} \cdot []^{\{\mathtt{i}_2\}}_{\mathtt{i}_2}$$

De esta manera, T_3 *no altera el sistema de manera anómala. La traza resultante es*

$$
\begin{aligned}
\epsilon &\xrightarrow{\texttt{COMM}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]_{\mathtt{i}_1}^{\{\mathtt{i}_1\}} \\
&\xrightarrow{\texttt{PROP}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]_{\mathtt{i}_1}^{\{\mathtt{i}_1,\mathtt{i}_2\}} \\
&\xrightarrow{\texttt{COMM}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]_{\mathtt{i}_1}^{\{\mathtt{i}_1,\mathtt{i}_2\}} \cdot [(x \leftarrow\!\shortmid -40)]_{\mathtt{i}_1}^{\{\mathtt{i}_1\}} \\
&\xrightarrow{\texttt{PROP}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]_{\mathtt{i}_1}^{\{\mathtt{i}_1,\mathtt{i}_2\}} \cdot [(x \leftarrow\!\shortmid -40)]_{\mathtt{i}_1}^{\{\mathtt{i}_1\,\mathtt{i}_2\}} \\
&\xrightarrow{\texttt{COMM}} [(x \leftarrow\!\shortmid 60) \cdot (y \leftarrow\!\shortmid 60)]_{\mathtt{i}_1}^{\{\mathtt{i}_1,\mathtt{i}_2\}} \cdot [(x \leftarrow\!\shortmid -40)]_{\mathtt{i}_1}^{\{\mathtt{i}_1,\mathtt{i}_2\}} \cdot []_{\mathtt{i}_2}^{\{\mathtt{i}_2\}}
\end{aligned}
$$

4. CORRECTITUD

4.1. Revisitando ejecución abstracta

Las ejecuciones abstractas a utilizar asocian sus eventos 1 a 1 con transacciones commiteadas en el sistema, previamente habido quitado la información relacionada con las réplicas:

$\delta \ ::= \ [\mathtt{u}_1 \cdot ... \cdot \mathtt{u}_n]$

El mapeo de la función $\mathtt{OP}$ es de eventos de $\mathcal{E}$ a operaciones δ. Escribimos $\alpha \approx \delta$ cuando el resultado de quitar a α la decoración de réplicas es δ. Es decir $\alpha = \mathsf{T}_\mathtt{i} \approx \mathsf{T} = \delta$.

Un cabo suelto por aclarar es $\mathtt{VAR}(\mathtt{e}) = \mathtt{OP}(\mathtt{e}).\mathtt{X}$. Es un reemplazo sintáctico que impacta únicamente la definición del axioma NoConflict bajo ejecución abstracta. De aquí en adelante, utilizamos la nueva definición.

4.2. Construcción de ejecuciones abstractas

La construcción de las ejecuciones abstractas transaccionales se hace a través del mecanismo que denominamos *extensión de una ejecución abstracta* $\mathcal{A} = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR} \rangle$ con un evento fresco $\mathtt{e}$, donde *fresco* implica $\mathtt{e} \notin \mathcal{E}$.

$$\mathcal{A} \overset{\mathtt{e} \triangleright \delta}{\oplus} \mathtt{E} = \ \langle \mathcal{E} \cup \{\mathtt{e}\}, \ \mathtt{OP}[\mathtt{e} \mapsto \delta], \ \mathtt{VIS} \cup (\mathtt{E} \times \{\mathtt{e}\}), \ \mathtt{AR} \cup (\mathtt{E} \times \{\mathtt{e}\}) \rangle$$

Amplía $\mathcal{E}$ con $\mathtt{e}$. La función $\mathtt{OP}$ contempla la nueva correspondencia de $\mathtt{e}$ con δ. El evento $\mathtt{e}$ está asociado a una transacción δ y la misma observa el conjunto de eventos *downward closed* $\mathtt{E} \subseteq \mathcal{E}$ respecto al orden total $\mathtt{AR}|_\mathtt{E}$. Por ende, tanto $\mathtt{VIS}$ como $\mathtt{AR}$ son unidos al conjunto $\mathtt{E} \times \mathtt{e}$. Si bien tanto arbitración como visibilidad se extienden de la misma manera, notar que ambos conjuntos no necesariamente son iguales, ya que la propagación cambia la arbitración pero no la visibilidad. El siguiente lema muestra que la operación $\_ \overset{\_ \triangleright \_}{\oplus} \_$ genera ejecuciones abstractas.

Lema 1. *Si $\mathcal{A}$ es una ejecución abstracta $\implies$ $\mathcal{A} \overset{\mathtt{e} \triangleright \delta}{\oplus} \mathtt{E}$ es una ejecución abstracta.*

Demostración. Probamos que el resultado cumple todas las propiedades de ejecución abstracta. De manera inductiva, sea $\mathcal{A} = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR} \rangle$ una ejecución abstracta y $\mathtt{E} \subseteq \mathcal{E}$ downward closed tal que $\mathtt{AR}|_\mathtt{E}$ es un orden total.
Tomo $\mathcal{A}' = \mathcal{A} \overset{\mathtt{e} \triangleright \delta}{\oplus} \mathtt{E} = \langle \mathcal{E}', \mathtt{OP}', \mathtt{VIS}', \mathtt{AR}' \rangle$. Debemos verificar todas las propiedades de Def. 1.

1. $\mathtt{OP}' = \mathtt{OP}[\mathtt{e} \mapsto \delta]$. Como $\mathtt{OP} : \mathcal{E} \mapsto \mathbb{T}$ y $\delta \in \mathbb{T} \implies \mathtt{OP}' : \mathcal{E}' \mapsto \mathbb{T}$.

2. $\mathtt{VIS}' = \mathtt{VIS} \cup (\mathtt{E} \times \{\mathtt{e}\}) = \mathtt{VIS} \cup \{(\mathtt{e}', \mathtt{e}) \mid \mathtt{e}' \in \mathcal{E}\}$. Sabemos que $\mathtt{VIS}$ es acíclica y la extensión $(\mathtt{E} \times \{\mathtt{e}\})$ no genera ciclos nuevos ya que todos los elementos son de la forma $(\_, \mathtt{e})$, y $\mathtt{e}$ es fresca. Por lo tanto, $\mathtt{VIS}'$ es acíclica.

3. $\mathtt{AR}' = \mathtt{AR} \cup (\mathtt{E} \times \{\mathtt{e}\})$. Por construcción, sabemos que $\mathtt{VIS} \subseteq \mathtt{AR}$ y $(\mathtt{E} \times \{\mathtt{e}\}) \subseteq (\mathtt{E} \times \{\mathtt{e}\})$. Por lo tanto, $\mathtt{VIS}' \subseteq \mathtt{AR}'$. Veamos que $\mathtt{AR}'$ es strict prefix order:

$$\frac{\sigma \circledS \mathsf{T}_{\mathtt{i}}^{\{\mathtt{i}\}} \qquad \mathtt{e}\ \mathit{fresca}}{\sigma \xrightarrow{\mathtt{e} \triangleright \mathsf{T}_{\mathtt{i}}} \sigma \cdot \mathtt{e} \triangleright \mathsf{T}_{\mathtt{i}}^{\{\mathtt{i}\}}}\ (\textsc{commit-e})$$

$$\frac{\sigma \equiv \sigma_0 \quad \sigma_0 \xrightarrow{\lambda} \sigma_0' \quad \sigma_0' \equiv \sigma'}{\sigma \xrightarrow{\lambda} \sigma'}\ (\textsc{str-e})$$

$$\frac{\mathsf{T}' := \mathsf{T}[\mathtt{I} \mapsto \mathsf{T}.\mathtt{I} \cup \{\mathtt{i}\}] \quad \dagger\,(\sigma_0, \mathsf{T}', \sigma_1) \quad \mathtt{arb}(\sigma_0 \cdot \mathtt{e} \triangleright \mathsf{T}' \cdot \sigma_1) \quad \sigma_1 \cdot \sigma_2 \equiv \sigma_2 \cdot \sigma_1 \quad \forall \_ \triangleright \mathsf{S} \in \sigma_2.\mathtt{i} \notin \mathsf{S}.\mathtt{I}}{\sigma_0 \cdot \mathtt{e} \triangleright \mathsf{T} \cdot \sigma_1 \cdot \sigma_2 \xrightarrow{\tau[\widetilde{\sigma_0} \cdot \mathtt{e} \cdot \widetilde{\sigma_1}]} \sigma_0 \cdot \mathtt{e} \triangleright \mathsf{T}' \cdot \sigma_1 \cdot \sigma_2}\ (\textsc{propagation-e})$$

Fig. 4.1: Semántica operacional con decoración de eventos

- Transitividad: $\mathtt{AR}$ es transitiva. Como $\mathtt{e}$ es fresca, las únicas tuplas en que son contenidas por $\mathtt{AR}'$ de manera que $\mathtt{e}$ aparece en alguna componente son $(\_, \mathtt{e})$. Por lo tanto, tomamos $\mathtt{a}, \mathtt{b} \in \mathcal{E} \land \mathtt{c} = \mathtt{e} : (\mathtt{a}, \mathtt{b}) \in \mathtt{AR}' \land (\mathtt{b}, \mathtt{c}) \in \mathtt{AR}'$. Por construcción, sabemos que $\mathtt{b} \in \mathtt{E}$ y que $\mathtt{E}$ es downward closed respecto de $\mathtt{AR}|_{\mathtt{E}}$. Formalmente:

  $$\forall \mathtt{e}' \in \mathtt{E},\ \mathtt{e}'' \in \mathcal{E} : (\mathtt{e}'', \mathtt{e}') \in \mathtt{AR} \implies \mathtt{e}'' \in \mathtt{E}$$

  Tomando $\mathtt{e}' = \mathtt{b}$ y $\mathtt{e}'' = \mathtt{a}$ obtenemos que $\mathtt{a} \in \mathtt{E} \implies (\mathtt{a}, \mathtt{c}) \in \mathtt{E} \times \{\mathtt{e}\} \subseteq \mathtt{AR}'$. Por ende, $\mathtt{AR}'$ es transitiva.

- Antisimetra: $\mathtt{AR}$ es antisimétrico y la extensión contiene únicamente tuplas $(\_, \mathtt{e})$, con $\mathtt{e}$ fresca. No puede presentarse el caso que dos tuplas distintas cumplan $(\mathtt{e}, \_) \in \mathtt{AR}' \land (\_, \mathtt{e}) \in \mathtt{AR}'$. Por lo tanto, $\mathtt{AR}'$ es antisimétrica.

- Downward Totality: $\mathtt{AR}$ es strict prefix order. Resta ver que:

  $$\forall \mathtt{e}', \mathtt{e}'' \in \mathtt{E}.\ (\mathtt{e}', \mathtt{e}) \in \mathtt{AR}' \land (\mathtt{e}'', \mathtt{e}) \in \mathtt{AR}' \implies (\mathtt{e}', \mathtt{e}'') \in \mathtt{AR}' \lor (\mathtt{e}'', \mathtt{e}') \in \mathtt{AR}'$$

  Esta propiedad se cumple ya que $\mathtt{AR}|_{\mathtt{E}}$ es un orden total y $\mathtt{AR}|_{\mathtt{E}} \subseteq \mathtt{AR} \subseteq \mathtt{AR}'$. Esto verifica que $\mathtt{AR}'$ es downward total.

□

Para poder enlazar una ejecución abstracta a un estado, hace falta un mecanismo que relacione un evento $\mathtt{e}$ con una transacción T. Para ello, introducimos:

$$\lambda \ ::= \ \mathtt{e} \triangleright \alpha \qquad\qquad \mu \ ::= \ \tau[\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_k] \mid \lambda$$

Una versión decorada de la semántica dada en Fig. 3.1. Este sistema de transición está dado por 2 tipos de labels: el primero asocia un evento a una transacción, y el segundo representa una sincronización que ordena totalmente los eventos $\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_k$.

La semántica operacional decorada está dada por la Fig. 4.1. La regla commit-e extiende commit con la asociación evento-operación mencionada previamente, tomando un evento $\mathtt{e}$ fresco. Por otro lado, se modifica propagation, obteniendo propagation-e, para no sólo agregar la decoración del evento e a la operación propagada sino también contemplar la arbitración que se produce al aplicar esta regla. Siendo $\widetilde{\sigma}$ la extracción de la secuencia de eventos que se encuentran en σ, la construcción $\tau[\widetilde{\sigma_0} \cdot \mathtt{e} \cdot \widetilde{\sigma_1}]$ indica el orden total de los eventos de $\widetilde{\sigma_0}$, seguidos por $\mathtt{e}$ y finalmente por $\widetilde{\sigma_1}$.

$$\frac{\mathtt{e}\ \textit{fresca} \qquad \sigma \xrightarrow{\mathtt{e} \triangleright \mathsf{T}_\mathtt{i}} \sigma'}{\langle \mathcal{A}, \sigma \rangle \ \rightarrow\ \langle \mathcal{A} \overset{\mathtt{e} \triangleright \mathsf{T}}{\oplus} \mathtt{E_i}, \sigma' \rangle}\ (\text{A-COMMIT})$$

$$\frac{\sigma \xrightarrow{\tau[\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_k]} \sigma' \qquad \mathtt{AR}' = \mathtt{AR} \cup [\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_k]}{\langle\langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR} \rangle, \sigma \rangle \ \rightarrow\ \langle\langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR}' \rangle, \sigma' \rangle}\ (\text{A-ARBITRATION})$$

Fig. 4.2: Reglas bajo Coherencia

4.3. Correspondencia entre estados y ejecuciones abstractas

Los estados de interés sobre los cuáles queremos operar son aquellos que se dicen *bien formados*, los que efectivamente pueden ser construidos partiendo de un sistema en blanco.

Definición 4 (**Estado bien formado**). *Un estado σ está bien formado si $\exists\ t$ tal que $\epsilon \overset{t}{\Longrightarrow} \sigma$. Escribimos $\mathtt{wf}(\sigma)$ cuando un σ cumple el predicado y caracterizamos con σ al conjunto de los estados bien formados.*

La noción de *coherencia* nos va a permitir asegurar cuando un estado bien formado puede corresponderse con una ejecución abstracta. En el futuro, utilizamos el conjunto $\mathtt{E_i} = \{\mathtt{e} \mid \mathtt{e} \triangleright \mathsf{T} \in \sigma \wedge \mathtt{i} \in \mathsf{T}.\mathtt{I}\}$ que simplemente apila el conjunto de eventos que son conocidos por la réplica $\mathtt{i}$.

Definición 5 (**Coherencia**). *Sean $\mathcal{A} = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR} \rangle$ una ejecución abstracta y σ un estado tal que $\mathtt{wf}(\sigma)$. Decimos que $\mathcal{A}$ es coherente respecto a σ, notado como $\langle \mathcal{A}, \sigma \rangle$, si cumple la conjunción de:*

1. $|\sigma| = |\mathcal{E}|$
2. $\mathtt{e} \triangleright \mathsf{T} \in \sigma \iff \mathtt{e} \in \mathcal{E}$
3. $(\mathtt{e}', \mathtt{e}) \in \mathtt{AR} \iff \forall\ \sigma' \equiv \sigma, \sigma' = \sigma_0 \cdot (\mathtt{e}' \triangleright \mathsf{T}') \cdot \sigma_1 \cdot (\mathtt{e} \triangleright \mathsf{T}) \cdot \sigma_2$
4. $\forall\ \mathtt{i}, \mathtt{AR}|_{\mathtt{E_i}}$ *es un orden total*

Estas condiciones garantizan: que se correspondan 1 a 1 las transacciones en el estado a las operaciones en la ejecución abstracta; que dos operaciones $\mathtt{e}$ y $\mathtt{e}'$ estén relacionadas en $\mathtt{AR}$ si y sólo si las mismas aparecen arbitradas en σ; finalmente que las operaciones en cada réplica determinen un orden total.

Ahora que podemos asociarlos, detallamos cómo ocurren las transformaciones de ambos a la par.

Lema 2. *Sean $\mathcal{A} = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR} \rangle$ una ejecución abstracta y σ que verifica $\mathtt{wf}(\sigma)$, tal que $\langle \mathcal{A}, \sigma \rangle$. Entonces:*

1. *Si $\sigma \xrightarrow{\mathtt{e} \triangleright \alpha} \sigma'$ entonces $\langle \mathcal{A} \overset{\mathtt{e} \triangleright \delta}{\oplus} \mathtt{E_i}, \sigma' \rangle$, con $\alpha \approx \delta$ y $\alpha.\mathtt{i} = \mathtt{i}$.*
2. *Si $\sigma \xrightarrow{\tau[\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_k]} \sigma'$ entonces $\langle \mathcal{A}', \sigma' \rangle$ donde $\mathcal{A}' = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR}' \rangle$ y $\mathtt{AR}' = \mathtt{AR} \cup [\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_k]$.*

La primer premisa asegura que al commitear una nueva transacción a σ, la extensión de $\mathcal{A}$ es consistente a la misma. La segunda indica que al arbitrarse los eventos $[\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_k]$, el conjunto $\mathtt{AR}$ contempla ese cambio. Estas premisas arrojan como resultado el sistema de transiciones de Fig. 4.2. Como mencionamos anteriormente, se puede ver que A-ARBITRATION es una transición en la cuál únicamente se extiende el conjunto $\mathtt{AR}$, por lo tanto $\mathtt{VIS} \subseteq \mathtt{AR}$.

Demostración. Realizamos un análisis inductivo, asumiendo que se cumple para $\mathcal{A}$, con separación por casos en la regla aplicada.

Ítem 1. La única reducción que aplica es (A-COMMIT). De esta manera, $\sigma \xrightarrow{\mathtt{e} \triangleright \alpha} \sigma'$ y $\sigma' = \sigma \cdot \mathtt{e} \triangleright \mathsf{T}_{\mathtt{i}}^{\{\mathtt{i}\}}$. Por definición de estado bien formado, $\mathtt{wf}(\sigma')$ se cumple.
Además, $\mathcal{A}' = \mathcal{A} \overset{\mathtt{e} \triangleright \mathsf{T}_{\mathtt{i}}}{\oplus} \mathtt{E}_{\mathtt{i}}$ es una ejecución abstracta por Lem. 1. Debemos comprobar que las propiedades de Def. 5 se verifican:

1.

$$\begin{aligned}
|\sigma'| &= |\sigma \cdot \mathtt{e} \triangleright \mathsf{T}_{\mathtt{i}}^{\{\mathtt{i}\}}| && \text{Por definición}\\
&= |\sigma| + |\mathtt{e} \triangleright \mathsf{T}_{\mathtt{i}}^{\{\mathtt{i}\}}| && \text{Por propiedades de } |_|\\
&= |\mathcal{E}| + |\mathtt{e} \triangleright \mathsf{T}_{\mathtt{i}}^{\{\mathtt{i}\}}| && \text{Por Def. 5}\\
&= |\mathcal{E}| + |\{\mathtt{e}\}| && \text{Por Def. 5}\\
&= |\mathcal{E} \cup \{\mathtt{e}\}| && \text{Por propiedades de } |_|\\
&= |\mathcal{E}'|
\end{aligned}$$

2. $\mathtt{e}' \triangleright \mathsf{T}_{\mathtt{i}}^{\{\mathtt{i}\}} \in \sigma' \iff \mathtt{e}' \in \mathcal{E}' \iff \mathtt{e}' \triangleright \mathsf{T}_{\mathtt{i}}^{\{\mathtt{i}\}} \in \sigma \vee \mathtt{e}' = \mathtt{e}$

   - El caso $\mathtt{e}' \triangleright \mathsf{T}_{\mathtt{i}}^{\{\mathtt{i}\}} \in \sigma$ se verifica ya que $\langle \mathcal{A}, \sigma \rangle$.
   - El caso $\mathtt{e}' = \mathtt{e}$ se verifica debido a que $\mathtt{e}$ es el evento que es agregado a la ejecución abstracta mediante $\_ \overset{\_ \triangleright \_}{\oplus} \_$.

3. Sabemos que la propiedad se sostiene para $(\mathtt{a}, \mathtt{b}) \in \mathtt{AR}$. Dado que $\mathtt{AR}' = \mathtt{AR} \cup (\mathtt{E}_{\mathtt{i}} \times \{\mathtt{e}\})$, resta ver que se cumpla para $(\mathtt{e}', \mathtt{e}) \in (\mathtt{E}_{\mathtt{i}} \times \{\mathtt{e}\})$. Es decir:

   $(\mathtt{e}', \mathtt{e}) \in \mathtt{AR}' \iff \forall\, \sigma'' \equiv \sigma', \sigma'' = \sigma_0 \cdot (\mathtt{e}' \triangleright \mathsf{T}') \cdot \sigma_1 \cdot (\mathtt{e} \triangleright \mathsf{T}) \cdot \sigma_2$

   Sabemos que $\mathtt{e}' \in \mathtt{E}_{\mathtt{i}} \subseteq \mathcal{E} \iff (\mathtt{e}' \triangleright \mathsf{T}') \in \sigma$, por Def. 5.2. Por otro lado, ocurre que $\{\mathtt{i}\} \subseteq \mathsf{T}'.\mathtt{I} \cap \mathsf{T}.\mathtt{I} \neq \emptyset \iff$ Por Def. 2, no se pueden swappear.

4. Debemos mostrar que $\forall\, \mathtt{j}.\mathtt{AR}|_{\mathtt{E}_{\mathtt{j}}}$ es un total order, con $\mathtt{E}_{\mathtt{j}} = \{\mathtt{e} \mid \mathtt{e} \triangleright \mathsf{T} \in \sigma \wedge \mathtt{j} \in \mathsf{T}.\mathtt{I}\}$.

   - $\mathtt{j} \neq \mathtt{i} \implies \mathtt{AR}'|_{\mathtt{E}_{\mathtt{j}}} = \mathtt{AR}|_{\mathtt{E}_{\mathtt{j}}}$. Por lo tanto, se cumple.
   - $\mathtt{j} = \mathtt{i}$. Notamos que $\mathtt{AR}'|_{\mathtt{E}_{\mathtt{i}}} = \mathtt{AR}|_{\mathtt{E}_{\mathtt{i}}} \cup (\mathtt{E}_{\mathtt{i}} \times \{\mathtt{e}\})$. Se puede observar fácilmente que se cumplen las propiedades de totalidad sobre $\mathtt{AR}'|_{\mathtt{E}_{\mathtt{i}}}$ considerando que $\mathtt{AR}|_{\mathtt{E}_{\mathtt{i}}}$ es total y es extendido con el conjunto $\mathtt{E}_{\mathtt{i}} \times \{\mathtt{e}\}$, que tiene a $\mathtt{e}$ como elemento maximal.

Ítem 2. De manera análoga, observamos que (A-PROPAGATION) es la única reducción que aplica. Sean $\sigma = \sigma_0 \cdot \mathtt{e} \triangleright \mathsf{T} \cdot \sigma_1 \cdot \sigma_2$ y $\mathsf{T}' = \mathsf{T}[\mathtt{I} \mapsto \mathsf{T}.\mathtt{I} \cup \{\mathtt{i}\}]$, tenemos que $\sigma \xrightarrow{\tau[\widetilde{\sigma_0} \cdot \mathtt{e} \cdot \widetilde{\sigma_1}]} \sigma'$ con $\sigma' = \sigma_0 \cdot \mathtt{e} \triangleright \mathsf{T}' \cdot \sigma_1 \cdot \sigma_2$. Además,

(I) $\mathtt{arb}(\sigma_0 \cdot \mathtt{e} \triangleright \mathsf{T}' \cdot \sigma_1)$

(II) $\sigma_1 \cdot \sigma_2 \equiv \sigma_2 \cdot \sigma_1$

(III) $\forall \_ \triangleright \mathsf{S} \in \sigma_2.\mathtt{i} \notin \mathsf{S}.\mathtt{I}$

Nuevamente, debemos comprobar que las propiedades de Def. 5 se verifican:

1. Dado que $\mathtt{OP} = \mathtt{OP}'$, se comprueba trivialmente.
2. Dado que $\mathtt{E} = \mathtt{E}'$, se comprueba trivialmente.
3. Sabemos que se cumple para $\mathtt{AR}$. Dado que $\mathtt{AR}' = \mathtt{AR} \cup [\widetilde{\sigma_0} \cdot \mathtt{e} \cdot \widetilde{\sigma_1}]$, resta ver la conjunción de:

   $(\mathtt{e}_0, \mathtt{e}) \in [\widetilde{\sigma_0} \cdot \mathtt{e} \cdot \widetilde{\sigma_1}] \iff \forall\, \sigma'' \equiv \sigma', \sigma'' = \sigma''_0 \cdot (\mathtt{e}_0 \triangleright \mathsf{T}_0) \cdot \sigma''_1 \cdot (\mathtt{e} \triangleright \mathsf{T}') \cdot \sigma''_2$
   $(\mathtt{e}, \mathtt{e}_1) \in [\widetilde{\sigma_0} \cdot \mathtt{e} \cdot \widetilde{\sigma_1}] \iff \forall\, \sigma''' \equiv \sigma', \sigma''' = \sigma'''_0 \cdot (\mathtt{e} \triangleright \mathsf{T}') \cdot \sigma'''_1 \cdot (\mathtt{e}_1 \triangleright \mathsf{T}_1) \cdot \sigma'''_2$

   $\Rightarrow$) de cada sentencia se verifica teniendo en cuenta $\mathtt{arb}(\sigma_0 \cdot \mathtt{e} \triangleright \mathsf{T}' \cdot \sigma_1)$ y Def. 3. Como $\mathtt{e}_0 \in \widetilde{\sigma_0}$ y $\mathtt{e}_1 \in \widetilde{\sigma_1}$, ese es el único orden posible en el que pueden aparecer en σ'.

   $\Leftarrow$) comenzamos analizando el primer *sólo si*. El antecedente, en conjunción con Def. 3, implica $\mathtt{arb}(\sigma''_0 \cdot (\mathtt{e}_0 \triangleright \mathsf{T}_0) \cdot \sigma''_1 \cdot (\mathtt{e} \triangleright \mathsf{T}') \cdot \sigma''_2)$. Por otra parte, habíamos asumido (I). En consecuencia, tomando $\sigma_0 = \sigma''_0 \cdot \mathtt{e}_0 \triangleright \mathsf{T}_0 \cdot \sigma''_1$ tenemos que $\mathtt{e}_0 \in \widetilde{\sigma_0}$, lo que significa $(\mathtt{e}_0, \mathtt{e}) \in [\widetilde{\sigma_0} \cdot \mathtt{e} \cdot \widetilde{\sigma_1}]$. El segundo *sólo si* se corrobora de manera análoga, observando que $\sigma_1 = \sigma'''_1 \cdot \mathtt{e}_1 \triangleright \mathsf{T}_1 \cdot \sigma'''_2$.
4. Debemos mostrar que $\forall\, \mathtt{j}.\mathtt{AR}|_{\mathtt{E_j}}$ es un total order, con $\mathtt{E_j} = \{\mathtt{e} \mid \mathtt{e} \triangleright \mathsf{T} \in \sigma \wedge \mathtt{j} \in \mathsf{T}.\mathtt{I}\}$.
   - $\mathtt{j} \neq \mathtt{i} \implies \mathtt{AR}'|_{\mathtt{E_j}} = \mathtt{AR}|_{\mathtt{E_j}}$. Por lo tanto, se cumple.
   - $\mathtt{j} = \mathtt{i}$. Llamo $\mathsf{T}' = \mathsf{T}[\mathtt{I} \mapsto \mathsf{T}.\mathtt{I} \cup \{\mathtt{i}\}]$. Tenemos que $\mathtt{E}'_\mathtt{i} = \mathtt{E_i} \cup \{\mathtt{e}\}$ dado que $\mathtt{i} \in \mathsf{T}'.\mathtt{I}$ y $\mathtt{e} \triangleright \mathsf{T}' \in \sigma'$, esto último por Def. 5.2. Adicionalmente, por (I) y (III), inferimos que $\forall \mathtt{e}' \in \mathtt{E_i}.\ \mathtt{e}' \in [\widetilde{\sigma_0} \cdot \mathtt{e} \cdot \widetilde{\sigma_1}]$. Por lo tanto, como $[\widetilde{\sigma_0} \cdot \mathtt{e} \cdot \widetilde{\sigma_1}]$ está totalmente ordenado, $\mathtt{AR}'|_{\mathtt{E_i}}$ es total.

□

4.4. Correctitud y completitud de la caracterización operacional

Mostraremos las pruebas de que la caracterización operacional propuesta es correcta y completa con respecto a las ejecuciones abstractas. En lo escrito a continuación, decoramos las reducciones con las siglas de los modelos de consistencia para referirnos a que dichas transiciones respetaron dicho modelo. Siendo $\mathtt{C}$ un modelo, escribimos $\xrightarrow{\beta}_{\mathtt{C}}$ o $\overset{t}{\Longrightarrow}_{\mathtt{C}}$ para indicar 1 o más reducciones respetando $\mathtt{C}$.

Lema 3 (Correctitud $\mathtt{TV}$)**.** *Si* $\langle \emptyset, \epsilon \rangle \Rightarrow_{\mathtt{TV}} \langle \mathcal{A}, \sigma \rangle$ *entonces* $\mathcal{A} = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR} \rangle$ *y* $\mathtt{VIS}$ *es una relación transitiva.*

Demostración. Inducción en el largo $n \geq 0$ de la derivación $\Rightarrow_{\mathtt{TV}}$

- $\mathbf{n = 0}$. $\mathcal{A} = \emptyset \implies \mathtt{VIS} = \emptyset \implies \mathtt{VIS}$ transitiva trivialmente.

- $\mathbf{n = k + 1}$. Asumimos que $\langle \emptyset, \epsilon \rangle \Rightarrow_{\texttt{TV}} \langle \mathcal{A}, \sigma \rangle$ tal que $\mathcal{A} = \langle \mathcal{E}, \texttt{OP}, \texttt{VIS}, \texttt{AR} \rangle$ y $\texttt{VIS}$ es una relación transitiva, en $\mathbf{k}$ pasos. Derivo en 1 paso $\langle \mathcal{A}, \sigma \rangle \rightarrow_{\texttt{TV}} \langle \mathcal{A}', \sigma' \rangle$, por lo tanto $\langle \emptyset, \epsilon \rangle \Rightarrow_{\texttt{TV}} \langle \mathcal{A}', \sigma' \rangle$ en $\mathbf{k + 1}$ pasos. Sea $\mathcal{A}' = \langle \mathcal{E}', \texttt{OP}', \texttt{VIS}', \texttt{AR}' \rangle$.

  La prueba se sigue por análisis de casos sobre la última computación.

  - (A-COMMIT). $\langle \mathcal{A}, \sigma \rangle \rightarrow_{\texttt{TV}} \langle \mathcal{A} \overset{\texttt{e}' \triangleright \texttt{T}'}{\oplus} \texttt{E}_\texttt{i}, \sigma' \rangle$ entonces $\sigma \xrightarrow{\texttt{e}' \triangleright \texttt{T}'_\texttt{i}} \sigma'$. La única regla que aplica es (COMMIT-E). Para que $\texttt{VIS}'$ sea transitiva debe ocurrir que:

    $$(\texttt{a}, \texttt{b}) \in \texttt{VIS}' \wedge (\texttt{b}, \texttt{c}) \in \texttt{VIS}' \implies (\texttt{a}, \texttt{c}) \in \texttt{VIS}'$$

    Recordamos que $\texttt{VIS}' = \texttt{VIS} \cup (\texttt{E}_\texttt{i} \times \{\texttt{e}'\})$. Inmediatamente notamos que no existen tuplas $(\texttt{e}', \_)$ en $\texttt{VIS}'$, por ende no puede ocurrir que $\texttt{a} = \texttt{e}' \vee \texttt{b} = \texttt{e}' \implies \{\texttt{a}, \texttt{b}\} \in \mathcal{E}$. Por otra parte, $\texttt{c} \neq \texttt{e}' \implies \texttt{c} \in \mathcal{E} \implies (\texttt{a}, \texttt{c}) \in \texttt{VIS}$ ya que $\texttt{VIS}$ es transitiva por hipótesis inductiva. Analizamos el caso $\texttt{c} = \texttt{e}'$.

    Ya que las únicas tuplas $(\_, \texttt{e}') \in \texttt{VIS}'$ son las que pertenecen a $(\texttt{E}_\texttt{i} \times \{\texttt{e}'\})$, ocurre que $\texttt{b} \in \texttt{E}_\texttt{i}$. Bajo el mismo razonamiento, debo ver que $\texttt{a} \in \texttt{E}_\texttt{i}$.

    Sea $\texttt{j} = \texttt{OP}(\texttt{b}).\texttt{i}$. Sabemos que $\texttt{a} \in \{\texttt{a}' \mid (\texttt{a}', \texttt{b}) \in \texttt{VIS}' \wedge \texttt{j} \in \texttt{OP}(\texttt{a}').\texttt{I}\} = \texttt{E}_\texttt{j} \subseteq \mathcal{E}$. En consecuencia, $\texttt{j} \in \texttt{OP}(\texttt{a}).\texttt{I} \overset{\texttt{TV}}{\Longrightarrow} \texttt{OP}(\texttt{b}).\texttt{I} \subseteq \texttt{OP}(\texttt{a}).\texttt{I}$. Además, $\texttt{i} \in \texttt{OP}(\texttt{b}).\texttt{I} \overset{\texttt{TV}}{\Longrightarrow} \texttt{OP}(\texttt{c}).\texttt{I} \subseteq \texttt{OP}(\texttt{a}).\texttt{I}$.

    En resumen, $\texttt{i} \in \texttt{OP}(\texttt{c}).\texttt{I} \subseteq \texttt{OP}(\texttt{b}).\texttt{I} \subseteq \texttt{OP}(\texttt{a}).\texttt{I} \implies \texttt{a} \in \texttt{E}_\texttt{i}$. Por lo tanto, $(\texttt{a}, \texttt{c}) \in \texttt{VIS}'$.
  - (A-ARBITRATION). Por definición, $\texttt{VIS} = \texttt{VIS}'$. Por hipótesis inductiva, $\texttt{VIS}'$ es transitiva.

□

Lema 4 (Completitud TV)**.** *Sea* $\mathcal{A} = \langle \mathcal{E}, \texttt{OP}, \texttt{VIS}, \texttt{AR} \rangle$ *una ejecución abstracta, tal que* $\texttt{VIS}$ *es transitiva. Entonces:*

$$\exists \sigma, \mathcal{A}' = \langle \mathcal{E}, \texttt{OP}, \texttt{VIS}, \texttt{AR}' \rangle \text{ tal que } \langle \emptyset, \epsilon \rangle \Rightarrow_{\texttt{TV}} \langle \mathcal{A}', \sigma \rangle,\ \texttt{AR} \subseteq \texttt{AR}' \text{ y } \texttt{VIS} \text{ es transitiva}$$

Demostración. La prueba se realiza por inducción en el tamaño de $\mathcal{E}$, es decir, $|\mathcal{E}| = n$.

- $\mathbf{n = 0}$. Este caso se mantiene trivialmente, basta tomar $\sigma = \epsilon$. Por lo tanto $\mathcal{E} = \emptyset$, $\texttt{AR} = \emptyset$ y $\texttt{VIS} = \emptyset$.

- $\mathbf{n = k + 1}$. Sea $\texttt{e}$ un elemento maximal en $\texttt{AR}$. Luego, definimos $\mathcal{E}' = \mathcal{E} \setminus \{\texttt{e}\}$ y $\mathcal{A}|_{\mathcal{E}'} = \langle \mathcal{E}', \texttt{OP}|_{\mathcal{E}'}, \texttt{VIS}|_{\mathcal{E}'}, \texttt{AR}|_{\mathcal{E}'} \rangle$. Chequear que $\mathcal{A}|_{\mathcal{E}'}$ es una ejecución abstracta donde $\texttt{VIS}|_{\mathcal{E}'}$ es transitiva es inmediato por HI. Como $\texttt{e}$ es maximal, podemos definir

  $$\texttt{AR} = \texttt{AR}|_{\mathcal{E}'} \cup (\mathcal{E}^l \times \{\texttt{e}\}) \text{ con } \mathcal{E}^l \subseteq \mathcal{E}'$$

  Por hipótesis inductiva sobre $\texttt{AR}|_{\mathcal{E}'}$, sabemos $\exists \sigma', \texttt{AR}''$ tal que $\langle \emptyset, \epsilon \rangle \Rightarrow_{\texttt{TV}} \langle \mathcal{A}'', \sigma' \rangle$, con $\mathcal{A}'' = \langle \mathcal{E}', \texttt{OP}|_{\mathcal{E}'}, \texttt{VIS}|_{\mathcal{E}'}, \texttt{AR}'' \rangle$, $\texttt{VIS}|_{\mathcal{E}'}$ transitiva y $\texttt{AR}|_{\mathcal{E}'} \subseteq \texttt{AR}''$. La prueba se lleva adelante mostrando que es posible realizar la siguiente computación:

  $$\langle \mathcal{A}'', \sigma' \rangle \Rightarrow_{\texttt{TV}} \langle \mathcal{A}', \sigma \rangle$$

Sea $\mathtt{i}$ una réplica fresca. Consideremos $\mathtt{OP}(\mathtt{e}) = [\mathtt{u}_1 \cdot ... \cdot \mathtt{u}_n] = \delta \approx \alpha = [\mathtt{u}_1 \cdot ... \cdot \mathtt{u}_n]_\mathtt{i}$. La estrategia para construir la computación es (i) propagar cada transacción relacionada a $\mathtt{e}$ en $\mathtt{AR}$ a la réplica $\mathtt{i}$ (ii) hacer el commit de $\mathtt{e}$ también en la réplica $\mathtt{i}$.

Paso (i). Proponemos el siguiente mecanismo para propagar las transacciones de $\mathcal{E}^l$ a la réplica $\mathtt{i}$, que se tomó fresca.

Sea $\sigma^l = [\mathtt{e}_1^l \triangleright \mathsf{T}_1^l, \ldots, \mathtt{e}_m^l \triangleright \mathsf{T}_m^l]$ la subsecuencia ordenada tal que $\mathtt{e}_j^l \in \mathcal{E}^l$. La misma representa todas las transacciones en σ relacionadas a $\mathtt{e}$ en $\mathtt{AR}$. A continuación, aplicamos la regla (A-ARBITRATION) m veces para propagar cada T_j^l a la réplica $\mathtt{i}$.

En la iteración 1 tomamos:

$$\begin{array}{lcl} \sigma_0 & = & \epsilon \\ \sigma_1 & = & \epsilon \\ \sigma_2 & \text{t.q.} & \sigma'' = \sigma_0 \cdot (\mathtt{e}_1^l \triangleright \mathsf{T}_1^l[\mathtt{I} \mapsto \mathsf{T}_1^l.\mathtt{I} \cup \{\mathtt{i}\}]) \cdot \sigma_1 \cdot \sigma_2 \end{array}$$

Entonces $\sigma'' \xrightarrow{\tau[\widetilde{\sigma_0} \cdot \mathtt{e}_1^l \cdot \widetilde{\sigma_1}]} \sigma_1''$ con $\mathtt{AR}_1'' = \mathtt{AR}''$.

Siendo $1 < j \leq 1$, en la iteración j tomamos:

$$\begin{array}{lcl} \sigma_0 & = & [\mathtt{e}_1^l \triangleright \mathsf{T}_1^l, \ldots, \mathtt{e}_{j-1}^l \triangleright \mathsf{T}_{j-1}^l] \\ \sigma_1 & = & \epsilon \\ \sigma_2 & \text{t.q.} & \sigma'' = \sigma_0 \cdot (\mathtt{e}_j^l \triangleright \mathsf{T}_j^l[\mathtt{I} \mapsto \mathsf{T}_j^l.\mathtt{I} \cup \{\mathtt{i}\}]) \cdot \sigma_1 \cdot \sigma_2 \end{array}$$

Entonces $\sigma_{j-1}'' \xrightarrow{\tau[\widetilde{\sigma_0} \cdot \mathtt{e}_j^l \cdot \widetilde{\sigma_1}]} \sigma_j''$ con $\mathtt{AR}_j'' = \mathtt{AR}_{j-1}'' \cup [\widetilde{\sigma_0} \cdot \mathtt{e}_j^l \cdot \widetilde{\sigma_1}]$.

Notar que $\mathtt{VIS}$ no fue alterado, por definición de (A-ARBITRATION). Por lo tanto, en cada iteración j se cumple $\dagger(\sigma_0, \mathsf{T}_j^l, \sigma_1) = \sigma_0 \circledcirc \mathsf{T}_j^l$ con $j \in \{1, \ldots, m\}$. El modelo de consistencia se sostiene en todas las iteraciones. Como resultado parcial, tenemos

$$\langle \mathcal{A}''', \sigma'' \rangle \Rightarrow_{\mathtt{TV}} \langle \mathcal{A}_m''', \sigma_m'' \rangle, \text{ con } \mathcal{A}_m''' = \langle \mathcal{E}', \mathtt{OP}|_{\mathcal{E}'}, \mathtt{VIS}|_{\mathcal{E}'}, \mathtt{AR}_m'' \rangle$$

Paso (ii). Buscamos realizar una última computación $\langle \mathcal{A}_m''', \sigma_m'' \rangle \rightarrow_{\mathtt{TV}} \langle \mathcal{A}', \sigma \rangle$ utilizando (A-COMMIT), con la transacción correspondiente a $\mathtt{e}$ siendo la commiteada. Para ello, primero notamos:

- $\mathtt{e}$ es fresca. $\mathtt{e} \notin \mathtt{E}' \implies$ Por Def. 5.2, $\mathtt{e} \triangleright \_ \notin \sigma_m''$.
- $\sigma_m'' \xrightarrow{\mathtt{e} \triangleright \mathsf{T}_\mathtt{i}} \sigma$. Debemos ver que se cumple $\sigma_m'' \circledcirc \mathsf{T}_\mathtt{i}^\mathtt{i}$, con $\mathsf{T} = \delta$.

$$\begin{array}{rcll} & & \forall \mathsf{S} \in \sigma_m''.\ \mathsf{T}.\mathtt{i} \in \mathsf{S}.\mathtt{I} \implies \mathsf{T}.\mathtt{I} \subseteq \mathsf{S}.\mathtt{I} & \iff \text{por (i)} \\ \forall j \in \{1, \ldots, m\}.\ \mathsf{T}.\mathtt{i} \in \mathsf{T}_j^l.\mathtt{I} & \implies & \mathsf{T}.\mathtt{I} \subseteq \mathsf{T}_j^l.\mathtt{I} & \end{array}$$

  Sabemos, por (i) y por definición de T, que:

$$\mathsf{T}.\mathtt{i} = \mathtt{i} \in \mathsf{T}_j^l.\mathtt{I} \wedge \mathsf{T}.\mathtt{I} = \{\mathtt{i}\} \subseteq \mathsf{T}_j^l.\mathtt{I}$$

  Por lo tanto, se cumple la condición.

Por definición de extensión de ejecución abstracta, obtenemos:

$$\mathcal{A}' = \mathcal{A}''_m \overset{\mathtt{e} \triangleright \delta}{\oplus} \mathtt{E_i} = \langle \quad \mathcal{E}' \cup \{\mathtt{e}\}, \; \mathtt{OP}|_{\mathcal{E}'}[\mathtt{e} \mapsto \delta], \; \mathtt{VIS}|_{\mathcal{E}'} \cup (\mathtt{E_i} \times \{\mathtt{e}\}), \; \mathtt{AR}''_m \cup (\mathtt{E_i} \times \{\mathtt{e}\}) \quad \rangle$$

Siendo $\mathtt{E_i} = \{\, \mathtt{e}' \mid \mathtt{e}' \triangleright \mathtt{T}' \in \sigma''_m \wedge \mathtt{i} \in \mathtt{T}'.\mathtt{I} \,\}$. En conclusión, ocurre:

- $\mathcal{E}' \cup \{\mathtt{e}\} = \mathcal{E}$
- $\mathtt{OP}|_{\mathcal{E}'}[\mathtt{e} \mapsto \delta] = \mathtt{OP}$
- $\mathtt{VIS}|_{\mathcal{E}'} \cup (\mathtt{E_i} \times \{\mathtt{e}\}) = \mathtt{VIS}$
- $\mathtt{AR} \subseteq \mathtt{AR}'$ debido a:

$$\begin{array}{rcll} \mathtt{AR} & = & \mathtt{AR}|_{\mathcal{E}'} \cup (\mathcal{E}^l \times \{\mathtt{e}\}) & \\ & \subseteq & \mathtt{AR}'' \cup (\mathcal{E}^l \times \{\mathtt{e}\}) & \text{Por HI} \\ & \subseteq & \mathtt{AR}''_1 \cup (\mathcal{E}^l \times \{\mathtt{e}\}) & \text{Por (A-ARBITRATION)} \\ & \vdots & & \\ & \subseteq & \mathtt{AR}''_m \cup (\mathcal{E}^l \times \{\mathtt{e}\}) & \text{Por (A-ARBITRATION)} \\ & = & \mathtt{AR}' & \end{array}$$

□

Lema 5 (Correctitud NC)**.** *Si* $\langle \emptyset, \epsilon \rangle \Rightarrow_{\mathtt{NC}} \langle \mathcal{A}, \sigma \rangle$ *entonces*

$$\forall \mathtt{e}, \mathtt{e}' \in \mathcal{E}.\, \mathtt{e} \neq \mathtt{e}' \wedge \mathtt{OP}(\mathtt{e}).\mathtt{X} \cap \mathtt{OP}(\mathtt{e}').\mathtt{X} \neq \emptyset \implies (\mathtt{e}, \mathtt{e}') \in \mathtt{VIS} \vee (\mathtt{e}', \mathtt{e}) \in \mathtt{VIS}$$

Demostración. Inducción en el largo $n \geq 0$ de la derivación $\Rightarrow_{\mathtt{NC}}$

- $\mathbf{n = 0}$. Se deduce trivialmente a partir de que $\mathcal{E} = \emptyset$.
- $\mathbf{n = k + 1}$. Asumimos que $\langle \emptyset, \epsilon \rangle \Rightarrow_{\mathtt{NC}} \langle \mathcal{A}, \sigma \rangle$ tal que $\mathcal{A} = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR} \rangle$ y se cumple la propiedad enunciada, en $\mathbf{k}$ pasos. Derivo en 1 paso $\langle \mathcal{A}, \sigma \rangle \longrightarrow_{\mathtt{NC}} \langle \mathcal{A}', \sigma' \rangle$, por lo tanto $\langle \emptyset, \epsilon \rangle \Rightarrow_{\mathtt{NC}} \langle \mathcal{A}', \sigma' \rangle$ en $\mathbf{k + 1}$ pasos. Sea $\mathcal{A}' = \langle \mathcal{E}', \mathtt{OP}', \mathtt{VIS}', \mathtt{AR}' \rangle$.

  La prueba se sigue por análisis de casos sobre la última computación.

  - (A-COMMIT). $\langle \mathcal{A}, \sigma \rangle \longrightarrow_{\mathtt{PP}} \langle \mathcal{A} \overset{\mathtt{e}' \triangleright \mathtt{T}'}{\oplus} \mathtt{E_i}, \sigma' \rangle$ entonces $\sigma \xrightarrow{\mathtt{e}' \triangleright \mathtt{T}'_\mathtt{i}} \sigma'$. La única regla que aplica es (COMMIT-E). Por hipótesis inductiva, sabemos que para todo par de eventos en $\mathcal{E}$ la propiedad se cumple. Debemos verificar que se sostiene en relación al evento fresco $\mathtt{e}'$. Es decir:

    $$\forall \mathtt{e} \in \mathcal{E}.\, \mathtt{OP}(\mathtt{e}').\mathtt{X} \cap \mathtt{OP}(\mathtt{e}).\mathtt{X} \neq \emptyset \implies (\mathtt{e}', \mathtt{e}) \in \mathtt{VIS}' \vee (\mathtt{e}, \mathtt{e}') \in \mathtt{VIS}'$$

    Lo que, por Def. 5.2, es equivalente a:

    $$\forall \mathtt{e} \triangleright \mathtt{T} \in \sigma.\, \mathtt{T}'.\mathtt{X} \cap \mathtt{T}.\mathtt{X} \neq \emptyset \implies (\mathtt{e}', \mathtt{e}) \in \mathtt{VIS}' \vee (\mathtt{e}, \mathtt{e}') \in \mathtt{VIS}'$$

Por (COMMIT-E), sabemos que $\sigma \circledcirc \mathsf{T}_\mathtt{i}^{\{\mathtt{i}\}} \overset{\mathtt{NC}}{\Longrightarrow} (\mathsf{T}'.\mathtt{X} \cap \mathsf{T}.\mathtt{X} \implies \mathsf{T}'.\mathtt{i} \subseteq \mathsf{T}.\mathtt{I})$, teniendo en cuenta que T' es la transacción siendo commiteada.

$\mathsf{T}'.\mathtt{i} \subseteq \mathsf{T}.\mathtt{I} \implies \mathsf{T} \in \mathtt{E}_\mathtt{i} \implies (\mathtt{e}, \mathtt{e}') \in (\mathtt{E}_\mathtt{i} \times \{\mathtt{e}'\}) \subseteq \mathtt{VIS} \cup (\mathtt{E}_\mathtt{i} \times \{\mathtt{e}'\}) = \mathtt{VIS}'$

- (A-ARBITRATION). Se desprende trivialmente de la hipótesis inductiva y los hechos $\mathcal{E}' = \mathcal{E}$ y $\mathtt{VIS}' = \mathtt{VIS}$.

□

Lema 6 (Completitud NC)**.** *Sea* $\mathcal{A} = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR} \rangle$ *una ejecución abstracta que verifica el predicado* NC*. Entonces:*

$$\exists \sigma, \mathcal{A}' = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR}' \rangle \text{ tal que } \langle \emptyset, \epsilon \rangle \Rightarrow_{\mathtt{NC}} \langle \mathcal{A}', \sigma \rangle,\ \mathtt{AR} \subseteq \mathtt{AR}' \text{ y}$$

$$\forall \mathtt{e}, \mathtt{e}' \in \mathcal{E}.\, \mathtt{e} \neq \mathtt{e}' \wedge \mathtt{OP}(\mathtt{e}).\mathtt{X} \cap \mathtt{OP}(\mathtt{e}').\mathtt{X} \neq \emptyset \implies (\mathtt{e}, \mathtt{e}') \in \mathtt{VIS} \vee (\mathtt{e}', \mathtt{e}) \in \mathtt{VIS}$$

Demostración. La prueba se realiza por inducción de manera análoga a Lem. 4. □

Lema 7 (Correctitud PP)**.** *Si* $\langle \emptyset, \epsilon \rangle \Rightarrow_{\mathtt{PP}} \langle \mathcal{A}, \sigma \rangle$ *entonces* $\mathcal{A} = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR} \rangle$ *y* $\mathtt{AR}; \mathtt{VIS} \subseteq \mathtt{VIS}$.

Demostración. Inducción en la longitud de la derivación $\Rightarrow_{\mathtt{PP}}$

- $\mathbf{n = 0}$. $\mathtt{AR} = \mathtt{VIS} = \emptyset \implies \mathtt{AR}; \mathtt{VIS} = \emptyset \implies \mathtt{AR}; \mathtt{VIS} \subseteq \mathtt{VIS}$ se satisface trivialmente.
- $\mathbf{n = k + 1}$. Entonces,

  $$\langle \emptyset, \epsilon \rangle \Rightarrow_{\mathtt{PP}} \langle \mathcal{A}, \sigma \rangle \rightarrow_{\mathtt{PP}} \langle \mathcal{A}', \sigma' \rangle$$

  Usando hipótesis inductiva, sabemos que vale $\langle \emptyset, \epsilon \rangle \Rightarrow_{\mathtt{PP}} \langle \mathcal{A}, \sigma \rangle$ con $\mathcal{A} = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR} \rangle$ y $\mathtt{AR}; \mathtt{VIS} \subseteq \mathtt{VIS}$, para $\mathbf{k}$ pasos. La prueba se sigue por análisis de casos sobre la última computación.

  - regla (A-COMMIT). $\langle \mathcal{A}, \sigma \rangle \rightarrow_{\mathtt{PP}} \langle \mathcal{A} \overset{\mathtt{e} \rhd \mathsf{T}}{\oplus} \mathtt{E}_\mathtt{i}, \sigma' \rangle$ entonces $\sigma \xrightarrow{\mathtt{e} \rhd \mathsf{T}_\mathtt{i}} \sigma'$.

    Por definición de $\mathcal{A} \overset{\mathtt{e} \rhd \delta}{\oplus} \mathtt{E}_\mathtt{i}$, con $\mathtt{E}_\mathtt{i} = \{\mathtt{e}' \mid \mathtt{e}' \rhd \mathsf{T}' \in \sigma \wedge \mathtt{i} \in \mathsf{T}'.\mathtt{I}\}$ tenemos

    (i) $\mathtt{VIS}' = \mathtt{VIS} \cup (\mathtt{E}_\mathtt{i} \times \{\mathtt{e}\})$.

    (ii) $\mathtt{AR}' = \mathtt{AR} \cup (\mathtt{E}_\mathtt{i} \times \{\mathtt{e}\})$

    Por lo tanto, tenemos que probar que

    $$\mathtt{AR}'; \mathtt{VIS}' \subseteq \mathtt{VIS}'$$

    Entonces,

    $$\begin{array}{lll} \mathtt{AR}'; \mathtt{VIS}' \subseteq \mathtt{VIS}' & \iff & (\mathtt{AR} \cup (\mathtt{E}_\mathtt{i} \times \{\mathtt{e}\})); \mathtt{VIS}' \subseteq \mathtt{VIS}' \quad \text{por (ii)} \\ & \iff & \mathtt{AR}; \mathtt{VIS}' \cup (\mathtt{E}_\mathtt{i} \times \{\mathtt{e}\}); \mathtt{VIS}' \subseteq \mathtt{VIS}' \quad \text{distribuyendo} \end{array}$$

    Notar que $\mathtt{e}$ es fresco, por lo tanto $\nexists (\mathtt{e}, \_) \in \mathtt{VIS}' = \mathtt{VIS} \cup (\mathtt{E}_\mathtt{i} \times \{\mathtt{e}\})$. Entonces, $(\mathtt{E}_\mathtt{i} \times \{\mathtt{e}\}); \mathtt{VIS}' \subseteq \mathtt{VIS}' = \emptyset$. Luego, resta probar $\mathtt{AR}; \mathtt{VIS}' \subseteq \mathtt{VIS}'$. En particular,

    $$\begin{array}{lll} \mathtt{AR}; \mathtt{VIS}' \subseteq \mathtt{VIS}' & \iff & \mathtt{AR}; \mathtt{VIS} \cup (\mathtt{E}_\mathtt{i} \times \{\mathtt{e}\}) \subseteq \mathtt{VIS} \cup (\mathtt{E}_\mathtt{i} \times \{\mathtt{e}\})) \quad \text{por (i)} \\ & \iff & \mathtt{AR}; \mathtt{VIS} \cup \mathtt{AR}; (\mathtt{E}_\mathtt{i} \times \{\mathtt{e}\}) \subseteq \mathtt{VIS} \cup (\mathtt{E}_\mathtt{i} \times \{\mathtt{e}\}) \quad \text{distribuyendo} \end{array}$$

    Hay dos casos:

- $\mathtt{AR}; \mathtt{VIS} \subseteq \mathtt{VIS} \cup (\mathtt{E_i} \times \{\mathtt{e}\})$. Es inmediato por hipotesis inductiva.
- $\mathtt{AR}; (\mathtt{E_i} \times \{\mathtt{e}\}) \subseteq \mathtt{VIS} \cup (\mathtt{E_i} \times \{\mathtt{e}\})$. La única posibilidad para la transición $\sigma \xrightarrow{\mathtt{e} \triangleright \mathsf{T_i}} \sigma'$ es que haya ocurrido la regla (COMMIT), y por lo tanto, se cumpla la condición $\sigma \circledS \mathsf{T}_\mathtt{i}^{\{\mathtt{i}\}}$. El caso interesante es cuando hay algo arbitrado, es decir, $\sigma = \sigma_0 \cdot \mathsf{S} \cdot \sigma_1$ por lo tanto:

$$\sigma_0 \cdot \mathsf{S} \bowtie \sigma_1 \ \wedge \ ((\mathsf{T}.\mathtt{I} \subseteq \mathsf{S}.\mathtt{I}) \vee (\sigma.\mathtt{I} \cap \mathsf{T}.\mathtt{I} = \emptyset))$$

  Luego, hay dos casos:

  1. $(\mathtt{e}'', \mathtt{e}') \in \mathtt{AR}$ con $\mathtt{e}' \in \mathtt{E_i}$.

$$\begin{aligned} (\mathtt{e}'', \mathtt{e}') \in \mathtt{AR} &\iff \forall\, \sigma'' \equiv \sigma, \sigma'' = \sigma_0'' \cdot (\mathtt{e}'' \triangleright \mathsf{S}') \cdot \sigma_1'' \cdot (\mathtt{e}' \triangleright \mathsf{S}) \cdot \sigma_2'' && \text{Def. 5} \\ &\iff \mathsf{S}'.\mathtt{I} \cap \mathsf{S}.\mathtt{I} \neq \emptyset && \text{Def. 2} \end{aligned}$$

     Además, como $\mathtt{e}' \in \mathtt{E_i}$, significa que $\{\mathtt{i}\} \subseteq \mathsf{S}.\mathtt{I} \cap \mathsf{S}'.\mathtt{I}$, y por lo tanto, $\{\mathtt{i}\} \subseteq \mathsf{S}'.\mathtt{I}$, lo que significa que $\mathtt{e}'' \in \mathtt{E_i}$.
  2. $\sigma.\mathtt{I} \cap \mathsf{T}.\mathtt{I} = \emptyset$ implica que $\mathtt{AR}; (\mathtt{E_i} \times \{\mathtt{e}\}) = \emptyset$, por lo que se cumple $\emptyset \subseteq \mathtt{VIS}$.

- regla (A-ARBITRATION). Entonces, $\langle \mathcal{A}, \sigma \rangle \longrightarrow_{\mathtt{PP}} \langle \mathcal{A}', \sigma' \rangle$ con $\mathcal{A}' = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR}' \rangle$ y $\mathtt{AR}' = \mathtt{AR} \cup [\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_k]$. Además, $\sigma \xrightarrow{\tau[\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_k]} \sigma'$. Vamos a probar que

$$\mathtt{AR}'; \mathtt{VIS} \subseteq \mathtt{VIS}$$

  notando que la única regla que aplica es (PROPAGATION), por lo tanto:

$$\sigma = \sigma_0 \cdot \mathtt{e} \triangleright \mathsf{T} \cdot \sigma_1 \cdot \sigma_2 \quad \xrightarrow{\tau[\widetilde{\sigma_0} \cdot \mathtt{e} \cdot \widetilde{\sigma_1}]} \quad \sigma' = \sigma_0 \cdot \mathsf{T}' \cdot \sigma_1 \cdot \sigma_2$$

  sabiendo que:

  (i) $\mathsf{T}' = \mathsf{T}[\mathtt{I} \mapsto \mathsf{T}.\mathtt{I} \cup \{\mathtt{i}\}]$,
  (ii) $\sigma_0 \circledS \mathsf{T}'$ (primera parte del predicado †),
  (iii) $\forall \mathsf{S} \in \sigma_1.\ \mathsf{S}.\mathtt{I} \subseteq \mathsf{T}.\mathtt{I}$ (segunda parte del predicado †),
  (iv) $\mathtt{arb}(\sigma_0 \cdot \mathtt{e} \triangleright \mathsf{T}' \cdot \sigma_1)$,
  (v) $\sigma_1 \cdot \sigma_2 \equiv \sigma_2 \cdot \sigma_1$
  (vi) $\forall \_ \triangleright \mathsf{S} \in \sigma_2.\mathtt{i} \notin \mathsf{S}.\mathtt{I}$.

  Primero usando la definición de $\mathtt{AR}'$, tenemos:

$$(\mathtt{AR} \cup [\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_k]); \mathtt{VIS} \subseteq \mathtt{VIS}$$

  que distribuyendo:

$$\mathtt{AR}; \mathtt{VIS} \cup [\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_k]; \mathtt{VIS} \subseteq \mathtt{VIS}$$

  Luego, por hipótesis inductiva tenemos que $\mathtt{AR}; \mathtt{VIS} \subseteq \mathtt{VIS}$, por lo que nos resta probar:

$$[\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_k]; \mathtt{VIS} \subseteq \mathtt{VIS}$$

  notando que existe un $j \in [0 \cdots k]$ tal que $\mathtt{e}_j = \mathtt{e}$ y

$$[\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_k] = \tau[\widetilde{\sigma_0} \cdot \mathtt{e} \cdot \widetilde{\sigma_1}] = \{(\mathtt{e}', \mathtt{e}) \mid \mathtt{e}' \in \sigma_0\} \cup \{(\mathtt{e}, \mathtt{e}'') \mid \mathtt{e}'' \in \sigma_1\}$$

  Hay dos casos a probar:

- $\forall \mathtt{e}' \in \sigma_0.(\mathtt{e}, \mathtt{e}'') \in \mathtt{VIS}$ entonces $(\mathtt{e}', \mathtt{e}'') \in \mathtt{VIS}$. Este caso se prueba usando la condición (ii), ya que esto significa que vale $\sigma_0 \circledcirc \mathsf{T}'$, con $\sigma_0 = \sigma_0' \cdot \mathsf{S} \ \cdot \sigma_1'$ y por lo tanto

$$\sigma_0' \cdot \mathsf{S} \ \bowtie \sigma_1' \ \wedge \ ((\mathsf{T}.\mathtt{I} \subseteq \mathsf{S}.\mathtt{I}) \vee (\sigma_0.\mathtt{I} \cap \mathsf{T}.\mathtt{I} = \emptyset))$$

  Por Def. 5.2, sabemos que existe $\mathtt{e}'' \triangleright \mathsf{S}'' \in \sigma \iff \mathtt{e}'' \in \mathcal{E}$, y en particular, como $(\mathtt{e}, \mathtt{e}'') \in \mathtt{VIS}$ implica que existe una réplica, tal que $\mathsf{S}''.\mathtt{i} \in \mathsf{T}'.\mathtt{I}$. Entonces, $\mathsf{S}''.\mathtt{i} \in \mathsf{T}.\mathtt{I} \subseteq \mathsf{S}.\mathtt{I}$, lo que significa que $(\mathtt{e}', \mathtt{e}'') \in \mathtt{VIS}$.
- $\forall \mathtt{e}'' \in \sigma_1.(\mathtt{e}'', \mathtt{e}') \in \mathtt{VIS}$ entonces $(\mathtt{e}, \mathtt{e}') \in \mathtt{VIS}$. Luego, por condición (iv) de la regla de propagación, sabemos que $\mathtt{arb}(\mathtt{e} \triangleright \mathsf{T}' \cdot \sigma_1)$, es decir, existe una réplica $\mathtt{j}$ tal que $\forall \mathsf{S} \in \sigma_1.\{\mathtt{j}\} \subseteq \mathsf{T}'.\mathtt{I} \cap \mathsf{S}.\mathtt{I}$. Además por la regla (iii) tenemos que $\forall \mathsf{S} \in \sigma_1.\ \mathsf{S}.\mathtt{I} \subseteq \mathsf{T}.\mathtt{I}$ Entonces, hay dos posibilidades:
  - $\mathtt{j} \neq \mathtt{i}$. Entonces, las transacciones ya estaban arbitradas, por lo que $(\mathtt{e}, \mathtt{e}'') \in \mathtt{AR}$, entonces la prueba es inmediata usando HI.
  - $\mathtt{j} = \mathtt{i}$. Este no puede ser un caso válido, ya que al momento de que ocurre la computación asociada al evento $\mathtt{e}''$ los eventos en σ, en este caso, $\mathtt{e}$ y $\mathtt{e}'$ garantizaban que las transacciones asociadas S' y T respectivamente, cumplían $(\mathsf{T}.\mathtt{I} \subseteq \mathsf{S}'.\mathtt{I} \vee \mathsf{S}'.\mathtt{I} \subseteq \mathsf{T}.\mathtt{I})$ lo cual contradice a que estas transacciones no estuvieran ordenadas.

□

Lema 8 (Completitud PP)**.** *Sea* $\mathcal{A} = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR} \rangle$ *una ejecución abstracta que verifica* $\mathtt{AR}; \mathtt{VIS} \subseteq \mathtt{VIS}$*. Entonces:*

$$\exists \sigma, \mathcal{A}' = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR}' \rangle \text{ tal que } \langle \emptyset, \epsilon \rangle \Rightarrow_{\mathtt{PP}} \langle \mathcal{A}', \sigma \rangle, \ \mathtt{AR} \subseteq \mathtt{AR}' \text{ y } \mathtt{AR}'; \mathtt{VIS} \subseteq \mathtt{VIS}$$

Demostración. La prueba se realiza por inducción de manera análoga a Lem. 4. □

Lema 9 (Correctitud TOTV)**.** *Si* $\langle \emptyset, \epsilon \rangle \Rightarrow_{\mathtt{TOTV}} \langle \mathcal{A}, \sigma \rangle$ *entonces* VIS *es total.*

Demostración. Inducción en el largo $n \geq 0$ de la derivación $\Rightarrow_{\mathtt{TOTV}}$

- $\mathbf{n = 0}$. $\mathtt{VIS} = \emptyset \implies \mathtt{VIS}$ es total trivialmente.
- $\mathbf{n = k + 1}$. Asumimos que $\langle \emptyset, \epsilon \rangle \Rightarrow_{\mathtt{TOTV}} \langle \mathcal{A}, \sigma \rangle$ tal que $\mathcal{A} = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR} \rangle$ y VIS es total, en $\mathbf{k}$ pasos. Derivo en 1 paso $\langle \mathcal{A}, \sigma \rangle \rightarrow_{\mathtt{TOTV}} \langle \mathcal{A}', \sigma' \rangle$, por lo tanto $\langle \emptyset, \epsilon \rangle \Rightarrow_{\mathtt{TOTV}} \langle \mathcal{A}', \sigma' \rangle$ en $\mathbf{k + 1}$ pasos. Sea $\mathcal{A}' = \langle \mathcal{E}', \mathtt{OP}', \mathtt{VIS}', \mathtt{AR}' \rangle$.

  La prueba se sigue por análisis de casos sobre la última computación.

  - regla (A-COMMIT). $\langle \mathcal{A}, \sigma \rangle \rightarrow_{\mathtt{PP}} \langle \mathcal{A} \overset{\mathtt{e}' \triangleright \mathsf{T}'}{\oplus} \mathtt{E_i}, \sigma' \rangle$ entonces $\sigma \xrightarrow{\mathtt{e}' \triangleright \mathsf{T}_\mathtt{i}'} \sigma'$. La única regla que aplica es $\mathtt{COMMIT} - \mathtt{E}$. Debemos comprobar que $\mathtt{VIS}' = \mathtt{VIS} \cup (\mathtt{E_i} \times \{\mathtt{e}'\})$ es total. Para ello, comprobamos las propiedades:
    - TRANSITIVIDAD: $\forall \mathtt{a}, \mathtt{b}, \mathtt{c} \in \mathcal{E}'.\ (\mathtt{a}, \mathtt{b}) \in \mathtt{VIS}' \wedge (\mathtt{b}, \mathtt{c}) \in \mathtt{VIS}' \implies (\mathtt{a}, \mathtt{c}) \in \mathtt{VIS}'$

      Notamos que $\mathtt{a} \neq \mathtt{e}' \wedge \mathtt{b} \neq \mathtt{e}'$ debido a que no puede ocurrir que $(\mathtt{e}', \_)$ por ser $\mathtt{e}'$ fresca. Entonces $\mathtt{a} \in \mathcal{E} \wedge \mathtt{b} \in \mathcal{E}$. Por otra parte, si $\mathtt{c} \in \mathcal{E}$, la propiedad se cumple trivialmente por hipótesis inductiva. Consideramos el caso restante $\mathtt{a} \in \mathcal{E} \wedge \mathtt{b} \in \mathcal{E} \wedge \mathtt{c} = \mathtt{e}'$.

Llamamos $\texttt{OP}(\texttt{a}) = \texttt{T}$ y $\texttt{OP}(\texttt{c}) = \texttt{T}'$. Por $\texttt{COMMIT} - \texttt{E}$, $\sigma \circledS \texttt{T}_\texttt{i}^{\{\texttt{i}\}} \xrightarrow{\texttt{TOTV}} \texttt{T}'.\texttt{I} \subseteq \texttt{T}.\texttt{I}$. Por lo tanto, $\texttt{i} = \texttt{T}'.\texttt{i} \in \texttt{T}'.\texttt{I} \subseteq \texttt{T}.\texttt{I} \implies \texttt{T} \in \texttt{E}_\texttt{i} \implies (\texttt{a}, \texttt{c}) \in (\texttt{E}_\texttt{i} \times \{\texttt{e}'\}) \subseteq \texttt{VIS}'$.

- Antisimetria: $\forall \texttt{a}, \texttt{b} \in \mathcal{E}'. (\texttt{a}, \texttt{b}) \in \texttt{VIS}' \wedge (\texttt{b}, \texttt{a}) \in \texttt{VIS}' \implies \texttt{a} = \texttt{b}$

  No puede ocurrir $(\texttt{a}, \texttt{b}) \in \texttt{VIS}' \vee (\texttt{b}, \texttt{a}) \in \texttt{VIS}'$ dado que $\mathcal{A}'$ es una ejecución abstracta y por lo tanto $\texttt{VIS}'$ es acíclica. La proposición se cumple trivialmente para todo $\texttt{a}, \texttt{b} \in \mathcal{E}'$.

- Totalidad: $\forall \texttt{a}, \texttt{b} \in \mathcal{E}'. (\texttt{a}, \texttt{b}) \in \texttt{VIS}' \wedge (\texttt{b}, \texttt{a}) \in \texttt{VIS}'$

  Si tomo $\texttt{a} \in \mathcal{E} \wedge \texttt{b} \in \mathcal{E}$, la propiedad se cumple trivialmente por hipótesis inductiva. Otro caso trivial es $\texttt{a} = \texttt{b} = \texttt{e}'$. Consideramos, sin pérdida de generalidad, $\texttt{a} \in \mathcal{E} \wedge \texttt{b} = \texttt{e}'$. Sí $\texttt{b} = \texttt{e}'$, la única alternativa es que $\texttt{a} \in \texttt{E}_\texttt{i}$ por construcción de $\texttt{VIS}'$ teniendo en cuenta que $\texttt{e}'$ es fresca. Siendo $\texttt{T} = \texttt{OP}(\texttt{a})$ y $\texttt{T}' = \texttt{OP}(\texttt{b})$, por $\texttt{COMMIT} - \texttt{E}$ se cumple $\sigma \circledS \texttt{T}_\texttt{i}^{\{\texttt{i}\}} \xrightarrow{\texttt{TOTV}} \texttt{T}'.\texttt{I} \subseteq \texttt{T}.\texttt{I}$. Por lo tanto, $\texttt{i} = \texttt{T}'.\texttt{i} \in \texttt{T}'.\texttt{I} \subseteq \texttt{T}.\texttt{I} \implies \texttt{a} \in \texttt{E}_\texttt{i} \implies (\texttt{a}, \texttt{b}) \in (\texttt{E}_\texttt{i} \times \{\texttt{e}'\}) \subseteq \texttt{VIS}'$.
- (A-PROPAGATION). Observando que $\texttt{VIS}' = \texttt{VIS}$, por hipótesis inductiva, se verifica trivialmente.

□

Lema 10 (Completitud TOTV)**.** *Sea* $\mathcal{A} = \langle \mathcal{E}, \texttt{OP}, \texttt{VIS}, \texttt{AR} \rangle$ *una ejecución abstracta que verifica* VIS *es total. Entonces:*

$$\exists \sigma, \mathcal{A}' = \langle \mathcal{E}, \texttt{OP}, \texttt{VIS}, \texttt{AR}' \rangle \text{ tal que } \langle \emptyset, \epsilon \rangle \Rightarrow_{\texttt{TOTV}} \langle \mathcal{A}', \sigma \rangle,\ \texttt{AR} \subseteq \texttt{AR}' \text{ y } \texttt{VIS} \text{ es total}$$

Demostración. La prueba se realiza por inducción de manera análoga a Lem. 4. □

Lema 11 (Correctitud PSI)**.** *Si* $\langle \emptyset, \epsilon \rangle \Rightarrow_{\texttt{PSI}} \langle \mathcal{A}, \sigma \rangle$ *entonces* $\mathcal{A} = \langle \mathcal{E}, \texttt{OP}, \texttt{VIS}, \texttt{AR} \rangle$*, tal que* VIS *es* transitiva *y* $\forall \texttt{e}, \texttt{e}' \in \mathcal{E}. \texttt{e} \neq \texttt{e}' \wedge \texttt{OP}(\texttt{e}).\texttt{X} \cap \texttt{OP}(\texttt{e}').\texttt{X} \neq \emptyset \implies (\texttt{e}, \texttt{e}') \in \texttt{VIS} \vee (\texttt{e}', \texttt{e}) \in \texttt{VIS}$.

Demostración. Inducción en la longitud de la derivación $\Rightarrow_{\texttt{PSI}}$

- $\mathbf{n = 0}$. $\mathcal{E} = \emptyset \implies \texttt{VIS} = \emptyset$ por lo que se satisface inmediatamente.
- $\mathbf{n = k + 1}$. Entonces,
$$\langle \emptyset, \epsilon \rangle \Rightarrow_{\texttt{PSI}} \langle \mathcal{A}, \sigma \rangle \rightarrow_{\texttt{PSI}} \langle \mathcal{A}', \sigma' \rangle$$
Usando hipótesis inductiva, para $\mathbf{k}$ pasos sabemos que vale $\langle \emptyset, \epsilon \rangle \Rightarrow_{\texttt{PSI}} \langle \mathcal{A}, \sigma \rangle$ con
  - $\mathcal{A} = \langle \mathcal{E}, \texttt{OP}, \texttt{VIS}, \texttt{AR} \rangle$,
  - VIS *transitiva* y
  - $\forall \texttt{e}, \texttt{e}' \in \mathcal{E}. \texttt{e} \neq \texttt{e}' \wedge \texttt{OP}(\texttt{e}).\texttt{X} \cap \texttt{OP}(\texttt{e}').\texttt{X} \neq \emptyset \implies (\texttt{e}, \texttt{e}') \in \texttt{VIS} \vee (\texttt{e}', \texttt{e}) \in \texttt{VIS}$.

  La prueba se sigue por análisis de casos sobre la última computación.

  - regla (A-COMMIT). $\langle \mathcal{A}, \sigma \rangle \rightarrow_{\texttt{PSI}} \langle \mathcal{A} \overset{\texttt{e} \triangleright \texttt{T}}{\oplus} \texttt{E}_\texttt{i}, \sigma' \rangle$ entonces $\sigma \xrightarrow{\texttt{e} \triangleright \texttt{T}_\texttt{i}} \sigma'$.
    Por definición de $\mathcal{A} \overset{\texttt{e} \triangleright \delta}{\oplus} \texttt{E}_\texttt{i}$, con $\texttt{E}_\texttt{i} = \{\texttt{e}' \mid \texttt{e}' \triangleright \texttt{T}' \in \sigma \wedge \texttt{i} \in \texttt{T}'.\texttt{I}\}$ tenemos

(i) $\texttt{VIS}' = \texttt{VIS} \cup (\texttt{E}_\texttt{i} \times \{\texttt{e}\})$.
(ii) $\texttt{AR}' = \texttt{AR} \cup (\texttt{E}_\texttt{i} \times \{\texttt{e}\})$
(iii) $\mathcal{E}' = \mathcal{E} \cup \{\texttt{e}\}$

Por lo tanto, tenemos que probar que (I) $\texttt{VIS}'$ es transitiva y (II) que vale $\forall \texttt{e}', \texttt{e}'' \in \mathcal{E}'.\, \texttt{e}' \neq \texttt{e}'' \wedge \texttt{OP}(\texttt{e}').\texttt{X} \cap \texttt{OP}(\texttt{e}'').\texttt{X} \neq \emptyset \implies (\texttt{e}', \texttt{e}'') \in \texttt{VIS}' \vee (\texttt{e}'', \texttt{e}') \in \texttt{VIS}'$.

Como la única posibilidad para la transición $\sigma \xrightarrow{\texttt{e} \rhd \texttt{T}_\texttt{i}} \sigma'$ es que haya ocurrido la regla (COMMIT), entonces sabemos que se cumplió la condición $\sigma \circledS \texttt{T}_\texttt{i}^{\{\texttt{i}\}}$:

$$\forall \texttt{S} \in \sigma.\ (\texttt{T}.\texttt{X} \cap \texttt{S}.\texttt{X} \neq \emptyset \vee \texttt{T}.\texttt{i} \in \texttt{S}.\texttt{I}) \implies \texttt{T}.\texttt{I} \subseteq \texttt{S}.\texttt{I}$$

Luego, como el antecedente se cumple, en particular $\texttt{T}.\texttt{i} \in \texttt{S}.\texttt{I}$ vale. En consecuencia, la prueba para mostrar que la visibilidad es causal, es decir la condición (I), es inmediata usando Lem. 3, el caso para la regla (A-COMMIT).

El caso para mostrar que no hay conflictos, es decir (II), se desprende de saber que $\texttt{T}.\texttt{I} \subseteq \texttt{S}.\texttt{I}$ implica $\texttt{T}.\texttt{i} \in \texttt{S}.\texttt{I}$. Por lo tanto, se demuestra usando Lem. 5, el caso para la regla (A-COMMIT).

- regla (A-ARBITRATION). Entonces, $\langle \mathcal{A}, \sigma \rangle \rightarrow_{\texttt{PP}} \langle \mathcal{A}', \sigma' \rangle$ con $\mathcal{A}' = \langle \mathcal{E}, \texttt{OP}, \texttt{VIS}, \texttt{AR}' \rangle$ y $\texttt{AR}' = \texttt{AR} \cup [\texttt{e}_0, \texttt{e}_1, \ldots, \texttt{e}_k]$. Además, $\sigma \xrightarrow{\tau[\texttt{e}_0, \texttt{e}_1, \ldots, \texttt{e}_k]} \sigma'$. Entonces, tenemos que probar que (i) $\texttt{VIS}'$ es transitiva y (ii) $\forall \texttt{e}, \texttt{e}' \in \mathcal{E}.\, \texttt{e} \neq \texttt{e}' \wedge \texttt{OP}(\texttt{e}).\texttt{X} \cap \texttt{OP}(\texttt{e}').\texttt{X} \neq \emptyset \implies (\texttt{e}, \texttt{e}') \in \texttt{VIS} \vee (\texttt{e}', \texttt{e}) \in \texttt{VIS}$. La única regla que aplica es (PROPAGATION), por lo tanto vale que

  $$\forall \texttt{S} \in \sigma.\ (\texttt{T}.\texttt{X} \cap \texttt{S}.\texttt{X} \neq \emptyset \vee \texttt{T}.\texttt{i} \in \texttt{S}.\texttt{I}) \implies \texttt{T}.\texttt{I} \subseteq \texttt{S}.\texttt{I}$$

  Luego, como el antecedente se cumple, en particular $\texttt{T}.\texttt{i} \in \texttt{S}.\texttt{I}$ vale. En consecuencia, la prueba para mostrar que la visibilidad es causal, es decir la condición (i), es inmediata usando Lem. 3, el caso para la regla (A-ARBITRATION).

  El caso para mostrar que no hay conflictos, es decir (ii), se desprende de saber que $\texttt{T}.\texttt{I} \subseteq \texttt{S}.\texttt{I}$ implica $\texttt{T}.\texttt{i} \in \texttt{S}.\texttt{I}$. Por lo tanto, se demuestra usando Lem. 5, el caso para la regla (A-ARBITRATION)

□

Lema 12 (Correctitud SI). *Si* $\langle \emptyset, \epsilon \rangle \Rightarrow_{\texttt{SI}} \langle \mathcal{A}, \sigma \rangle$ *entonces* $\mathcal{A} = \langle \mathcal{E}, \texttt{OP}, \texttt{VIS}, \texttt{AR} \rangle$ *tal que* $\texttt{AR}; \texttt{VIS} \subseteq \texttt{VIS}$ *y* $\forall \texttt{e}, \texttt{e}' \in \mathcal{E}.\, \texttt{e} \neq \texttt{e}' \wedge \texttt{OP}(\texttt{e}).\texttt{X} \cap \texttt{OP}(\texttt{e}').\texttt{X} \neq \emptyset \implies (\texttt{e}, \texttt{e}') \in \texttt{VIS} \vee (\texttt{e}', \texttt{e}) \in \texttt{VIS}$

Demostración. Inducción en la longitud de la derivación $\Rightarrow_{\texttt{SI}}$

- $\mathbf{n = 0}$. $\mathcal{E} = \emptyset \implies \texttt{VIS} = \texttt{AR} = \emptyset$ por lo que se satisface inmediatamente.
- $\mathbf{n = k + 1}$. Entonces,

  $$\langle \emptyset, \epsilon \rangle \Rightarrow_{\texttt{SI}} \langle \mathcal{A}, \sigma \rangle \rightarrow_{\texttt{SI}} \langle \mathcal{A}', \sigma' \rangle$$

  Usando hipótesis inductiva, para $\mathbf{k}$ pasos sabemos que vale $\langle \emptyset, \epsilon \rangle \Rightarrow_{\texttt{SI}} \langle \mathcal{A}, \sigma \rangle$ con

  - $\mathcal{A} = \langle \mathcal{E}, \texttt{OP}, \texttt{VIS}, \texttt{AR} \rangle$,
  - $\texttt{AR}; \texttt{VIS} \subseteq \texttt{VIS}$ y
  - $\forall \texttt{e}, \texttt{e}' \in \mathcal{E}.\, \texttt{e} \neq \texttt{e}' \wedge \texttt{OP}(\texttt{e}).\texttt{X} \cap \texttt{OP}(\texttt{e}').\texttt{X} \neq \emptyset \implies (\texttt{e}, \texttt{e}') \in \texttt{VIS} \vee (\texttt{e}', \texttt{e}) \in \texttt{VIS}$

La prueba se sigue por análisis de casos sobre la última computación.

- regla (A-COMMIT). $\langle \mathcal{A}, \sigma \rangle \rightarrow_{\mathtt{SI}} \langle \mathcal{A} \overset{\mathtt{e} \rhd \mathtt{T}}{\oplus} \mathtt{E_i}, \sigma' \rangle$ entonces $\sigma \xrightarrow{\mathtt{e} \rhd \mathtt{T_i}} \sigma'$.

  Por definición de $\mathcal{A} \overset{\mathtt{e} \rhd \delta}{\oplus} \mathtt{E_i}$, con $\mathtt{E_i} = \{\mathtt{e}' \mid \mathtt{e}' \rhd \mathtt{T}' \in \sigma \wedge \mathtt{i} \in \mathtt{T}'.\mathtt{I}\}$ tenemos

  (i) $\mathtt{VIS}' = \mathtt{VIS} \cup (\mathtt{E_i} \times \{\mathtt{e}\})$

  (ii) $\mathtt{AR}' = \mathtt{AR} \cup (\mathtt{E_i} \times \{\mathtt{e}\})$

  (iii) $\mathcal{E}' = \mathcal{E} \cup \{\mathtt{e}\}$

  Por lo tanto, tenemos que probar que

  (I) $\mathtt{AR}'; \mathtt{VIS}' \subseteq \mathtt{VIS}'$

  (II) $\forall \mathtt{e}', \mathtt{e}'' \in \mathcal{E}'.\, \mathtt{e}' \neq \mathtt{e}'' \wedge \mathtt{OP}(\mathtt{e}').\mathtt{X} \cap \mathtt{OP}(\mathtt{e}'').\mathtt{X} \neq \emptyset \implies (\mathtt{e}', \mathtt{e}'') \in \mathtt{VIS}' \vee (\mathtt{e}'', \mathtt{e}') \in \mathtt{VIS}'$

  Como la única posibilidad para la transición $\sigma \xrightarrow{\mathtt{e} \rhd \mathtt{T_i}} \sigma'$ es que haya ocurrido la regla (COMMIT), entonces sabemos que se cumplió la condición $\sigma \Subset \mathtt{T}_\mathtt{i}^{\{\mathtt{i}\}}$:

  $$\mathtt{AR}; \mathtt{VIS} \subseteq \mathtt{VIS} \wedge$$
  $$(\forall \mathtt{e}, \mathtt{e}' \in \mathcal{E}.\, \mathtt{e} \neq \mathtt{e}' \wedge \mathtt{OP}(\mathtt{e}).\mathtt{X} \cap \mathtt{OP}(\mathtt{e}').\mathtt{X} \neq \emptyset \implies (\mathtt{e}, \mathtt{e}') \in \mathtt{VIS} \vee (\mathtt{e}', \mathtt{e}) \in \mathtt{VIS})$$

  En consecuencia, la prueba para mostrar que la condición (I), es inmediata usando Lem. 7, el caso para la regla (A-COMMIT). De manera análoga, para mostrar que no hay conflictos, es decir cumplir la condición (II), usamos Lem. 5, el caso para la regla (A-COMMIT).

- regla (A-ARBITRATION). Entonces, $\langle \mathcal{A}, \sigma \rangle \rightarrow_{\mathtt{PP}} \langle \mathcal{A}', \sigma' \rangle$ con $\mathcal{A}' = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR}' \rangle$ y $\mathtt{AR}' = \mathtt{AR} \cup [\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_k]$. Además, $\sigma \xrightarrow{\tau[\mathtt{e}_0, \mathtt{e}_1, \ldots, \mathtt{e}_k]} \sigma'$. Entonces, tenemos que probar que

  (I) $\mathtt{AR}'; \mathtt{VIS} \subseteq \mathtt{VIS}$

  (II) $\forall \mathtt{e}, \mathtt{e}' \in \mathcal{E}.\, \mathtt{e} \neq \mathtt{e}' \wedge \mathtt{OP}(\mathtt{e}).\mathtt{X} \cap \mathtt{OP}(\mathtt{e}').\mathtt{X} \neq \emptyset \implies (\mathtt{e}, \mathtt{e}') \in \mathtt{VIS} \vee (\mathtt{e}', \mathtt{e}) \in \mathtt{VIS}$

  La única regla que aplica es (PROPAGATION), por lo tanto vale que

  $$\mathtt{AR}; \mathtt{VIS} \subseteq \mathtt{VIS} \wedge$$
  $$(\forall \mathtt{e}, \mathtt{e}' \in \mathcal{E}.\, \mathtt{e} \neq \mathtt{e}' \wedge \mathtt{OP}(\mathtt{e}).\mathtt{X} \cap \mathtt{OP}(\mathtt{e}').\mathtt{X} \neq \emptyset \implies (\mathtt{e}, \mathtt{e}') \in \mathtt{VIS} \vee (\mathtt{e}', \mathtt{e}) \in \mathtt{VIS})$$

  Luego, como el antecedente se cumple, usamos el mismo razonamiento que para la regla (COMMIT), la condición (I) vale usando Lem. 7, el caso para la regla (A-ARBITRATION), mientras que el caso (II) que expresa que no hay conflictos, se demuestra por medio de Lem. 5, el caso para la regla (A-ARBITRATION)

□

Teorema 4.4.1 (Correctitud)**.** *Si* $\langle \emptyset, \epsilon \rangle \Rightarrow_{\mathtt{C}} \langle \mathcal{A}, \sigma \rangle$ *entonces* $\mathcal{A} = \langle \mathcal{E}, \mathtt{OP}, \mathtt{VIS}, \mathtt{AR} \rangle$*, tal que*

$\mathtt{C} = \mathtt{CC}$ *y* TRANSVIS

$\mathtt{C} = \mathtt{PSI}$ *y* (TRANSVIS $\wedge$ NOCONFLICT)

$\mathtt{C} = \mathtt{PC}$ *y* PREFIX

$\mathtt{C} = \mathtt{SI}$ *y* (PREFIX $\wedge$ NOCONFLICT)

$\mathtt{C} = \mathtt{SER}$ *y* TOTALVIS

Demostración. La prueba es inmediata usando los lemas anteriores. □

5. CONCLUSIONES

Presentamos un framework operacional que permite especificar modelos de consistencia transaccional sobre bases de datos replicadas. La ventaja de esta caracterización es la simplicidad con la que uno puede alcanzar dichas garantías instanciando dos operadores. En particular, nuestro enfoque operacional ofrece una base teórica para desarrollar técnicas que permiten razonar sobre la correctitud de los programas que correr sobre bases de datos replicadas con transacciones.

Por otro lado, remarcamos que nuestro modelo sienta las bases al desarrollo de posibles implementaciones, en contraposición a los modelos axiomáticos existentes. Por lo tanto, si bien no damos detalles de implementación a bajo nivel, damos las condiciones necesarias para alcanzar distintos niveles de consistencia a partir de una máquina operacional.

Nuestro modelo presenta la definición de estado de una manera simple: una secuencia de transacciones. Esencialmente este estado sólo puede ser modificado por tres reglas, de las cuáles sólo la propagación es no trivial. Si bien esta solución no contempla los casos de error, es una primera propuesta, pero planeamos dar una versión robusta de nuestro modelo como parte del trabajo futuro. Remarcamos, que nuestro *framework* respeta un principio fundamental: un diseño poco acoplado y altamente cohesivo, por lo que considerar el caso para *rollback* significa agregar una nueva regla, sin alterar las preexistentes.

Hemos ilustrado el uso del modelo especificando varios modelos de consistencia existentes y encapsulándolos a través de los operadores. También hemos validado este modelo con las especificaciones axiomáticas propuestas en [7] demostrando la equivalencia que hay entre estos. Otro punto a remarcar, es que nuestro framework describe exactamente cómo se altera el estado sistema generando en cada paso, una extensión correcta con respecto al modelo de consistencia a alcanzar. Si bien los operadores paramétricos que utilizamos son efectivamente predicados, estos son quiénes restringen las computaciones. Este enfoque es la principal diferencia con [7]. Allí desarrollan un modelado operacional en el que se generan las posibles trazas (sobreaproximación) para luego descartar las que no cumplen ciertas garantías de consistencia, similar a la técnica *generate-and-test*.

En cuánto al trabajo futuro, planeamos ofrecer un modelo operacional más a bajo nivel, considerando operaciones de lectura. Esto significa agregar restricciones adicionales para razonar sobre la correctitud de los programas al realizar lecturas. Por ejemplo, considere una lectura en una transacción. Esta debe asegurar leer la última escritura vista que puede ser hecha en la misma transacción, o en otra transacción realizada por una misma sesión o incluso otra. Esto agregaría valor a nuestro modelo. Además planeamos considerar semáforos sobre las réplicas para introducir nuevas transacciones. Esto es, mecanismos para bloquear una réplica que procesa una transacción.

Bibliografía

[1] M. S. Ardekani, P. Sutra, and M. Shapiro. Non-monotonic snapshot isolation: Scalable and strong consistency for geo-replicated transactional systems. In *2013 IEEE 32nd International Symposium on Reliable Distributed Systems*, pages 163–172. IEEE, 2013.

[2] P. Bailis and A. Ghodsi. Eventual consistency today: Limitations, extensions, and beyond. *Queue*, 11(3):20–32, 2013.

[3] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P. O'Neil. A critique of ansi sql isolation levels. *ACM SIGMOD Record*, 24(2):1–10, 1995.

[4] S. Burckhardt et al. Principles of eventual consistency. *Foundations and Trends® in Programming Languages*, 1(1-2):1–150, 2014.

[5] S. Burckhardt, D. Leijen, M. Fähndrich, and M. Sagiv. Eventually consistent transactions. In *European Symposium on Programming*, pages 67–86. Springer, 2012.

[6] S. Burckhardt, D. Leijen, J. Protzenko, and M. Fähndrich. Global sequence protocol: A robust abstraction for replicated shared state. In *29th European Conference on Object-Oriented Programming (ECOOP 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[7] A. Cerone, G. Bernardi, and A. Gotsman. A framework for transactional consistency models with atomic visibility. In *26th International Conference on Concurrency Theory (CONCUR 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[8] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazon's highly available key-value store. *ACM SIGOPS operating systems review*, 41(6):205–220, 2007.

[9] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Acm Sigact News*, 33(2):51–59, 2002.

[10] A. Lakshman and P. Malik. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2):35–40, 2010.

[11] W. Lloyd, M. J. Freedman, M. Kaminsky, and D. G. Andersen. Don't settle for eventual: scalable causal consistency for wide-area storage with cops. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 401–416, 2011.

[12] I. MongoDB. Mongodb. *URL https://www. mongodb. com/. Cited on*, page 9, 2016.

[13] C. H. Papadimitriou. The serializability of concurrent database updates. *Journal of the ACM (JACM)*, 26(4):631–653, 1979.

[14] D. B. Terry, A. J. Demers, K. Petersen, M. J. Spreitzer, M. M. Theimer, and B. B. Welch. Session guarantees for weakly consistent replicated data. In *Proceedings of 3rd International Conference on Parallel and Distributed Information Systems*, pages 140–149. IEEE, 1994.

[15] P. Viotti and M. Vukolić. Consistency in non-transactional distributed storage systems. *ACM Computing Surveys (CSUR)*, 49(1):1–34, 2016.