Facultad de Matemática, Astronomía y Física

Trabajo especial de la Licenciatura en Ciencias de la Computación

Lógicas modales con datos infinitos

Autora: Gisela C. Rossi

Director: Dr. Carlos Areces

Marzo de 2015

Índice general

Resumen

En esta tesis extendemos la lógica dinámica proposicional (PDL, *Propositional Dynamic Logic*) con variables que toman valores en un dominio infinito. Esta extensión, llamada PDL *parametrizada* o PPDL es interpretada sobre sistemas de transición parametrizados cuyas aristas están etiquetadas con letras o variables y cuyos estados están etiquetados con proposiciones no parametrizadas. Nuestro resultado es demostrar que el problema de satisfabilidad para PPDL es decidible cuando es interpretado sobre la subclase de sistemas de transición parametrizados en los cuales las variables pueden ser reseteadas.

We extend propositional dynamic logic (PDL) with variables ranging over an infinite domain. This extension, called parametrized PDL or PPDL for short, is interpreted over parametrized transitions systems whose edges are labeled with letters or variables and whose states are labeled with non-parametrized propositions. The result of this work shows that the satisfiability problem for PPDL is decidable when interpreted over the subclass of parametrized transition systems in which variables can be reset.

Reconocimientos Este trabajo fue realizado durante una pasantía en INRIA (Institut national de recherche en informatique et en automatique) realizada en la segunda mitad de 2013. Durante esta pasantía, trabajé con el Dr. Michael Rusinowitch y el Dr. Walid Belkhir, del equipo CASSIS. Los resultados de esta colaboración fueron publicados como "A parametrized propositional dynamic logic with application to service synthesis" en la conferencia Advances in Modal Logic 2014. Estos resultados forman la parte principal de mi tesis, además han motivado futuros temas de investigación como el problema de model checking para PPDL y su complejidad.

Agradecimientos A la Universidad Nacional de Córdoba, pública y gratuita, sin la cual probablemente no hubiese podido seguir estudios superiores.
A Carlos Areces, quien ha sido no sólo mi director en esta tesis sino también mi mentor en los últimos cuatro años. Su consejo y su ayuda han sido y son invaluables.
A Daniel, por su inmenso cariño y apoyo a lo largo de estos años.

Capítulo 1

Introducción

Contribuciones. Introducimos sistemas de transición parametrizados (PTS, Parameterized Transition Systems) y PDL parametrizada (PPDL, Parameterized Propositional Dynamic Logic). Las transiciones en PTS, son etiquetadas con variables a las que puede ser asignada un valor. Una vez que se le ha asignado un valor a una variable, esta queda *ligada* a dicho valor. Para poder asignarla a un nuevo valor, primero debe ser *reseteada*. Dicho reseteo sólo puede darse en ciertos estados establecidos en la definición del PTS. Este mecanismo es natural para expresar procesos de iteración, como por ejemplo cuando un servidor tiene que scanear una lista de identificadores de elementos o sesiones. La posibilidad de usar variables y resetearlas es útil para aplicaciones del mundo real donde las acciones de los servidores están parametrizadas con términos construidos a partir de datos tomados de alfabetos infinitos (identificadores, código, direcciones ...). Los estados de un PTS están etiquetados con constantes proposicionales no parametrizadas.

En PDL standard encontramos dos entidades: fórmulas y programas, donde los programas son expresiones regulares construidas sobre un conjunto finito de acciones atómicas usando concatenación, unión y el operador de Kleene. Para PPDL además consideramos expresiones regulares *parametrizadas*, permitiendo variables en las expresiones regulares. Las variables en cuestión toman valores sobre un conjunto infinito de acciones Σ. Estas expresiones parametrizadas son las que nos permiten trabajar con datos infinitos. Con el fin de liberar una variable después de que ha sido ligada a una acción, introduciremos el operador de *reseteo* $\mathsf{res}(.)$. Por ejemplo, la expresión $(x; \mathsf{res}(x))^\star$ donde $\mathsf{res}(x)$ denota el reseteo de la variable x, representa todas las posibles trazas finitas en $\Sigma^\star$, es decir trazas de la forma $a_1 a_2 \dots a_n$, donde $a_i \in \Sigma$ y $n \in \mathbb{N}$. La fórmula PPDL $\phi_1 = \forall x.[(x; \mathsf{res}(x))^\star]\mathtt{p}$, donde $[-]$ representa el operador modal de *necesidad*, $\mathtt{p}$ es una constante proposicional y x es una variable, significa que $\mathtt{p}$ vale globalmente, es decir, $\mathtt{p}$ vale en todos los estados alcanzables del modelo. Después introducimos juegos de satisfactibilidad para PPDL y probamos su completitud, luego mostramos la decidibilidad del problema de satisfactibilidad de PPDL.

Organización del trabajo. Este trabajo esta organizado como sigue: El capítulo 2 introduce algunas definiciones básicas que serán útiles para la comprensión del trabajo. El capítulo 3 muestra un ejemplo del problema que queremos resolver y por qué resulta natural usar este enfoque. El capítulo 4 introduce los sistemas de transición parametrizados y PDL parametrizada. El capítulo 5 introduce juegos de satisfactibilidad para PPDL. El capítulo 6 prueba la decidibilidad del problema de satisfactibilidad de PPDL.

Capítulo 2

Definiciones Básicas

En este capítulo definimos algunos conceptos básicos que servirán para la comprensión del trabajo. Estas definiciones se basan en [Zalta, 2007] y [Blackburn *et al.*, 2001].

Definición 2.1. Sea $\mathcal{X}$ un conjunto finito de variables, Σ un conjunto infinito de acciones atómicas. Una sustitución es un mapeo idempotente $\{x_1 \mapsto \alpha_1, \ldots, x_n \mapsto \alpha_n\} \cup \bigcup_{a \in \Sigma}\{a \mapsto a\}$ con variables $x_1, \ldots, x_n$ en $\mathcal{X}$ y $\alpha_1, \ldots, \alpha_n$ en $\mathcal{X} \cup \Sigma$.

Sea σ una sustitución, llamamos $\{x_1, \ldots, x_n\}$ a su *dominio propio*, y lo denotamos con $dom(\sigma)$. Denotamos con $Dom(\sigma)$ el conjunto $dom(\sigma) \cup \Sigma$. Denotamos con $codom(\sigma)$ el conjunto $\{a \in \Sigma \mid \exists x \in dom(\sigma) \text{ s.t. } \sigma(x) = a\}$. La sustitución vacía (*i.e.*, con un dominio propio vacío) es denotada con $\emptyset$. El conjunto de sustituciones de $\mathcal{X} \cup \Sigma$ a un conjunto A es denotado con $\zeta_{\mathcal{X},A}$, o con $\zeta_{\mathcal{X}}$, o simplemente con ζ si no hay ambigüedad. Si σ_1 y σ_2 son sustituciones que coinciden en $dom(\sigma_1) \cap dom(\sigma_2)$, entonces $\sigma_1 \cup \sigma_2$ denota su unión en el sentido usual. Si $dom(\sigma_1) \cap dom(\sigma_2) = \emptyset$ entonces denotamos con $\sigma_1 \uplus \sigma_2$ su unión *disjunta*. Definimos una función $\mathcal{V} : \Sigma \cup \mathcal{X} \longrightarrow \mathcal{P}(\mathcal{X})$ con $\mathcal{V}(\alpha) = \{\alpha\}$ si $\alpha \in \mathcal{X}$, y $\mathcal{V}(\alpha) = \emptyset$, sino. Para una función $F : A \to B$, y $A' \subseteq A$, la restricción de F en A' es denotada con $F_{|A'}$.

Definición 2.2. Sea $\mathbb{P}$ un conjunto de símbolos proposicionales y MOD un conjunto de símbolos modales. Definimos las fórmulas de la lógica modal básica, donde $\mathtt{m} \in MOD$ y $\mathtt{p} \in \mathbb{P}$, donde $\mathtt{p}$ es una variable proposicional:

$$\phi ::= \mathtt{p} \mid \top \mid \neg\phi \mid \phi \wedge \phi \mid [\mathtt{m}]\phi$$

Los booleanos se definen de la forma usual. Para todo $\mathtt{m}$, $\langle \mathtt{m} \rangle \phi$ se define como $\neg[\mathtt{m}]\neg\phi$. Es decir, diamante y cuadrado son conectivos duales.

Definición 2.3. Un modelo de Kripke $\mathfrak{M}$ para la lógica modal básica es una tripla $\mathfrak{M} = (W, \{R^m\}_{m \in MOD}, V)$ donde: W, el dominio, es un conjunto no vacío cuyos elementos son llamados *estados*. Cada R^m en el modelo es una relación binaria sobre W. Por último, V es una función de valuación que asigna a cada símbolo proposicional p un subconjunto $V(p)$ de W (este conjunto puede entenderse como el conjunto de estados en $\mathfrak{M}$ donde p es verdadera).

Un representante importante de la familia de las lógicas modales es la lógica proposicional dinámica (PDL, Proposicional Dynamic Logic). En esta lógica la formula $[\alpha]\phi$ se interpreta como: toda ejecución del programa α desde el estado actual lleva a un estado satisface ϕ. Notar que los símbolos modales son los programas. PDL es muy expresiva, tiene una estructura inductiva en los programas.

Definición 2.4. La sintaxis de PDL se basa en dos conjuntos de símbolos, un conjunto $\mathbb{P}$ contable de símbolos proposicionales atómicos y un conjunto Σ contable de programas atómicos a:

$$\phi ::= [\alpha]\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathtt{p} \mid \neg\phi \mid \top$$
$$\alpha ::= a \mid \alpha;\alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \phi?$$

En cuanto a la semántica, las formulas son interpretadas por conjuntos de estados y los programas son interpretados por relaciones binarias sobre estados en un sistema de transición. Más precisamente, la semántica de las formulas y los programas está dada por un modelo de Kripke $\mathfrak{M} = (W, \{R^a\}_{a\in\mathbb{A}}, V)$ donde:

- $xR(\alpha;\beta)y$ sii existe un estado z tal que $xR(\alpha)z$ y $zR(\beta)y$
- $xR(\alpha \cup \beta)y$ sii $xR(\alpha)y$ o $xR(\beta)y$
- $xR(\alpha^*)y$ sii existe un número no negativo n y existen estados $z_0, ..., z_n$ tales que $z_0 = x, z_n = y$ y para todo $k = 1..n, z_{k-1}R(\alpha)z_k$
- $xR(\phi?)y$ sii $x = y$ y $y \in V(\phi)$
- $V(\top) = W$
- $V(\neg\phi) = W \setminus V(\phi)$
- $V(\phi \vee \psi) = V(\phi) \cup V(\psi)$
- $V([\alpha]\phi) = \{x$: para todos los estados y, si $xR(\alpha)y$ entonces $y \in V(\phi)\}$

Definición 2.5. Una fórmula está en NNF (Negation Normal Form) si el operador de negación ($\neg$) está aplicado sólo a variables y los otros únicos operadores permitidos son conjunción y disyunción.

Capítulo 3

Ejemplo motivador

La figura 3.1 representa una página web de comercio que permite a los clientes buscar y comprar boletos de avión dada una autenticación previa. Los agentes en este ejemplo son: CLIENT, AUTHENTICATION, FLIGHT, PAYMENT y FILE. Se pueden diferenciar dos clases de agentes: por un lado un agente representa al cliente y atiende a su pedido de búsqueda y compra del boleto de avión (CLIENT) y por el otro lado, agentes que responden a este pedido y realizan las tareas necesarias para llevarlo a cabo AUTHENTICATION, FLIGHT, PAYMENT y FILE.

Los servidores se comunican con mensajes que toman valores sobre un conjunto de términos posiblemente infinito. Esta comunicación se visualiza en la Figura 3.1 gracias a los operadores '?', '!', donde '?' (resp. '!') significa recibir un mensaje (resp. enviar un mensaje). Mostramos antes de seguir un ejemplo de esta comunicación. Hacemos notar que el sistema de comunicación de los servidores es por paso de mensajes y con sincronización. El cliente inicialmente desea logearse en el sistema, ingresa su nombre de usuario (Id) y su contraseña (Pwd). El servidor CLIENT manda un mensaje *login(Id,Pwd)*, podemos verlo en la figura 3.1 como *!login(Id,Pwd)*. Este mensaje es recibido por el servidor AUTHENTICATION, que se encarga de verificar la identidad del cliente dentro del sistema, esto podemos verlo ya que es el único servidor con una transición *?login(Id,Pwd)*. En forma similar el segundo mensaje enviado por CLIENT es *!flight(Id,from,to,dpt,ret)* que no es una comunicación con el servidor AUTHENTICATION sino con el agente encargado de realizar las búsquedas. Vemos de esta forma que lo distintos agentes interactúan entre si mediante el intercambio de estos mensajes.

Ahora nos detenemos en explicar a que nos referimos cuando hablamos de mensajes que toman valores sobre un conjunto de términos posiblemente infinito. Considérese el conjunto de todos los parámetros Id. Este conjunto puede ser arbitrariamente grande y es intuitivo abstraerlo como un conjunto infinito. Como nuestros agentes intercambian mensajes sobre conjuntos infinitos elegimos razonar sobre ellos con PPDL, un lenguaje que como discutimos en la siguiente sección fue diseñado específicamente para poder manejar conjuntos de datos infinitos.

El problema consiste entonces en revisar si los servidores AUTHENTICATION, FLIGHT, PAYMENT y FILE pueden *colaborar* para satisfacer los pedidos de CLIENT en presencia de ciertas restricciones globales:

1. Cada archivo abierto debe ser cerrado.

2. La información de vuelos del cliente debe ser guardada en un archivo apropiado.

Estos requerimientos pueden ser convertidos en una fórmula PPDL dando como resultado:

$$\begin{cases} \psi_1 & = \forall x \forall y [(\mathsf{res}(x); x)^\star; \mathtt{Open}; \mathtt{y}] \big(\exists z \langle (\mathsf{res}(z); z)^\star; \mathtt{Close}; \mathtt{y} \rangle \mathtt{tt}\big), \text{ and} \\ \psi_2 & = \forall x [(\mathsf{res}(x); x)^\star; \mathtt{!}; \mathtt{pay}; \mathtt{Id}; \mathtt{Owner}; \mathtt{Nbr}; \mathtt{CSC}] \\ & \qquad \big(\exists f\ \exists z \langle (\mathsf{res}(z); z)^\star \rangle; \mathtt{Write}; \mathtt{pay}; \mathtt{Id}; \mathtt{Owner}; \mathtt{Nbr}; \mathtt{CSC}; \mathtt{f}; \mathtt{tt}\big) \end{cases}$$

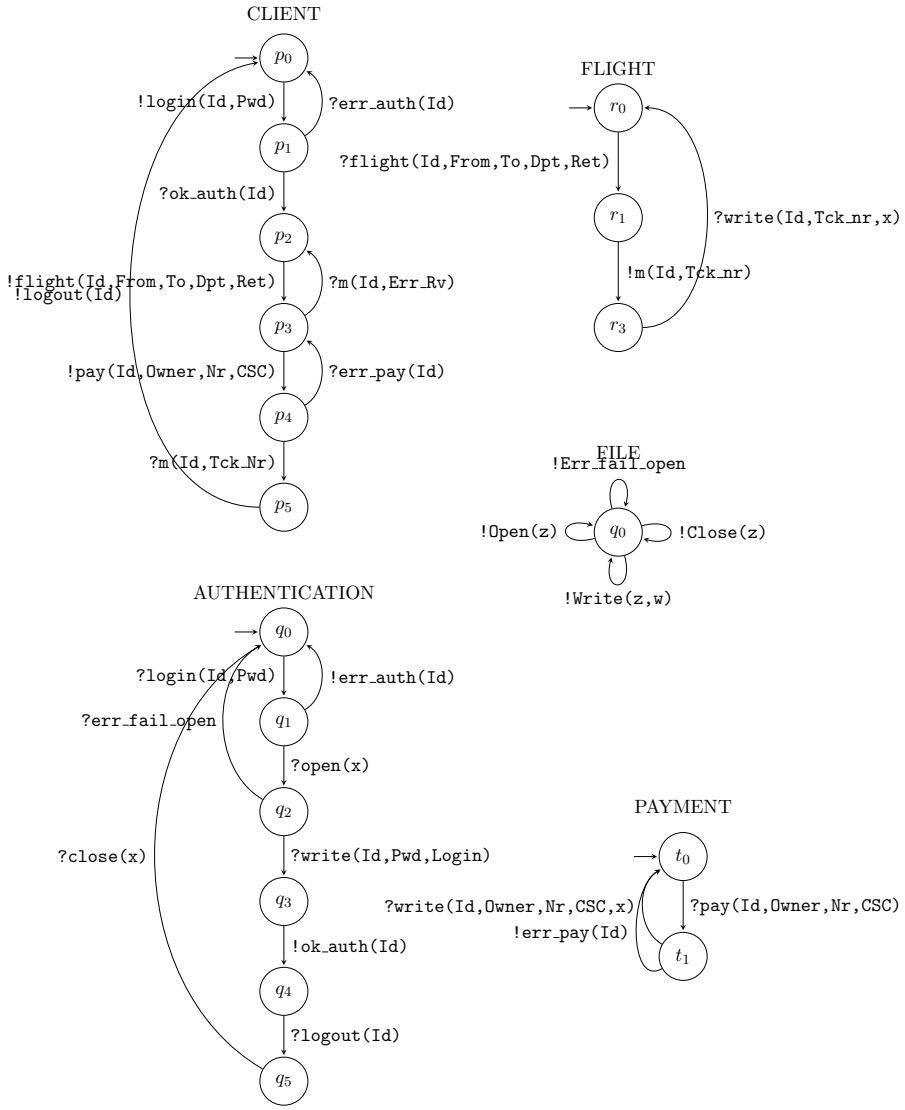


Figura 3.1: Los agentes del ejemplo

Capítulo 4

PDL Parametrizado

En este capítulo definimos la lógica proposicional dinámica parametrizada (PPDL, Parametrized Propositional Dynamic Logic). Las fórmulas de PPDL son interpretadas sobre *sistemas de transición parametrizados* cuyas aristas son etiquetadas con variables o proposiciones atómicas y cuyos estados son etiquetados con proposiciones atómicas no-parametrizadas. Primero introducimos la sintaxis de PPDL y las ideas principales detrás de esta lógica, luego introducimos los sistemas de transición parametrizados y definimos sus trazas. Finalmente, definimos la semántica de PPDL sobre sistemas de transición parametrizados.

Sintaxis de PPDL. En PDL standard los programas son expresiones regulares construidas sobre un conjunto finito de acciones atómicas usando concatenación, unión y el operador de Kleene. Para PPDL consideramos expresiones regulares *parametrizadas*. En este contexto, permitimos variables en las expresiones regulares. Estas variables toman valores sobre un conjunto infinito de acciones. Además, para poder liberar una variable que ha sido ligada a una acción, introducimos el operador de *reseteo* $\mathsf{res}(\cdot)$. Por ejemplo la expresión $x^\star$, donde x es una variable, representa todas las trazas $a^\star$ en $\Sigma^\star$, donde a es una acción atómica en Σ. Mientras que la expresión $(x; \mathsf{res}(x))^\star$ representa todas las posibles trazas finitas en $\Sigma^\star$, donde $\mathsf{res}(x)$ denota el reseteo de la variable x, i.e. trazas de la forma $a_1 a_2 \dots a_n$, donde $a_i \in \Sigma$ y $n \in \mathbb{N}$.

Sea $\mathbb{P}$ un conjunto finito de constantes proposicionales con $\top$ y $\bot$, Σ un conjunto infinito de acciones atómicas (o programas atómicos), y $\mathcal{X}$ un conjunto finito de variables que toman valores sobre Σ. La sintaxis de una fórmula PPDL está dada por la siguiente gramática:

$$\phi ::= [\alpha]\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathtt{p} \mid \forall x.\, \phi \mid \neg\phi \mid \top$$
$$\alpha ::= a \mid x \mid \alpha;\alpha \mid \alpha \cup \alpha \mid \alpha^* \mid \mathsf{res}(x) \mid \phi?$$

donde $a \in \Sigma$, $x \in \mathcal{X}$ y $\mathtt{p} \in \mathbb{P}$.

El operador Diamante $\langle\cdot\rangle$ (respectivamente, el cuantificador existencial $\exists$) puede ser definido en términos del operador Cuadrado $[\cdot]$ (respectivamente el cuantificador universal $\forall$) en la forma standard: $\langle\alpha\rangle\phi \stackrel{def}{=} \neg[\alpha]\neg\phi$ ($\exists x.\phi \stackrel{def}{=} \neg(\forall x.\neg\phi)$).

$\mathbb{L}$ es el conjunto de literales sobre $\mathbb{P}$, $\mathbb{L} = \{p, \neg p \mid p \in \mathbb{P}\}$. Para l un literal, definimos $\bar{l}$ como $\bar{l} = p$ si $l = \neg p$ y $\bar{l} = \neg p$ si $l = p$.

Para una fórmula ϕ, definimos el conjunto finito de acciones atómicas que aparecen en ϕ,

denotado por $\Sigma(\phi)$, inductivamente como sigue:

$$\begin{array}{rcl}
\Sigma([\alpha]\psi) & = & \Sigma(\alpha) \cup \Sigma(\psi) \\
\Sigma(\psi_1 \vee \psi_2) & = & \Sigma(\psi_1) \cup \Sigma(\psi_2) \\
\Sigma(\psi_1 \wedge \psi_2) & = & \Sigma(\psi_1) \cup \Sigma(\psi_2) \\
\Sigma(\exists x.\psi) & = & \Sigma(\psi) \\
\Sigma(\psi?) & = & \Sigma(\psi) \\
\Sigma(\neg\psi) & = & \Sigma(\psi) \\
\Sigma(\alpha^\star) & = & \Sigma(\alpha) \\
\Sigma(\alpha_1;\alpha_2) & = & \Sigma(\alpha_1) \cup \Sigma(\alpha_2) \\
\Sigma(\alpha_1 \cup \alpha_2) & = & \Sigma(\alpha_1) \cup \Sigma(\alpha_2) \\
\Sigma(a) & = & \{a\}, a \in \Sigma \\
\Sigma(x) & = & \emptyset, x \in \mathcal{X} \\
\Sigma(\mathtt{p}) & = & \emptyset, \mathtt{p} \in \mathbb{P} \\
\Sigma(\mathsf{res}(x)) & = & \emptyset, x \in \mathcal{X}
\end{array}$$

Para una ocurrencia de la variable x en una fórmula, sea $\tau(x) = \exists$ (respectivamente $\tau(x) = \forall$) si x está cuantificada existencialmente (universalmente).

Si λ es una fórmula o un programa, denotaremos con $\mathtt{Res}(\lambda)$ el conjunto de variables reseteadas en λ. A lo largo de este trabajo, las fórmulas son presentadas en NNF (i.e, la negación solo opera en constantes proposicionales en $\mathbb{P}$).

Denotaremos por $\phi[x := a]$ a la aplicación de la sustitución $\{x \mapsto a\}$ a ϕ. Se define inductivamente como sigue:

$$\begin{array}{rcl}
\mathtt{p}[x := a] & = & \mathtt{p}, \text{ si } \mathtt{p} \in \mathbb{P} \\
(\psi_1 \wedge \psi_2)[x := a] & = & \psi_1[x := a] \wedge \psi_2[x := a] \\
([\alpha]\psi)[x := a] & = & ([\alpha[x := a]]\psi[x := a]) \\
(\forall y.\psi)[x := a] & = & (\forall y.\psi[x := a]), \text{ si } x \neq y \\
(\forall y.\psi)[x := a] & = & \forall y.\psi, \text{ si } x = y \\
(\neg\psi)[x := a] & = & \neg(\psi[x := a]) \\
(\alpha \cup \beta)[x := a] & = & (\alpha[x := a] \cup \beta[x := a]), \\
\alpha^\star[x := a] & = & (\alpha[x := a])^\star \\
\psi?[x := a] & = & (\psi[x := a])? \\
(\mathtt{res}(y))[x := a] & = & \mathtt{res}(y) \\
(\alpha;\beta)[x := a] & = & \begin{cases} (\alpha[x := a];\beta), \text{ si } x \in \mathtt{Res}(\alpha) \\ (\alpha[x := a];\beta[x := a]), \text{ otros.} \end{cases}
\end{array}$$

$$\beta[x := a] = \begin{cases} a \text{ si } \beta = x \\ \beta \text{ si } \beta \in \Sigma \cup \mathcal{X} \text{ y } \beta \neq x \end{cases}$$

Sistemas de transición parametrizados. Las fórmulas PPDL son interpretadas sobre *sistemas de transición parametrizados* (PTS, Parametrized Transition Systems). Antes de introducirlos formalmente, explicaremos las ideas principales detrás de ellos y mostraremos que son una extensión natural de los ya conocidos sistemas de transición etiquetados. Comenzaremos recordando la definición de los sistemas de transición etiquetados.

Definición 4.1. Un sistema de transición etiquetado (LTS, Labelled Transition System) es una tupla $\langle \mathbb{P}, \Sigma, Q, q_0, \Delta, \Pi \rangle$ donde

- $(\mathbb{P}, \Sigma)$ donde Σ es el conjunto de acciones (con las que son etiquetadas las transiciones) y $\mathbb{P}$ es el conjunto de proposiciones atómicas (con las que son etiquetados los estados)
- Q es un conjunto de estados (posiblemente infinito)

- $q_0 \in Q$ es el estado inicial
- $\Delta : Q \times \Sigma \to 2^Q$ es la función de transición
- $\Pi : Q \to 2^{\mathbb{P}}$ función de etiquetado que asigna constantes proposicionales a cada estado

Los *sistemas de transición parametrizados* extienden los LTS para agregar variables y una función de reseteo de las mismas. Es decir, las transiciones en un sistema de transiciones parametrizado son etiquetadas con acciones o variables. Tenemos un número finito de variables que toman valores sobre un conjunto infinito de acciones. En una transición etiquetada con una variable x y una acción de entrada l, si x no está ligada entonces tomar la transición significa ligar x a l. Por otro lado, si x ya está ligada, entonces la transición puede ser tomada solamente si x está ligada a l. Dado que queremos reusar variables, agregamos un mecanismo adicional que liberará las variables dependiendo de los estados del autómata. Esto es, algunas variables son reseteadas en algunos estados, i.e. las variables pueden ser liberadas y así nuevas acciones pueden ser asignadas a ellas. La definición formal es la siguiente:

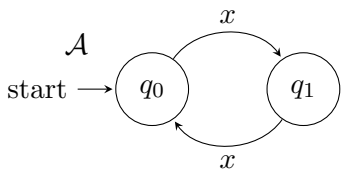
Definición 4.2. Un sistema de transición parametrizado (PTS, Parametrized Transition System) es una tupla $\langle \Sigma, \mathcal{X}, Q, q_0, \delta, \pi, \kappa \rangle$ donde:

- Σ es un conjunto infinito de acciones
- $\mathcal{X}$ es un conjunto finito de variables,
- Q es un conjunto finito de estados
- $q_0 \in Q$ es el estado inicial,
- $\delta : Q \times (\Sigma_{fin} \cup \mathcal{X}) \to 2^Q$ es una función de transición donde Σ_{fin} es un subconjunto finito de Σ,
- $\pi : Q \to 2^{\mathbb{P}}$ asigna valores de verdad a cada constante proposicional en $\mathbb{P}$ para cada estado
- $\kappa : \mathcal{X} \to 2^Q$ es la función reset que asocia a cada variable el conjunto (posiblemente vacío) de estados donde es reseteada.

Denotaremos con $\Sigma_{\mathcal{A}}$ el subconjunto finito de acciones de Σ que aparecen en el PTS $\mathcal{A}$. Para una función de reset $\kappa : \mathcal{X} \to 2^Q$, definimos la función $\kappa^{-1} : Q \to 2^{\mathcal{X}}$ como $\kappa^{-1}(q) = \{x \mid q \in \kappa(x)\}$. La definición formal de configuración y traza para PTSs es la siguiente.

Definición 4.3. Sea $\mathcal{A} = \langle \Sigma, \mathcal{X}, Q, q_0, \delta, \pi, \kappa \rangle$ un PTS. Una *configuración* es un par (q, γ) donde $q \in Q$ y γ es una sustitución. Definimos una relación de transición sobre las configuraciones como sigue: $(q_1, \gamma_1) \xrightarrow{a} (q_2, \gamma_2)$, donde $a \in \Sigma$, sii existe una sustitución σ tal que $dom(\sigma) \cap dom(\gamma_1) = \emptyset$ y existe una etiqueta $\alpha \in \Sigma \cup \mathcal{X}$ tal que $q_2 \in \delta(q_1, \alpha)$, $(\gamma_1 \uplus \sigma)(\alpha) = a$ y $\gamma_2 = (\gamma_1 \uplus \sigma)_{|D}$, con $D = Dom(\gamma_1 \uplus \sigma) \setminus \kappa^{-1}(q_2)$. Una traza de PTS es una secuencia $a_1 a_2 \dots a_n$ tal que existen estados q_i y sustituciones σ_i, $i = 1, \dots, n$ tal que $(q_0, \emptyset) \xrightarrow{a_1} (q_1, \sigma_1) \dots \xrightarrow{a_n} (q_n, \sigma_n)$.

Ejemplo 1. Sea $\mathcal{A}$ el PTS mostrado a continuación donde la variable x es reseteada en q_0.



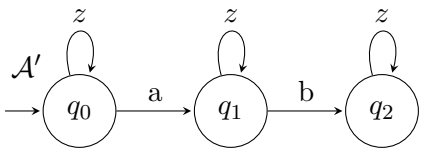
El comportamiento de $\mathcal{A}$ es el siguiente. Comenzando en el estado inicial q_0:

- Realiza la transición $q_0 \to q_1$ tomando una acción y ligando la variable x a ella, luego entra en el estado q_1,
- Realiza la transición $q_1 \to q_0$ tomando la acción asignada a x, luego entra en el estado q_0,
- Desde el estado q_0, resetea la variable x, que ya no estará ligada al símbolo de entrada. Comenzar nuevamente.

Ilustramos la corrida de $\mathcal{A}$ en la traza $w = aabb$, comenzando desde la configuración inicial $(\emptyset, q_0)$ como sigue:

$$(\emptyset, q_0) \xrightarrow{a} (\{x \mapsto a\}, q_1) \xrightarrow{a} (\emptyset, q_0) \xrightarrow{b} (\{x \mapsto b\}, q_1) \xrightarrow{b} (\emptyset, q_0)$$

Sea $\mathcal{A}'$ el PTS mostrado a continuación donde la variable z es reseteada en q_0 y q_2.



Ilustramos posibles corridas de $\mathcal{A}'$ en la traza $w = aabb$, comenzando desde la configuración inicial $(\emptyset, q_0)$ como sigue:

$$(\emptyset, q_0) \xrightarrow{a} (\emptyset, q_0) \xrightarrow{a} (\emptyset, q_1) \xrightarrow{b} (\emptyset, q_2) \xrightarrow{b} (\emptyset, q_2)$$

$$(\emptyset, q_0) \xrightarrow{a} (\emptyset, q_1) \xrightarrow{a} (\{z \mapsto a\}, q_1) \xrightarrow{b} (\emptyset, q_2) \xrightarrow{b} (\emptyset, q_2)$$

$$(\emptyset, q_0) \xrightarrow{a} (\emptyset, q_0) \xrightarrow{a} (\emptyset, q_1) \xrightarrow{b} (\{z \mapsto b\}, q_1) \xrightarrow{b} (\emptyset, q_2)$$

La instanciación de un PTS consiste en instanciar sus variables con todas las posibles acciones en Σ, produciendo un sistema con posiblemente infinitos estados y transiciones. Para esta definición debemos tener presente definición de $\zeta_{\mathcal{X},\Sigma}$ dada en el capítulo 2.

Definición 4.4 (Instanciacion de un PTS)**.** Sea $\mathcal{A} = \langle \Sigma, \mathcal{X}, Q, q_0, \delta, \pi, \kappa \rangle$ un PTS. La *instanciación* de $\mathcal{A}$, denotada por $\mathcal{C}(\mathcal{A})$, es el LTS $(\mathbb{P}, \Sigma, S, s_0, \Delta, \Pi)$, donde:

$$\begin{aligned} S &= Q \times \zeta_{\mathcal{X},\Sigma}, \\ s_0 &= (q_0, \emptyset), \\ (q', \sigma') \in \Delta(a, (q, \sigma)) &\text{ sii } (q, \sigma) \xrightarrow{a} (q', \sigma'), \text{ y} \\ \Pi((q, \sigma)) &= \pi(q), \quad \text{para toda } \sigma \in \xi_{\mathcal{X},\Sigma} \text{ y } q \in Q. \end{aligned}$$

Semántica de PPDL sobre sistemas de transición parametrizados. Las fórmulas de PPDL son interpretadas sobre *configuraciones* de sistemas de transición parametrizados, o equivalentemente sobre los estados de la instanciación de PTSs. Esto es, dada una estructura $\mathcal{A} = \langle \Sigma, \mathcal{X}, Q, q_0, \delta, \pi, \kappa \rangle$, la interpretación de una fórmula ϕ sobre $\mathcal{C}(\mathcal{A}) = (\Sigma, S, s_0, \Delta, \Pi)$, será denotada por $[\![\phi]\!]_{\mathcal{A}}$, o simplemente $[\![\phi]\!]$. Tendremos que $[\![\phi]\!] \subseteq S$ si ϕ es una fórmula y que $[\![\alpha]\!] \subseteq S \times S$ si α es un programa. Formalmente $[\![.]\!]$ se define como:

$$
\begin{aligned}
[\![\bot]\!] &= \emptyset \\
[\![\mathtt{p}]\!] &= \Pi^{-1}(\mathtt{p}), \text{ donde } \mathtt{p} \in \mathbb{P} \\
[\![\psi_1 \wedge \psi_2]\!] &= [\![\psi_1]\!] \cap [\![\psi_2]\!] \\
[\![\psi_1 \vee \psi_2]\!] &= [\![\psi_1]\!] \cup [\![\psi_2]\!] \\
[\![[\alpha]\psi]\!] &= \{s \mid \forall s' \text{ si } (s,s') \in [\![\alpha]\!] \text{ luego } \\
&\qquad s' \in [\![\psi]\!]\}, \text{ si } \alpha \notin \mathcal{X} \\
[\![[\alpha(x)]\psi]\!] &= [\![\forall x.[\alpha(x)]\psi]\!], \text{ si } x \in \mathcal{X} \\
[\![\forall x.\psi]\!] &= \bigcup_{a \in \Sigma} [\![\psi[x := a]]\!] \\
[\![\neg\psi]\!] &= S \setminus [\![\psi]\!]
\end{aligned}
\qquad
\begin{aligned}
[\![\alpha;\beta]\!] &= [\![\alpha]\!] \circ [\![\beta]\!] \\
[\![\alpha \cup \beta]\!] &= [\![\alpha]\!] \cup [\![\beta]\!] \\
[\![\alpha^\star]\!] &= \bigcup_{n \geq 0} [\![\alpha]\!]^n \\
[\![\psi?]\!] &= \{(s,s) \mid s \in [\![\psi]\!]\}, \\
[\![\mathsf{res}(x)]\!] &= \mathtt{Id} \\
[\![a]\!] &= \{(s,s') \mid s' \in \Delta(a,s)\}, \\
&\qquad \text{si } a \in \Sigma,
\end{aligned}
$$

Una fórmula ϕ es *satisfactible* si existe un PTS $\mathcal{A} = \langle \Sigma, \mathcal{X}, Q, q_0, \delta, \pi, \kappa \rangle$ y un estado $s \in Q$, tal que $s \in [\![\phi]\!]_{\mathcal{A}}$. La fórmula ϕ se dice valida, denotada $\models \phi$, si es verdadera en todos los estados de todos los sistemas de transición parametrizados. Notar que $\not\models \phi$ sii $\neg\phi$ es satisfactible.

Ejemplo 2. Damos algunas fórmulas en PPDL para ilustrar la combinación de cuantificadores con modalidades.

Sean $\phi_1 = \forall x.[x]\phi'_1$, $\phi_2 = \exists x.[x]\phi'_2$, $\phi_3 = \forall x.\langle x\rangle\phi'_3$, y $\phi_4 = \exists x.\langle x\rangle\phi'_4$. donde ϕ'_i son fórmulas de PPDL.

La fórmula ϕ_1 vale en un estado q sii para toda instanciación de la variable x, digamos con una acción $a \in \Sigma$, cada transición saliente desde q y etiquetada con a produce un estado donde $\phi'_1[x := a]$ vale.

La fórmula ϕ_2 vale en un estado q sii existe una instanciación de la variable x, digamos con una acción $a \in \Sigma$, tal que cada transición saliente desde q y etiquetada con a produce un estado donde $\phi'_2[x := a]$ vale.

La fórmula ϕ_3 vale en un estado q sii para toda instanciación de la variable x, digamos con una acción $a \in \Sigma$, existe una transición saliente desde q y etiquetada con a que produce un estado donde $\phi'_3[x := a]$ vale.

La fórmula ϕ_4 vale en un estado q sii existe una instanciación de la variable x, digamos con una acción $a \in \Sigma$, existe una transición saliente desde q y etiquetada con a que produce un estado donde $\phi'_4[x := a]$ vale.

□

Capítulo 5

Juegos de satisfactibilidad

Comenzaremos el capítulo con algunas intuiciones sobre juegos de satisfactibilidad de dos jugadores para pasar luego a algunas definiciones básicas. Por último, estudiaremos los juegos de satisfactibilidad para PPDL.

Un juego de dos jugadores tiene lugar entre el jugador $\exists$ (llamado Eloise) y el jugador $\forall$ (llamado Abelard). Los jugadores parten de una posición inicial y van generando nuevas posiciones a través de las elecciones que van tomando. Las elecciones permitidas están dentro de un marco de reglas bien definidas. El juego se termina cuando se cumple la condición de triunfo definida para alguno de los dos jugadores.

El rol de Eloise es el de una verificadora, es decir "Quiero mostrar que el conjunto inicial de fórmulas es satisfactible". Mientras que el rol de Abelard es el de un refutador es decir "Quiero mostrar que el conjunto inicial de fórmulas es insatisfactible". El conjunto de formulas a satisfacer se denotará por Γ. Escribiremos Φ, Γ para denotar que tanto Φ como las fórmulas en Γ deben satisfacerse.

Por ejemplo, en una posición $\Phi_0 \vee \Phi_1, \Gamma$ Eloise elije Φ_i y el juego continua desde la posición Φ_i, Γ. Y por otro lado, en una posición $\Phi_0 \wedge \Phi_1, \Gamma$ Abelard elije Φ_i y el juego continua desde la posición $\Phi_i, \Phi_{1-i}, \Gamma$. Podemos pensar las elecciones "o" como $\exists$-elecciones para Eloise y pensar las elecciones "y" como $\forall$-elecciones para Abelard. La idea es que Eloise (Abelard) tiene una estrategia ganadora sii el conjunto inicial de fórmulas es satisfactible (insatisfactible).

Necesitamos obligar a Abelard a tomar decisiones para que el juego avance y para esto introducimos un componente que se llama "foco". En una posición dada, una fórmula se encuentra en foco. Escribimos $[\Phi], \Gamma$ para representar la posición Φ, Γ donde Φ está en foco. Abelard elige qué fórmula esta en foco. En el ejemplo antes mencionado de una posición $\Phi_0 \wedge \Phi_1, \Gamma$ Abelard elije Φ_i y el juego continua desde la posición $[\Phi_i], \Phi_{1-i}, \Gamma$. Durante un juego él tiene permitido cambiar el foco de una fórmula a otra.

Adaptamos las notaciones usadas en [Lange and Stirling, 2001] para nuestros propósitos.

Definición 5.1. Un *juego de dos jugadores* es una tupla $\langle \mathrm{Pos}_E, \mathrm{Pos}_A, M, p^\star \rangle$, donde $\mathrm{Pos}_E, \mathrm{Pos}_A$ son conjuntos disjuntos de posiciones: las posiciones de Eloise (jugador $\exists$) y las posiciones de Abelard (jugador $\forall$). $M \subseteq (\mathrm{Pos}_E \cup \mathrm{Pos}_A) \times (\mathrm{Pos}_E \cup \mathrm{Pos}_A)$ es un conjunto de *movidas permitidas*, y $p^\star$ es la posición inicial.

Un juego de satisfactibilidad $\mathcal{G}_S(\phi)$ sobre una fórmula PPDL donde las variables son instanciadas con valores del conjunto de acciones $S \subseteq \Sigma$, es un juego posiblemente infinito entre los jugadores Abelard y Eloise. Recordar que asumimos que ϕ está en NNF.

Las posiciones del juego son configuraciones (i.e, un par compuesto de un estado y una sustitución). El nombre de las reglas está indicado a la izquierda de la regla de inferencia; y el nombre del jugador está indicado a la derecha. Para analizar satisfactibilidad de una fórmula, consideramos todas sus variables libres como existencialmente cuantificadas. A continuación, las reglas.

$$\mathtt{R}_{\wedge}\colon \frac{\Big[(\phi_0 \wedge \phi_1, \sigma)\Big], \Gamma}{\Big[(\phi_i, \sigma)\Big], (\phi_{1-i}, \sigma), \Gamma}\ \forall$$

Si una conjunción $\phi_0 \wedge \phi_1$ está en foco, Abelard descompone la posición en sus términos de la conjunción y asigna el foco, arbitrariamente a uno de ellos. Ambos términos de la conjunción quedan en el conjunto de fórmulas a satisfacer, y el segundo conyunto (el que no es puesto en foco ahora) podrá ser puesto en foco en un futuro usando la regla de cambio de foco FC explicada más adelante.

$$\mathtt{R}_{\vee}\colon \frac{\Big[(\phi_0 \vee \phi_1, \sigma)\Big], \Gamma}{\Big[(\phi_i, \sigma)\Big], \Gamma}\ \exists$$

Si una disyunción $\phi_0 \vee \phi_1$ está en foco, Eloise asigna el foco arbitrariamente a uno de sus disyuntos. Solo el disyunto elegido para estar en foco queda en el conjunto de fórmulas a satisfacer, ya que ella sólo necesita que uno de los disyuntos se satisfaga. Esta técnica de "descarte" por parte de Eloise se repite en otras reglas bajo la misma idea.

Las reglas para las fórmulas empezando con modalidades son las siguientes:

$$\mathtt{R}_{1}\colon \frac{\Big[(\langle \alpha_0 \cup \alpha_1 \rangle \phi, \sigma)\Big], \Gamma}{\Big[(\langle \alpha_i \rangle \phi, \sigma)\Big], \Gamma}\ \exists$$

Si una fórmula con una unión en un Diamante $\langle \alpha_0 \cup \alpha_1 \rangle$ está en foco, Eloise elije arbitrariamente entre α_0 y α_1, descartando el otro del Diamante. Queda en foco la fórmula original "achicada".

$$\mathtt{R}_{2}\colon \frac{\Big[([\alpha_0 \cup \alpha_1] \phi, \sigma)\Big], \Gamma}{\Big[([\alpha_i] \phi, \sigma)\Big], ([\alpha_{1-i}] \phi, \sigma), \Gamma}\ \forall$$

Si una fórmula con una unión en un Cuadrado $[\alpha_0 \cup \alpha_1]$ está en foco, Abelard descompone creando dos nuevas fórmulas a satisfacer $[\alpha_i]\phi$ y $[\alpha_{1-i}]\phi$, y asigna el foco, arbitrariamente a una de ellas. Ambas formulas nuevas quedan en el conjunto de fórmulas a satisfacer, y la segunda (la que no es puesta en foco ahora) podrá ser puesta en foco en un futuro usando la regla de cambio de foco FC explicada más adelante.

$$\mathtt{R}_{3}\colon \frac{\Big[(\langle \alpha_0 ; \alpha_1 \rangle \phi, \sigma)\Big], \Gamma}{\Big[(\langle \alpha_0 \rangle \langle \alpha_1 \rangle \phi, \sigma)\Big], \Gamma}\ \forall$$

Si una fórmula con una concatenación en un Diamante $\langle \alpha_0 ; \alpha_1 \rangle$ está en foco, Abelard descompone la posición en el Diamante del primer subtérmino seguido del Diamante del segundo subtermino. El foco se mantiene en la fórmula total.

$$\mathtt{R}_4\colon \frac{\Big[([\alpha_0;\alpha_1]\phi,\sigma)\Big],\Gamma}{\Big[([\alpha_0][\alpha_1]\phi,\sigma)\Big],\Gamma}\,\forall$$

Análogo al caso anterior, si una fórmula con una concatenación en un Cuadrado $[\alpha_0;\alpha_1]$ está en foco, Abelard descompone la posición en el Cuadrado del primer subtérmino seguido del Cuadrado del segundo subtermino. El foco se mantiene en la fórmula total.

$$\mathtt{R}_5\colon \frac{\Big[(\langle\alpha^\star\rangle\phi,\sigma)\Big],\Gamma}{\Big[(\phi,\sigma)\vee(\langle\alpha\rangle\langle\alpha^\star\rangle\phi,\sigma)\Big],\Gamma}\,\forall$$

Si una fórmula con una repetición en un Diamante $\langle\alpha^\star\rangle$ está en foco, Abelard descompone la posición una disyunción entre la fórmula ϕ y la fórmula $\langle\alpha\rangle\langle\alpha^\star\rangle\phi$ (esta última siendo equivalente a $\langle\alpha^+\rangle\phi$). El foco se mantiene en la fórmula total.

$$\mathtt{R}_6\colon \frac{\Big[([\alpha^\star]\phi,\sigma)\Big],\Gamma}{\Big[(\phi,\sigma)\wedge([\alpha][\alpha^\star]\phi,\sigma)\Big],\Gamma}\,\forall$$

Si una fórmula con una repetición en un Cuadrado $[\alpha^\star]$ está en foco, Abelard descompone la posición una conjunción entre la fórmula ϕ y la fórmula $[\alpha][\alpha^\star]\phi$ (esta última siendo equivalente a $[\alpha^+]$). El foco se mantiene en la fórmula total.

Las reglas para el operador de reseteo son las siguientes:

$$\mathtt{R}^2_{\mathtt{r}}\colon \frac{\Big[(\langle\mathsf{res}(x);\alpha\rangle\phi,\sigma)\Big],\Gamma}{\Big[(\langle\alpha\rangle\phi,\sigma_{|Dom(\sigma)\setminus\{x\}}\Big],\Gamma}\,\forall$$

Si una fórmula con una concatenación dentro de un Diamante, con el primer subtermino un reseteo $\langle\mathsf{res}(x);\alpha\rangle$ está en foco, Abelard elimina de σ el valor de la variable x y asigna el foco a $\langle\alpha\rangle\phi$ con la nueva sustitución.

$$\mathtt{R}^1_{\mathtt{r}}\colon \frac{\Big[([\mathsf{res}(x);\alpha]\phi,\sigma)\Big],\Gamma}{\Big[([\alpha]\phi,\sigma_{|Dom(\sigma)\setminus\{x\}}\Big],\Gamma}\,\forall$$

Análogo al caso anterior, si una fórmula con una concatenación dentro de un Diamante, con el primer subtermino un reseteo $[\mathsf{res}(x);\alpha]$ está en foco, Abelard elimina de σ el valor de la variable x y asigna el foco a $[\alpha]\phi$ con la nueva sustitución.

Las reglas para el operador de prueba son las siguientes (recordar que el operador de prueba $\alpha?$ puede entenderse como "proceder si α es cierto, en caso contrario falle"):

$$\mathtt{R}^1_{?}\colon \frac{\Big[\langle\psi?\rangle\phi,\sigma)\Big],\Gamma}{(\psi,\sigma),\Big[(\phi,\sigma)\Big],\Gamma}\,\forall$$

Si una fórmula con un operador de prueba dentro de un Diamante $\langle\psi?\rangle$ está en foco, Abelard descompone la fórmula en ψ y ϕ, quedando ambas en el conjunto de fórmulas a satisfacer y asignando el foco a ϕ.

$$\mathtt{R}^2_?\colon \frac{\Big[[\psi?]\phi, \sigma)\Big], \Gamma}{\Big[(\neg\psi, \sigma) \vee (\phi, \sigma)\Big], \Gamma}\ \forall$$

Si una fórmula con un operador de prueba dentro de un Cuadrado $[\psi?]$ está en foco, Abelard transforma la fórmula en una disyunción de $\neg\psi$ y ϕ, quedando en foco esta disyunción.

Las reglas para las variables que aparecen en una modalidad:

$$\mathtt{R}^1_x\colon \frac{\Big[([x]\phi, \sigma)\Big], \Gamma \qquad \text{si } \tau(x) = \forall}{\Big[([x]\phi, \sigma \uplus \{x \mapsto a\}\Big], \Gamma}\ \forall$$

Si una fórmula con una variable cuantificada universalmente dentro de una modalidad $[x]\phi$ está en foco, Abelard puede elegir cualquier $a \in S$ y asignarlo a x, y agregar esto a la sustitución σ. El foco se asigna a la fórmula original con la nueva sustitución.

$$\mathtt{R}^2_x\colon \frac{\Big[([x]\phi, \sigma)\Big], \Gamma \qquad \text{si } \tau(x) = \exists}{\Big[([x]\phi, \sigma \uplus \{x \mapsto a\}\Big], \Gamma}\ \exists$$

Si una fórmula con una variable cuantificada existencialmente dentro de una modalidad $[x]\phi$ está en foco, Eloise puede elegir cualquier $a \in S$ y asignarlo a x, y agregar esto a la sustitución σ. El foco se asigna a la fórmula original con la nueva sustitución.

Las reglas para las fórmulas cuantificadas universalmente y existencialmente son las siguientes:

$$\mathtt{R}_\forall\colon \frac{\Big[(\forall x\, \phi, \sigma)\Big], \Gamma}{\Big[(\phi, \sigma_{|Dom(\sigma)\setminus\{x\}} \uplus [x \mapsto a])\Big], \Gamma}\ \forall$$

Si una fórmula con una variable cuantificada universalmente $\forall x\, \phi$ está en foco, Abelard puede elegir cualquier $a \in S$ y asignarlo a x, y agregar esto a la sustitución $\sigma_{|Dom(\sigma)\setminus\{x\}}$. El foco se asigna a ϕ con la nueva sustitución.

$$\mathtt{R}_\exists\colon \frac{\Big[(\exists x\, \phi, \sigma)\Big], \Gamma}{\Big[(\phi, \sigma_{|Dom(\sigma)\setminus\{x\}} \uplus [x \mapsto a])\Big], \Gamma}\ \exists$$

Si una fórmula con una variable cuantificada existencialmente $\exists x\, \phi$ está en foco, Eloise puede elegir cualquier $a \in S$ y asignarlo a x, y agregar esto a la sustitución $\sigma_{|Dom(\sigma)\setminus\{x\}}$. El foco se asigna a ϕ con la nueva sustitución.

Notar que la diferencia entre las dos últimas reglas está en quien elije a y en la suposición de el jugador que pueda elegir, eligirá a para su conveniencia. Por ejemplo, si Abelard elije, el a estará elegido de tal forma que "complique" la satisfactibilidad del conjunto de fórmulas.

La aplicación sucesiva de las reglas arriba definidas produce una configuración en la cual todas las fórmulas son o bien constantes proposicionales o bien de la forma $(\langle\alpha\rangle\phi, \sigma)$ o $([\alpha]\phi, \sigma)$ donde $\alpha \in \Sigma \cup \mathcal{X}$ y $\sigma(\alpha) \in \Sigma$. Mostramos a continuación reglas para estos casos.

$$\mathrm{X}_1\colon \frac{\Big[(\langle\alpha_1\rangle\phi_1,\sigma_1)\Big],\ldots,(\langle\alpha_n\rangle\phi_n,\sigma_n),([\beta_1]\psi_1,\gamma_1),\ldots,([\beta_m]\psi_m,\gamma_m),\ldots,p_1,\ldots,p_l}{\Big[(\phi_1,\sigma_1)\Big],(\psi_{j_1},\gamma_{j_1}),\ldots,(\psi_{j_q},\gamma_{j_q})}\ \exists$$

donde $\alpha_i,\beta_i \in \Sigma \cup \mathcal{X}$ y $\forall i = 1,\ldots,q : \sigma_1(\alpha_1) = \gamma_{j_i}(\beta_{j_i}), j_i \in \{1,\ldots,m\}$ y $\sigma_l(\alpha_l),\gamma_{l'}(\beta_{l'}) \in \Sigma$ para todo l, l'.

Como la fórmula en foco es un Diamante $\langle\alpha_1\rangle\phi_1$, se pasa a considerar la satisfactibilidad de ϕ_1, con las restricciones de todas las formulas Cuadrado que "coincidan" con $\sigma_1(\alpha_1)$.

$$\mathrm{X}_2\colon \frac{(\langle\alpha_1\rangle\phi_1,\sigma_1),\ldots,(\langle\alpha_n\rangle\phi_n,\sigma_n),\Big[([\beta_1]\psi_1,\gamma_1)\Big],\ldots,([\beta_m]\psi_m,\gamma_m),\ldots,p_1,\ldots,p_l}{(\phi_k,\sigma_k),\Big[(\psi_{j_1},\gamma_{j_1})\Big],\ldots,(\psi_{j_q},\gamma_{j_q})}\ \forall$$

donde $\alpha_i,\beta_i \in \Sigma \cup \mathcal{X}$ y $\forall i = 1,\ldots,q : \sigma_k(\alpha_k) = \gamma_{j_i}(\beta_{j_i}), j_i \in \{1,\ldots,m\}$, y $\sigma_l(\alpha_l),\gamma_{l'}(\beta_{l'}) \in \Sigma$ para todo l, l'.

Como la fórmula en foco es un Cuadrado $[\beta_1]\psi_1$, se pasa a considerar la satisfactibilidad de ψ_1, con las restricciones de todas las formulas Cuadrado y algún Diamante que "coincida" con $\gamma_1(\beta_1)$ elegido por Abelard donde potencialmente habría un problema de satisfactibilidad.

Como mencionamos anteriormente existe también una regla que permite a Abelard cambiar el foco del juego:

$$\mathrm{FC}\colon \frac{\Big[(\phi,\sigma)\Big],(\psi,\gamma),\Gamma}{(\phi,\sigma),\Big[(\psi,\gamma)\Big],\Gamma}\ \forall$$

Abelard elige arbitrariamente otra fórmula del conjunto de fórmulas a satisfacer y le asigna el foco.

La mayor diferencia entre los juegos de satisfactibilidad para PDL standard y el que acabamos de definir es que nuestros juegos tienen (posiblemente) un número infinito de posiciones e infinitas ramas. Esto se debe al hecho de que el tamaño de nuestras configuraciones no esta acotado. Es necesario tener esto en consideración al definir las condiciones para que uno de los jugadores gane un juego de satisfactibilidad dado.

Las condiciones de triunfo tienen que tener en cuenta el hecho de que el constructor de menor punto fijo asociado a un operador $.^\star$ es finalmente satisfecho y no es postergado infinitamente.

Definición 5.2. Abelard gana el juego (posiblemente infinito) $\pi = C_0,\ldots,C_n,\ldots$ sii

1. $C_m = \Big[(l,\sigma)\Big],\Gamma$ y ($l = \bot$ or $\bar{l} \in \Gamma$), para algún m, o
2. La fórmula $\langle\alpha^\star\rangle\phi$ aparece infinitamente bajo el foco en π y Abelard ha aplicado la regla del foco (FC) sólo un número finito de veces. Esto es, existe una secuencia infinita $i_1, i_2,\ldots$ de enteros tal que $C_{i_j} = \Big[(\langle\alpha^\star\rangle\phi,\sigma_{i_j})\Big],\Gamma_{i_j}$ para todo $j = 1,2,\ldots$ y existe algún i_k tal que no se aplica la regla del foco (FC) desde las configuraciones C_m para todo $m \geq i_k$.

Definición 5.3. Eloise gana el (posiblemente infinito) juego $\pi = C_0,\ldots,C_n,\ldots$ si

3. $C_n = \Big[(q_1,\sigma_1)\Big],\ldots,(q_k,\sigma_k)$ y $\{q_1,\ldots,q_k\}$ es satisfactible, cuando q_i son literales y es una valuación consistente, o
4. La fórmula $[\alpha^\star]\phi$ aparece un número infinito de veces en π bajo el foco, esto es, existe una secuencia infinita $i_1, i_2,\ldots$ de enteros tal que $C_{i_j} = \Big[([\alpha^\star]\phi,\sigma_{i_j})\Big],\Gamma_{i_j}$ aparece en π, o

5. La fórmula ϕ aparece un número infinito de veces en π bajo el foco y Abelard ha aplicado la regla del foco (FC) infinitamente.

Puede mostrarse que estas condiciones de victoria son mutuamente excluyentes, por lo tanto:

Lema 1. *El juego $\mathcal{G}_S(\phi)$ tiene un único ganador, donde $S \subseteq \Sigma$.*

Demostración. El único caso no trivial es cuando S es un conjunto infinito. Sostenemos que las condiciones de victoria son mutuamente excluyentes. Sólo discutimos el caso de un juego infinito en el cual tanto $\psi_1 = \left[\langle\alpha_1^\star\rangle\phi_1\right]$ como $\psi_2 = \left[[\alpha_2^\star]\phi_2\right]$ aparecen infinitamente seguido bajo el foco. En este caso la regla (FC) ha sido aplicada entre ψ_1 y ψ_2 al menos una vez. Por lo tanto, es aplicada infinitamente seguido en el juego. □

Definición 5.4. Una estrategia para la jugadora Eloise es una función $\rho : \mathrm{Pos}_E \to \mathrm{Pos}_E \cup Pos_A$, tal que $(\wp, \rho(\wp)) \in M$ para todas $\wp \in \mathrm{Pos}_E$. Un juego (posiblemente infinito) $\pi = \langle\wp_1, \wp_2, \ldots\rangle$ *sigue* una estrategia ρ para la jugadora Eloise sii $\wp_{i+1} = \rho(\wp_i)$ para todas $i \in \mathbb{N}$ tal que $\wp_i \in \mathrm{Pos}_E$. Sea $\mathcal{W}$ un conjunto de juegos (posiblemente infinito). Una estrategia ρ es *ganadora* para Eloise desde un conjunto $S \subseteq \mathrm{Pos}_E \cup \mathrm{Pos}_A$ de acuerdo con $\mathcal{W}$ sii todo juego que comienza en una posición en S y siguiendo ρ satisface la condición de triunfo para Eloise.

La caracterización teórica del juego de PPDL es correcta y completa.

Teorema 2. *Lo siguiente vale:*

(Correctitud). Si Eloise gana el juego $\mathcal{G}(\Phi_0)$ entonces Φ_0 es satisfactible.

(Completitud). Si Φ_0 es satisfactible entonces Eloise gana el juego $\mathcal{G}(\Phi_0)$.

Demostración. Asumimos que usa la siguiente estrategia "monótona": mantiene un conjunto de listas de todas las fórmulas de la forma $\langle\alpha^\star\rangle\phi$, todas subfórmulas de Φ_0 en orden decreciente por tamaño. Cada lista corresponde a un juego en $\mathcal{G}(\Phi_0)$.

Abelard comienza tratando de colocar el foco en alguna ψ presente en la configuración. Si no hay tal fórmula, trata con una sub-fórmula de ψ. En todos los casos, si puede encontrar una contradicción colocando el foco en una constante proposicional, gana por la condición ganadora 1. Esta es su única forma de ganar si la lista de fórmulas está vacía.

Si la fórmula bajo el foco es $[\alpha^\star]\phi$ y ϕ contiene una fórmula en la lista, coloca el foco en ϕ después de que ϕ ha sido *regenerada*. □

La prueba de completitud se basa en el lema siguiente:

Lema 3. *Tenemos que $\phi \wedge \langle\alpha^\star\rangle\psi$ es satisfactible sii $\phi \wedge (\psi \vee \langle\alpha\rangle(\langle\alpha^\star\rangle\psi \vee \neg\phi))$ es satisfactible.*

Demostración. Usando las leyes de De Morgan para probar que
$\neg\Big(\phi \wedge (\psi \vee \langle\alpha\rangle(\langle\alpha^\star\rangle\psi \vee \neg\phi))\Big) \equiv (\phi \to (\neg\psi[\alpha]([\alpha^\star]\neg\psi \vee \phi)))$. Luego, si asumimos que:

1. $\phi \wedge \langle\alpha\rangle\psi$, tiene un modelo y
2. $\models \phi \to (\neg\psi[\alpha]([\alpha^\star]\neg\psi \vee \phi))$ vale,

llegamos a una contradicción. □

Capítulo 6

Decidibilidad de PPDL

La idea de la prueba de la decidibilidad de PPDL consiste en reducir el problema de satisfabilidad de una fórmula en PPDL al problema de satisfabilidad de la misma fórmula en la cual las acciones toman valores en un conjunto de acciones *finito*. Resulta que resolver el juego resultante es decidible gracias a que es un juego finito de dos Jugadores con condiciones de triunfo de Büchi. La construcción de este conjunto finito de acciones se da a continuación.

Definición 6.1. Sea ϕ una fórmula PPDL y sean $\Sigma(\phi) = \{a_1, \ldots, a_n\}$ y $\mathcal{X} = \{x_1, \ldots, x_k\}$. Definimos $\mathcal{G}_C(\phi)$ como el juego de satisfabilidad en el cual las variables son instanciadas sobre un conjunto finito de constantes:

$$C = \{a_1, \ldots, a_n, c_1, \ldots, c_k\} \tag{6.1}$$

La idea es que el juego $\mathcal{G}_C(\phi)$ es usado para simular el juego $\mathcal{G}_\Sigma(\phi)$. Antes de mostrar esto, necesitamos introducir una variante de los juegos de satisfabilidad en la cual Eloise puede, un número finito de veces, duplicar una fórmula bajo el foco.

Definición 6.2. Definimos $\mathcal{G}_S^{\mathcal{D}}$ para ser el juego de satisfabilidad en el cual agregamos la regla de duplicación para Eloise:

$$\text{DP: } \frac{\Big[(\phi, \sigma)\Big], \Gamma}{\Big[(\phi, \sigma)\Big], (\phi, \sigma), \Gamma} \exists$$

que puede ser aplicada un número finito de veces.

El resultado principal de esta sección sigue, así como la estructura de la prueba:

Teorema 4. *La satisfabilidad de PPDL es decidible.*

El Corolario 8 muestra que $\mathcal{G}_\Sigma$ y $\mathcal{G}_C$ son equivalentes. El Lema 5 muestra que $\mathcal{G}_\Sigma$ y $\mathcal{G}_\Sigma^{\mathcal{D}}$ son equivalentes. También, por Lema 5 tenemos que $\mathcal{G}_C$ y $\mathcal{G}_C^{\mathcal{D}}$ son equivalentes. Por lo tanto $\mathcal{G}_\Sigma^{\mathcal{D}}$ y $\mathcal{G}_C^{\mathcal{D}}$ son equivalentes también. La equivalencia debe ser entendida como que Eloise gana en un juego sii ella gana en el otro.

Lema 5. *Los juegos $\mathcal{G}_\Sigma$ y $\mathcal{G}_\Sigma^{\mathcal{D}}$ son equivalentes. Esto es, Eloise gana en $\mathcal{G}_\Sigma(\phi)$ sii gana en $\mathcal{G}_\Sigma^{\mathcal{D}}(\phi)$.*

Demostración. Asuma que $C_0 \to C_1 \to \ldots C_n$ es una serie de aplicaciones de la regla de duplicación $\mathcal{G}_\Sigma^{\mathcal{D}}$ y que $C_0 = \Big[(\phi, \sigma)\Big], \Gamma$. Por lo tanto, $C_n = \Big[(\phi, \sigma)\Big], (\phi, \sigma), \ldots, (\phi, \sigma), \Gamma$. Por lo tanto, necesitamos probar que Eloise gana desde C_n sii ella gana desde C_0. Pero ya que Eloise gana desde C_n sii $\mathcal{C} = (\phi, \sigma) \wedge \bigwedge(\phi, \sigma) \wedge \bigwedge \Gamma$ es satisfactible, y por otro lado $\mathcal{C}$ es satisfactible sii $(\phi, \sigma) \wedge \bigwedge \Gamma$ es satisfactible, entonces el resultado se sigue. □

De modo de poder relacionar las configuraciones de $\mathcal{G}_C(\phi)$ a las de $\mathcal{G}_\Sigma(\phi)$, definimos una relación $\lhd$ entre configuraciones de $\mathcal{G}_C(\phi)$ y configuraciones de $\mathcal{G}_\Sigma(\phi)$. Denotamos por Ψ el conjunto finito de fórmulas que aparecen en una configuración, i.e. $\Psi((\psi,\sigma)) = \{\psi\}$ y $\Psi([C_1],\ldots,C_n) = \bigcup_i \Psi(C_i)$. Primero, definimos la relación de *coherencia* entre sustituciones.

Definición 6.3. Sea C un subconjunto finito de Σ. La relación de coherencia $\bowtie_C \subseteq \zeta \times \zeta$ entre sustituciones (ver Definición 2.1) se define como sigue: $\bar{\sigma} \bowtie_C \sigma$ sii se cumplen las siguientes tres condiciones:

1. $dom(\bar{\sigma}) = dom(\sigma)$,
2. Si $\bar{\sigma}(x) \in C$ entonces $\bar{\sigma}(x) = \sigma(x)$, y si $\sigma(x) \in C$, entonces $\bar{\sigma}(x) = \sigma(x)$, para cualquier variable $x \in dom(\sigma)$, y
3. para cualquier variables $x, y \in dom(\sigma)$, $\bar{\sigma}(x) = \bar{\sigma}(y)$ sii $\sigma(x) = \sigma(y)$.

Para relacionar las configuraciones de $\mathcal{G}_\Sigma$ y $\mathcal{G}_C$, donde C es el conjunto de letras definido en la ecuación (6.1), definimos a continuación una relación binaria entre configuraciones, denotada por $\lhd$.

Definición 6.4. Sea ϕ una fórmula PPDL con $\Sigma(\phi) = \{a_1,\ldots,a_n\}$. Sea Γ (respectivamente $\widehat{\Gamma}$) una lista de configuraciones en $\mathcal{G}_\Sigma(\phi)$ (resp. $\mathcal{G}_C(\phi)$) de la forma: $\Gamma = (\psi_1,\sigma_1),\ldots,(\psi_m,\sigma_m),\cdots$, y $\widehat{\Gamma} = (\widehat{\psi}_1,\widehat{\sigma}_1),\ldots,(\widehat{\psi}_m,\widehat{\sigma}_m)$, donde ψ_i y $\widehat{\psi}_i$ son fórmulas en PPDL, y σ_i y $\widehat{\sigma}_i$ son sustituciones. Sea f una función suryectiva total del conjunto de configuraciones de $\mathcal{G}_\Sigma(\phi)$ al conjunto de configuraciones de $\mathcal{G}^{\mathcal{D}}_C(\phi)$. Definimos $\Gamma \lhd_f^{\Sigma,C} \widehat{\Gamma}$ sii se cumple lo siguiente:

1. $\Psi(\Gamma) = \Psi(\widehat{\Gamma})$.
2. Si $f((\psi_i,\sigma_i)) = (\widehat{\psi}_j,\widehat{\sigma}_j)$ entonces $\psi_i = \widehat{\psi}_j$ y $\sigma_i \bowtie_C \widehat{\sigma}_j$.
3. Si $(\widehat{\psi},\widehat{\sigma})$ está bajo el foco en $\widehat{\Gamma}$ y (ψ,σ) está bajo el foco en Γ, entonces $f((\psi,\sigma)) = (\widehat{\psi},\widehat{\sigma})$.

Además, escribimos $\mathcal{G}_\Sigma(\phi) \lhd \mathcal{G}_C(\phi)$ sii existe una función suryectiva f tal que $\left[\phi\right],\emptyset \lhd_f^{\Sigma,C} \left[\phi\right],\emptyset$.

Escribimos $\lhd_f$ en lugar de $\lhd_f^{\Sigma,C}$ si no hay ambigüedad.
Las siguientes afirmaciones pueden demostrarse fácilmente.

Afirmación 1. *Sea $C \subseteq \Sigma$ un conjunto finito de letras, $\bar{\sigma}$ y σ dos substituciones, x una variable, y a una letra en C. Las siguientes afirmaciones son ciertas:*

1. *Si $\bar{\sigma} \bowtie_C \sigma$ entonces $|codom(\bar{\sigma})| = |codom(\sigma)|$ y $\bar{\sigma}_{|D} \bowtie_C \sigma_{|D}$, donde $D \subseteq Dom(\sigma)$.*
2. *En consecuencia, si $(\bar{\sigma}_1 \uplus \bar{\sigma}_2) \bowtie (\sigma_1 \uplus \sigma_2)$ con $dom(\bar{\sigma}_i) = dom(\sigma_i)$, luego $\bar{\sigma}_i \bowtie \sigma_i$, para $i = 1,2$.*

Lema 6. *[Belkhir* et al.*, 2013] Sean $C_1 \subseteq \Sigma$ y $C_2 \subseteq \Sigma$ dos conjuntos de acciones. Sea $C = C_1 \cap C_2$ tal que $|C| > |\mathcal{X}|$. Sea a_1 una acción en C_1 y $x \in \mathcal{X}$ y sean $\sigma_i : \mathcal{X} \to C_i$, $i = 1,2$ dos substituciones donde $\sigma_1 \bowtie_C \sigma_2$. Luego, existe una función $\boldsymbol{\Theta}^{C_1,C_2}$ que satisface $\sigma_1 \uplus \{x \mapsto a_1\} \bowtie_C \boldsymbol{\Theta}(\sigma_1, x, a_1, \sigma)$.*

El siguiente Lema muestra que la regla de inferencia de la sección 5 preserva la relación $\lhd$:

Lema 7. *Sea ϕ una fórmula PPDL con un conjunto finito de acciones $C = \{a_1,\ldots,a_n,c_1,\ldots,c_k\}$ Sea Γ (respectivamente $\widehat{\Gamma}$) una lista de configuraciones en $\mathcal{G}_\Sigma(\phi)$ (respectivamente $\mathcal{G}^{\mathcal{D}}_C(\phi)$). Si $\Gamma \lhd_f \widehat{\Gamma}$ entonces*

1. *para cada* Γ' *tal que* $\Gamma \xrightarrow{I} \Gamma'$ *es una movida en* $\mathcal{G}_\Sigma(\phi)$*, existe una función total suryectiva* f' *y una lista de configuraciones* $\widehat{\Gamma}'$ *tal que* $\widehat{\Gamma} \xrightarrow{I} \widehat{\Gamma}'$ *es una movida posible en* $\mathcal{G}_C^{\mathcal{D}}(\phi)$ *y* $\Gamma' \lhd_{f'} \widehat{\Gamma}'$*, y*

2. *para cada* $\widehat{\Gamma}'$ *tal que* $\widehat{\Gamma} \xrightarrow{II} \widehat{\Gamma}'$ *es una movida en* $\mathcal{G}_C^{\mathcal{D}}(\phi)$*, existe una función suryectiva* f' *y una lista de configuraciones* Γ' *tal que* $\Gamma \xrightarrow{II} \Gamma'$ *es una movida posible en* $\mathcal{G}_\Sigma(\phi)$ *y* $\Gamma' \lhd_{f'} \widehat{\Gamma}'$*,*

Demostración. Asumamos $\mathcal{X} = \{x_1, \ldots, x_k\}$. Discutimos varios casos dependiendo de la regla aplicada.

1. Las reglas aplicadas aquí para Abelard pueden ser $\mathtt{R}_\wedge$, $\mathtt{R}_2$, $\mathtt{R}_3$, $\mathtt{R}_4$, $\mathtt{R}_5$, $\mathtt{R}_6$, $\mathtt{R}^1_{\mathtt{r}}$, $\mathtt{R}^2_{\mathtt{r}}$, $\mathtt{R}^1_x$, $\mathtt{R}^1_?$, $\mathtt{R}^2_?$, $\mathtt{R}_\forall$, X_1, X_2 y FC.
   Para la regla $\mathtt{R}_\wedge$, Sean:
$$\begin{cases} \Gamma = \Big[(\phi_0 \wedge \phi_1, \sigma)\Big], \Upsilon & \text{and} \\ \Gamma' = \Big[(\phi_i, \sigma)\Big], (\phi_{1-i}, \sigma), \Upsilon & \text{and} \\ \widehat{\Gamma} = \Big[(\phi_0 \wedge \phi_1, \widehat{\sigma})\Big], \widehat{\Upsilon} \end{cases}$$
   donde $\sigma \bowtie_{\Sigma(\phi)} \widehat{\sigma}$. En este caso sean
$$\begin{cases} \widehat{\Gamma}' \stackrel{def}{=} \Big[(\phi_i, \widehat{\sigma})\Big], (\phi_{1-i}, \widehat{\sigma}), \widehat{\Upsilon} \\ \sigma \stackrel{def}{=} \widehat{\sigma}, \text{ and} \\ f' \stackrel{def}{=} f_{|\Gamma} \cup \{\big((\phi_i, \sigma), (\phi_i, \widehat{\sigma})\big)\} \cup \{\big((\phi_{1-i}, \sigma), (\phi_{1-i}, \widehat{\sigma})\big)\} \end{cases}$$
   Luego, $\Gamma \lhd_{f'} \widehat{\Gamma}'$.
   Las reglas $\mathtt{R}_2$, $\mathtt{R}_3$, $\mathtt{R}_4$, $\mathtt{R}_5$, $\mathtt{R}_6$, $\mathtt{R}^1_?$ y $\mathtt{R}^2_?$ pueden ser manejadas similarmente.
   Para las reglas $\mathtt{R}^1_{\mathtt{r}}$,$\mathtt{R}^2_{\mathtt{r}}$ la afirmación se deduce del hecho que si $\sigma \bowtie_{\Sigma(\phi)} \widehat{\sigma}$ luego $\sigma_{|Dom(\sigma)\setminus\{x\}} \bowtie_{\Sigma(\phi)} \widehat{\sigma}_{|Dom(\widehat{\sigma})\setminus\{x\}}$, ver Ítem 1 de la Afirmación 1.
   Para la regla $\mathtt{R}_\forall$, asumamos que
$$\begin{cases} \Gamma &= \Big[(\forall x\phi, \sigma)\Big], \Lambda \quad \text{and} \\ \Gamma' &= \Big[(\phi, \sigma_{x \mapsto a})\Big], \Lambda \quad \text{and} \\ \widehat{\Gamma} &= \Big[(\forall x\phi, \widehat{\sigma})\Big], \widehat{\Lambda} \end{cases}$$
   Distinguimos dos casos:

   Caso 1. Si $\nexists C \in \Lambda$ tal que $f(C) = (\forall x\phi, \widehat{\sigma})$, luego en este caso la movida relacionada en $\widehat{\mathcal{G}}_{\mathcal{D}}$ es
$$\underbrace{\Big[(\forall x\phi, \widehat{\sigma})\Big], \widehat{\Lambda}}_{\widehat{\Gamma}} \xrightarrow{(\mathtt{R}_\forall)} \underbrace{\Big[(\phi, \widehat{\sigma}')\Big], \widehat{\Lambda}}_{\widehat{\Gamma}'}$$
   donde la sustitución $\widehat{\sigma}'$ esta definida por $\widehat{\sigma}' \stackrel{def}{=} \mathbf{\Theta}^{\Sigma,\Sigma_f}(\sigma, x, a, \widehat{\sigma})$.

   Caso 2. Si $\exists C \in \Lambda$ tal que $f(C) = (\forall x\phi, \widehat{\sigma})$, luego en este caso la movida relacionada en $\widehat{\mathcal{G}}_{\mathcal{D}}$ son
$$\underbrace{\Big[(\forall x\phi, \widehat{\sigma})\Big], \widehat{\Lambda}}_{\widehat{\Gamma}} \xrightarrow{(\mathtt{DP})} \Big[(\forall x\phi, \widehat{\sigma})\Big], (\forall x\phi, \widehat{\sigma}), \widehat{\Lambda} \xrightarrow{(\mathtt{R}_\forall)} \underbrace{\Big[(\phi, \widehat{\sigma}')\Big], (\forall x\phi, \widehat{\sigma}), \widehat{\Lambda}}_{\widehat{\Gamma}'}$$
   donde $\widehat{\sigma}'$ es definida por $\widehat{\sigma}' \stackrel{def}{=} \mathbf{\Theta}^{\Sigma,\Sigma_f}(\sigma, x, a, \widehat{\sigma})$.

Notar que en los dos casos tenemos que $\widehat{\sigma}' \bowtie_{\Sigma(\phi)} \sigma$, Lema 6. Además, en los dos casos la función f' está definida por

$$\begin{cases} f' & = f_{|Dom(f)\setminus\{(\forall x\phi,\sigma)\}}, \quad \text{and} \\ f'((\phi, \sigma_{x\mapsto a})) & = (\phi, \widehat{\sigma}') \end{cases}$$

Luego $\Gamma \lhd_{f'} \widehat{\Gamma}'$.

Por la regla X1, asumamos que

$$\begin{cases} \Gamma & = \Big[(\langle\alpha_1\rangle\phi_1, \sigma_1)\Big], \ldots, (\langle\alpha_n\rangle\phi_n, \sigma_n), ([\beta_1]\psi_1, \gamma_1), \ldots, ([\beta_m]\psi_m, \gamma_m), \ldots, p_1, \ldots, p_l \\ \text{and} & \\ \widehat{\Gamma} & = \Big[(\langle\widehat{\alpha}_1\rangle\widehat{\phi}_1, \widehat{\sigma}_1)\Big], \ldots, (\langle\widehat{\alpha}_k\rangle\widehat{\phi}_k, \widehat{\sigma}_k), ([\widehat{\beta}_1]\widehat{\psi}_1, \widehat{\gamma}_1), \ldots, ([\widehat{\beta}_r]\widehat{\psi}_r, \widehat{\gamma}_r), \ldots, p_1, \ldots, p_{l'} \end{cases}$$

Por lo tanto,

$$\begin{cases} \Gamma' & = \Big[(\phi_1, \sigma_1)\Big], (\psi_{j_1}, \gamma_{j_1}), \ldots, (\psi_{j_q}, \gamma_{j_q}) \quad \text{and} \\ \widehat{\Gamma}' & = \Big[(\widehat{\phi}_1, \widehat{\sigma}_1)\Big], (\widehat{\psi}_{j'_1}, \widehat{\gamma}_{j'_1}), \ldots, (\widehat{\psi}_{j'_p}, \widehat{\gamma}_{j'_p}) \end{cases}$$

donde, por un lado, $\alpha_i, \beta_i \in \Sigma \cup \mathcal{X}$ y $\forall i = 1, \ldots, q : \sigma_1(a_1) = \gamma_{j_i}(b_{j_i}), j_i \in \{1, \ldots, m\}$ y, por otro lado, $\widehat{\alpha}_i, \widehat{\beta}_i \in \Sigma \cup \mathcal{X}$ y $\forall i = 1, \ldots, p : \widehat{\sigma}_1(\widehat{\alpha}_1) = \widehat{\gamma}_{j'_i}(\widehat{\beta}_{j'_i}), j'_i \in \{1, \ldots, r\}$. Finalmente, sea $f' \stackrel{def}{=} f_{|\Gamma'}$. Por lo tanto, $\Gamma' \lhd_{f'} \widehat{\Gamma}'$.

Para la regla FC, asumamos que

$$\begin{cases} \Gamma & = \Big[(\phi, \sigma)\Big], (\psi, \gamma), \Upsilon \quad \text{and} \\ \Gamma' & = \Big[(\psi, \gamma)\Big], (\phi, \sigma), \Upsilon \quad \text{and} \\ \widehat{\Gamma} & = \Big[(\widehat{\phi}, \widehat{\sigma})\Big], \widehat{\Upsilon} \end{cases}$$

La idea es elegir una fórmula $(\widehat{\psi}, \widehat{\gamma})$ de Υ tal que $\widehat{\Gamma}' = \Big[(\widehat{\psi}, \widehat{\gamma})\Big], \widehat{\Upsilon}$. Definimos $(\widehat{\psi}, \widehat{\gamma}) := f((\psi, \gamma))$. Luego tenemos $\Gamma' \lhd_f \widehat{\Gamma}'$ ya que $\widehat{\psi} = \psi$ y $\widehat{\gamma} \bowtie \gamma$.

2. Las posibles reglas para Eloise son $\mathtt{R}_\exists$, X_1 y $\mathtt{R}^2_x$. La regla $\mathtt{R}_\exists$ (respectivamente $\mathtt{R}^2_x$) es exactamente como la regla $\mathtt{R}_\forall$ (respectivamente $\mathtt{R}^1_x$) sólo que Eloise es la que mueve en vez de Abelard.

□

Afirmación 2. *Notar que si*

$$\Big[(q_1, \sigma_1)\Big], (q_2, \sigma_2), \ldots, (q_n, \sigma_n) \ \lhd_f^{\Sigma, C} \ \Big[(q'_1, \sigma'_1)\Big], (q'_2, \sigma'_2), \ldots, (q'_m, \sigma'_m), \tag{6.2}$$

donde q_i, q'_i son constantes proposicionales, entonces $\{q_1, \ldots, q_n\} = \{q'_1, \ldots, q'_m\}$ y por lo tanto $\{(q_1, \sigma_1), (q_2, \sigma_2), \ldots, (q_n, \sigma_n)\}$ es satisfactible sii $\{(q'_1, \sigma'_1), (q'_2, \sigma'_2), \ldots, (q'_m, \sigma'_m)\}$ es satisfactible.

Corolario 8. *Sea ϕ una fórmula PPDL. Si $\mathcal{G}_\Sigma(\phi) \lhd \mathcal{G}_C(\phi)$ entonces Eloise gana en $\mathcal{G}_\Sigma(\phi)$ sii ella gana en $\mathcal{G}_C(\phi)$.*

Demostración. Se sigue del Lema 7 que existe un juego infinito en $\mathcal{G}_\Sigma(\phi)$ sii existe un juego infinito en $\mathcal{G}_C(\phi)$. Además, este juego es ganador para Eloise en $\mathcal{G}_\Sigma(\phi)$ por la condición ganadora i sii es ganador para ella en $\mathcal{G}_C(\phi)$ por la condición ganadora i donde $i = 3, 4, 5$. Para los juegos finitos, se deduce por la Afirmación 2 que la fórmula de una lista de configuraciones en $\mathcal{G}_\Sigma(\phi)$ son constantes proposicionales (satisfactibles) sii la fórmula de la lista de configuraciones relacionada en $\mathcal{G}_C(\phi)$ son constantes proposicionales (satisfactibles). □

Capítulo 7

Conclusión

En esta tesis hemos introducido una extensión de PDL llamada PPDL en la cual las acciones pueden ser letras o variables que toman valores sobre un dominio infinito. Podemos notar que si trabajamos sobre PTS con alfabetos *finitos*, entonces toda fórmula de PPDL es equivalente a una fórmula (quizas exponencialmente más larga) de PDL. Hemos probado que el problema de satisfactibilidad para PPDL es decidible cuando es interpretado sobre la subclase de sistemas de transición parametrizados en los cuales las variables pueden ser reseteadas.

PPDL, en combinación con PTS, resulta natural para expresar procesos de iteración. Su manejo de variables que pueden tomar valores sobre un rango infinito es útil para aplicaciones del mundo real. En [Belkhir *et al.*, 2014] mostramos que PPDL puede utilizarse para el problema de composición de servidores web en presencia de restricciones en el orden global del intercambio de mensajes entre los agentes. Expresamos la especificación del cliente y de los servidores disponibles como sistemas de transición parametrizados y expresamos las restricciones de comportamiento como una formula PPDL que el orquestador generado debe satisfacer. Resulta que el modelo de dicha formula representa el orquestador deseado.

Existen propuestas anteriores para este problema y problemas similares. Muchas extensiones de PDL fueron desarrolladas, por ejemplo con asignaciones proposicionales [Balbiani *et al.*, 2013], con operadores de intersección [Balbiani and Vakarelov, 2003], con operadores nuevos [Lutz, 2005], etc. En [Göller and Lohrey, 2006] se estudiaron problemas de model-checking de PDL, y su extensión con intersección, sobre varias clases de sistemas de estados infinitos.

Sin embargo PPDL se caracteriza por ser una extensión natural de PDL que es bien conocida y con una teoría de prueba bien desarrollada. Las pruebas en esta tesis se inspiraron en [Lange and Stirling, 2001]: ellas se basan en un enfoque teórico de juegos para formular satisfactibilidad junto con un mecanismo de *foco*, en vez de el enfoque teórico más tradicional basado en autómatas.

Bibliografía

[Balbiani and Vakarelov, 2003] Philippe Balbiani and Dimiter Vakarelov. PDL with intersection of programs: A complete axiomatization. *Journal of Applied Non-Classical Logics*, 13(3-4):231–276, 2003.

[Balbiani *et al.*, 2013] Philippe Balbiani, Andreas Herzig, and Nicolas Troquard. Dynamic logic of propositional assignments: A well-behaved variant of PDL. In *LICS*, pages 143–152. IEEE Computer Society, 2013.

[Belkhir *et al.*, 2013] Walid Belkhir, Yannick Chevalier, and Michael Rusinowitch. Fresh-variable automata: Application to service composition. In *15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC*, pages 153–160. IEEE Computer Society, 2013.

[Belkhir *et al.*, 2014] Walid Belkhir, Gisela Rossi, and Michael Rusinowitch. A parametrized propositional dynamic logic with application to service synthesis. In *Advances in Modal logic, Volume 10*, WS-FM'11, 2014.

[Blackburn *et al.*, 2001] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. CUP, 2001.

[Göller and Lohrey, 2006] Stefan Göller and Markus Lohrey. Infinite state model-checking of propositional dynamic logics. In Zoltán Ésik, editor, *CSL*, volume 4207 of *Lecture Notes in Computer Science*, pages 349–364. Springer, 2006.

[Lange and Stirling, 2001] Martin Lange and Colin Stirling. Focus games for satisfiability and completeness of temporal logic. In *LICS*, pages 357–365, 2001.

[Lutz, 2005] Carsten Lutz. PDL with intersection and converse is decidable. In C.-H. Luke Ong, editor, *CSL*, volume 3634 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2005.

[Zalta, 2007] Edward N. Zalta, editor. *Propositional Dynamic Logic*, 2007.