Universidad Nacional de Córdoba
Facultad de Matemática, Astronomía y Física

Lógicas Modales con Operadores de Cambio de Accesibilidad

Tesis presentada para obtener el

Doctorado en Ciencias de la Computación

Raul Alberto Fervari

Director

Dr. Carlos Areces	FaMAF, Universidad Nacional de Córdoba, Argentina

Jurado

Dr. Javier Blanco	FaMAF, Universidad Nacional de Córdoba, Argentina
Dr. Hans van Ditmarsch	LORIA, CNRS - Université de Lorraine, France
Dr. Pedro Sánchez Terraf	FaMAF, Universidad Nacional de Córdoba, Argentina

Córdoba, Argentina, 2014

Contenidos

Agradecimientos

Hace casi 4 años decidí empezar mi doctorado, y sigo pensando que fue una buena decisión. Llegué a FaMAF en junio de 2010 para encontrarme con Carlos y empezar a trabajar en Lógica. Las primeras ideas acerca de Swap Logic llegaron a fines de 2010, y después aparecieron nuevas ideas (las cuales me guiaron a esta tesis). Además de lógica, aprendí muchas otras cosas de los viajes que pude hacer y de toda la gente que conocí durante mi doctorado. Y precisamente, este texto introductorio es acerca de las personas. De aquellas que estuvieron conmigo desde el principio, y de aquellas que conocí gracias a este camino que decidí tomar. Me gustaría agradecerle a todos ellos.

En primer lugar, me gustaría agradecerle a Carlos Areces por su dirección. Gracias por enseñarme tantas cosas. Gracias por el tiempo dedicado a esta tesis. Gracias también por sus consejos, ayuda y apoyo (especialmente en momentos de pánico! :D) durante estos años. Gracias por presentarme tanta gente interesante, y por compartir conmigo tantas cosas.

Quiero agradecer especialmente a Guillaume Hoffmann por trabajar conmigo desde que llegó a Argentina. Gracias por su interés en los temas en los que yo estaba trabajando, y por colaborar en varios resultados de esta tesis. Por supuesto, gracias por todas las discusiones, salidas, el tiempo que coincidimos en Nancy, y por tantas otras cosas. También quiero agradecerle a Mauricio Martel por trabajar conmigo en los resultados de indecidibilidad, por estar disponible para viajar a Córdoba tantas veces como fueron necesarias, y por su entusiasmo.

Gracias a Hans van Ditmarsch y François Schwarzentruber, por todas las ideas interesantes surgidas durante mi visita a LORIA, algunas de ellas incluidas en la última parte de esta tesis. Gracias por todas las horas de trabajo, y por compartir nuevas ideas para investigar en el futuro. Le agredezco particularmente a Hans por ser parte de mi jurado.

Le quiero agradecer a todas las personas en FaMAF que me escucharon quejarme acerca de escribir mi tesis durante varios meses, especialmente durante el almuerzo (Pedro, Damián, Franco, Leti, Silvia, Miguel, y especialmente Chun), y a Eze por las charlas en el pasillo. Gracias al grupo LIIS, nuestro grupo. Gracias a Seba ("el filósofo"), por algo de bibliografía y la cita a Borges en el último capítulo. Gracias, por supuesto, a Javier y Pedro ST por aceptar ser parte de mi jurado.

Gracias a toda la gente de LORIA que me recibió en Nancy, en especial a los argentinos (Juan y César). Gracias por hacerme sentir cómodo, por las comidas, las reuniones, y todo el tiempo en la sala de música (incluso los fines de semana!). Gracias a César por recibirme en su casa en Nancy, y por finalmente visitarme en Córdoba. Gracias a la gente que conocí en la ESSLLI'13 (Facu, Nacho, Gustavo, Inari, etc.) por grandes momentos en Opole.

Además de la gente que jugó un papel importante en la parte académica de mi doctorado, hay mucha más gente de otros lugares, mis amigos de siempre. Quiero

agradecerles a todos ellos: amigos de mi pueblo, de la universidad, y a aquellos que conocí en alguna otra parte.

Finalmente, gracias a mi familia por todo su apoyo en cada decisión que tomo.

CLASIFICACIÓN: F.4.1 Mathematical Logic (Modal Logic).

PALABRAS CLAVE: lógicas modales, operadores de cambio de accesibilidad, operadores dinámicos, bisimulaciones, poder expresivo, complejidad, decidibilidad, lógicas dinámicas epistémicas.

Resumen

En esta tesis investigamos operadores modales dinámicos que pueden cambiar el modelo durante la evaluación de una fórmula. En particular, extendemos el lenguaje modal básico con modalidades que son capaces de invertir, borrar o agregar pares de elementos relacionados. Estudiamos la versión local de los operadores (es decir, la realización de modificaciones desde el punto de evaluación) y la versión global (cambiar arbitrariamente el modelo). Investigamos varias propiedades de los lenguajes introducidos, desde un punto de vista abstracto. En primer lugar, se introduce la semántica formal de los modificadores de modelo, e inmediatamente se introduce una noción de bisimulación. Las bisimulaciones son una herramienta importante para investigar el poder expresivo de los lenguajes introducidos en esta tesis. Se demostró que todas los lenguajes son incomparables entre sí en términos de poder expresivo (a excepción de los dos versiones de swap, aunque conjeturamos que también son incomparables). Continuamos por investigar el comportamiento computacional de este tipo de operadores. En primer lugar, demostramos que el problema de satisfactibilidad para las versiones locles de las lógicas que cambian la relación que investigamos es indecidible. También demostramos que el problema de model checking es PSPACE-completo para las seis lógicas. Finalmente, investigamos model checking fijando el modelo y fijando la fórmula (problemas conocidos como complejidad de fórmula y complejidad del programa, respectivamente). Es posible también definir métodos para comprobar satisfactibilidad que no necesariamente terminan. Introducimos métodos de tableau para las lógicas que cambian las relaciones y demostramos que todos estos métodos son correctos y completos y mostramos algunos aplicaciones.

En la última parte de la tesis, se discute un contexto concreto en el que pueden aplicarse las lógicas modales que cambian la relación: *Lógicas Dinámicas Epistémicas* ($\mathcal{DEL}$, por las siglas en inglés). Definimos una lógica que cambia la relación capaz de codificar $\mathcal{DEL}$, e investigamos su comportamiento computacional.

1

Lógicas con Operadores de Cambio de Accesibilidad

Ya no me digas que se siente. Si no se cambia hoy, no se cambia más...

from "Agua de la Miseria", Luis Alberto Spinetta.

1.1 LENGUAGE MODAL BÁSICO

Definición 1.1.1 (Sintaxis de la Lógica Modal Básica)**.** *Sea* PROP *un conjunto finito y contable de símbolos de proposición. Entonces, el conjunto* FORM *de fórmulas de* $\mathcal{ML}$ *sobre* PROP *está definido como:*

$$\mathsf{FORM} ::= \bot \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond\varphi,$$

donde $p \in \mathsf{PROP}$, *y* $\varphi, \psi \in \mathsf{FORM}$. *Otros operadores están definidos de la manera usual.* $\top$ *es* $\neg\bot$, $\varphi \vee \psi$ *se define como* $\neg(\neg\varphi \wedge \neg\psi)$ *y* $\Box\varphi$ *es* $\neg\Diamond\neg\varphi$.

Ahora definimos las estructuras donde interpretamos las fórmulas del lenguage modal básico. Los modelos son sólo grafos dirigidos con etiquetas en los nodos, llamados *Modelos de Kripke*.

Definición 1.1.2 (Modelos de Kripke)**.** *Un Modelo de Kripke* $\mathcal{M}$ *es una 3-upla* $\mathcal{M} = \langle W, R, V\rangle$, *donde* W *es un conjunto no vacío de elementos o estados;* $R \subseteq W \times W$ *es la relación de accesibilidad; y* $V : \mathsf{PROP} \to \mathcal{P}(W)$ *es una valuación. Sea* w *un estado en* $\mathcal{M}$, *el par* $(\mathcal{M}, w)$ *es un pointed model; usualmente omitimos los paréntesis y denotamos* $\mathcal{M}, w$ *a un pointed model.*

La Figura 1 muestra un ejemplo de un modelo de Kripke. Podemos observar que el modelo $\mathcal{M}$ es un grafo con 3 elementos, $\{w, v, u\}$. w está etiquetado con p, u con p y q, y v no posee etiqueta. Formalmente, $\mathcal{M}=\langle W, R, V\rangle$, donde $W=\{w, v, u\}$, $R=\{(w, v), (w, u), (v, v), (v, u), (u, v)\}$, y $V(p)=\{w, u\}, V(q)=\{u\}$.

Ahora definimos la semántica. Los operadores de $\mathcal{ML}$ describen propiedades *locales* de los modelos, lo que significa que las fórmulas son evaluadas en algún punto específico. Para eso usamos pointed models.

Definición 1.1.3 (Semántica del Lenguaje Modal Básico)**.** *Sea* $\mathcal{M}, w$ *un modelo y sea* φ *una fórmula, decimos que* $\mathcal{M}, w$ *satisface* φ *(notación,* $\mathcal{M}, w \models \varphi$*) sii*

$\mathcal{M}, w \models \bot$		*nunca*
$\mathcal{M}, w \models p$	*sii*	$w \in V(p)$
$\mathcal{M}, w \models \neg\varphi$	*sii*	$\mathcal{M}, w \not\models \varphi$
$\mathcal{M}, w \models \varphi \wedge \psi$	*sii*	$\mathcal{M}, w \models \varphi$ *y* $\mathcal{M}, w \models \psi$
$\mathcal{M}, w \models \Diamond\varphi$	*sii*	*existe* $v \in W$ *t.q.* $(w, v) \in R, \mathcal{M}, v \models \varphi$.

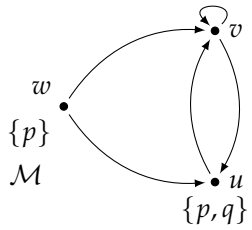


Figure 1: Ejemplo de un modelo de Kripke.

Una fórmula φ de $\mathcal{ML}$ es satisfactible *si existe un pointed model $\mathcal{M}, w$ tal que $\mathcal{M}, w \models \varphi$.*

$\mathcal{ML}$ posee mejores propiedades computacionales que la Lógica de Primer Orden $\mathcal{FOL}$. En particular, el problema de satisfactibilidad es decidible y su complejidad es PSPACE-completo. Por otro lado, model checking está en P.

1.2 CAMBIOS DE ACCESIBILIDAD

Muchas situaciones de la vida real son dinámicas. La manera en que cierto escenario cambia después de una acción determinada, diferentes factores externos modificando el entorno o simplemente el impacto del tiempo, de una manera muy general, son *dinámicos*. Por ejemplo, todos los días, las personas que viven en Córdoba se levantan por la mañana asumiendo que un autobús pasará por el mismo lugar que lo hizo el día anterior. Pero leyendo las noticias de la mañana, se enteran de que no circularán autobuses en la ciudad por algunos días como consecuencia de una huelga. Después de dos días, según lo prometido, el servicio vuelve a la normalidad. En esta situación, algunos de los factores (la huelga) *cambiaron* el escenario habitual, es decir, levantarse, ir a la parada para tomar el autobús y llegar al trabajo. Después de un determinado período de tiempo, todo cambió de nuevo al escenario habitual.

Consideremos ahora un escenario más interesante en el que la huelga es parcial, sólo algunas de las líneas de autobuses se vieron afectados. Esta es una instancia del escenario propuesto por Johan van Benthem en [van Benthem, 2005], en el que algún factor "malévolo" (en este caso la huelga) hace que las conexiones desde el hogar al trabajo desaparezcan. Como van Benthem propuso, esta situación se puede modelar como un juego de dos jugadores que podemos llamar un *juego de sabotaje*. Podemos representar las líneas de autobús como un gráfico, en el que dos puntos están conectados si hay un autobús que los une. Las líneas de puntos representan los caminos donde los autobuses fueron interrumpidos por la huelga.

Si supiéramos de antemano acerca de la huelga parcial, podríamos planificar rutas alternativas para llegar al trabajo. Sabemos que algunas partes de la ruta de casa al trabajo han sido saboteadas. Si el autobús no pasa por la parada habitual, tenemos que encontrar una alternativa desde donde podemos continuar nuestra ruta al trabajo. Esta situación, en la cual tenemos que encontrar una forma alternativa de ir a alguna parte, puede ser representada introduciendo un factor "benevolente", que añade aristas que no existían antes. Estas nuevas aristas nos permiten ir a una parte aislada previamente del gráfico (por ejemplo, caminando) donde los autobuses

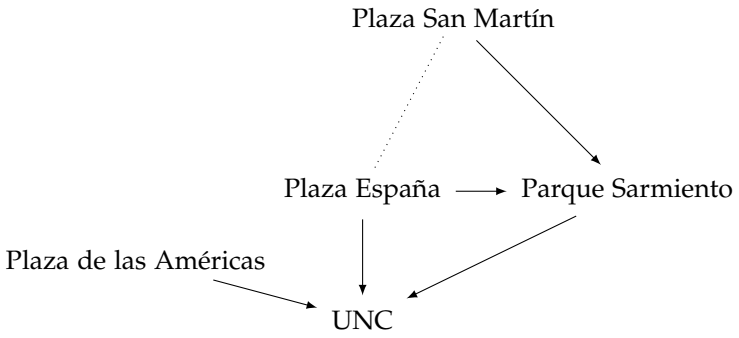


Figure 2: Grafo representando una huelga afectado las líneas de autobuses en el centro de la ciudad de Córdoba .

funcionan. En la siguiente situación las líneas de puntos son trayectorias recorridas a pie (formalmente, las nuevas aristas añadidas al modelo original).

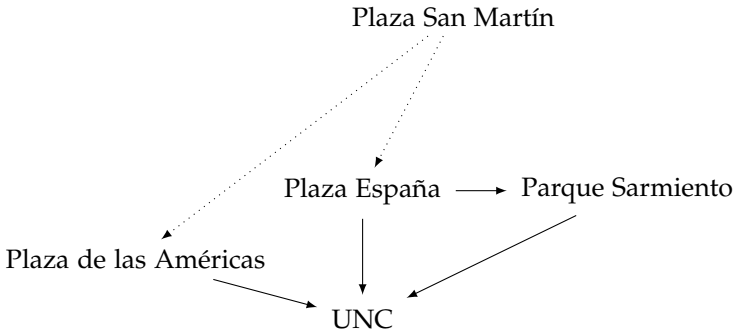


Figure 3: Grafo representando rutas alternativas (a pie).

Debería quedar claro en este punto el tipo de escenarios dinámicos en los cuales estamos interesados. Lo que es importante es que podemos *formalizar* todas estas situaciones y representarlos en un marco dinámico. La manera más obvia de hacerlo sería mediante la representación de todo el espacio de posibles evoluciones del grafo, pero esto es difícil de manejar. En su lugar, podemos modelar y obtener herramientas matemáticas para representar de forma dinámica estas situaciones. No sólo estamos pensando en problemas simples tales como huelgas en los servicios de autobús o el descubrimiento de un nuevo camino: estamos pensando en representaciones abstractas de este tipo de situaciones, así como en un marco dinámico para lidiar con ellas. En este caso, podemos tomar algunos objetos matemáticos (grafos) y diseñar herramientas para estudiar sus propiedades formales en este marco. Los lenguajes modales son adecuados para describir las propiedades de los grafos, y también podemos incluir operadores que captan las posibles evoluciones de los grafos.

1.3 CONOCIENDO LAS PRIMITIVAS

Primero introduzcamos la sintaxis.

Definición 1.3.1 (Sintaxis de las Lógicas con Operadores de Cambio de Accesibilidad)**.** *Sea* PROP *un conjunto infinito y contable de símbolos de proposición. Entonces el conjunto* FORM *de fórmulas sobre* PROP *está definido como:*

$$\mathsf{FORM} ::= \bot \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \blacklozenge\varphi,$$

donde $p \in \mathsf{PROP}$, $\blacklozenge \in \{\Diamond, \langle \mathsf{sw}\rangle, \langle \mathsf{sb}\rangle, \langle \mathsf{br}\rangle, \langle \mathsf{gsw}\rangle, \langle \mathsf{gsb}\rangle, \langle \mathsf{gbr}\rangle\}$ *y* $\varphi, \psi \in \mathsf{FORM}$*. Otros operadores están definidos de la manera usual. En particular,* $\blacksquare\varphi$ *se define como* $\neg\blacklozenge\neg\varphi$*.*

Denotamos $\mathcal{ML}(\blacklozenge)$ *a la extensión de* $\mathcal{ML}$ *con el operador* $\blacklozenge$*, para* $\blacklozenge \in \{\langle \mathsf{sw}\rangle, \langle \mathsf{sb}\rangle, \langle \mathsf{br}\rangle, \langle \mathsf{gsw}\rangle, \langle \mathsf{gsb}\rangle, \langle \mathsf{gbr}\rangle\}$*.*

Las fórmulas de $\mathcal{ML}(\langle \mathsf{sb}\rangle)$, $\mathcal{ML}(\langle \mathsf{sw}\rangle)$, $\mathcal{ML}(\langle \mathsf{br}\rangle)$, $\mathcal{ML}(\langle \mathsf{gsb}\rangle)$, $\mathcal{ML}(\langle \mathsf{gbr}\rangle)$, y $\mathcal{ML}(\langle \mathsf{gsw}\rangle)$ son evaluadas en modelos relacionales estándar, y el significado de los operadores modales básicos no cambia. Cuando evaluamos fórmulas dinámicas, necesitamos mantener el registro de las aristas que han sido modificadas. En el resto de la tesis usaremos wv como una abreviación para $\{(w,v)\}$ o (w,v). El contexto siempre quitará la ambigüedad.

Definición 1.3.2 (Modelos y Model Variants)**.** *Un modelo* $\mathcal{M}$ *es una 3-upla* $\mathcal{M} = \langle W, R, V\rangle$*, donde W es un conjunto no vacío de elementos o estados;* $R \subseteq W \times W$ *es la relación de accesibilidad; y* $V : \mathsf{PROP} \to \mathcal{P}(W)$ *es una valuación.*

Dado un modelo $\mathcal{M} = \langle W, R, V\rangle$*, definimos las siguientes notaciones:*

(sabotaging)	$\mathcal{M}^-_S = \langle W, R^-_S, V\rangle$*, con* $R^-_S = R \backslash S$*,* $S \subseteq R$*.*
(swapping)	$\mathcal{M}^*_S = \langle W, R^*_S, V\rangle$*, con* $R^*_S = (R \backslash S^{-1}) \cup S$*,* $S \subseteq R^{-1}$*.*
(bridging)	$\mathcal{M}^+_B = \langle W, R^+_B, V\rangle$*, con* $R^+_B = R \cup B$*,* $B \subseteq (W \times W) \backslash R$*.*

Sea w un estado en $\mathcal{M}$*, el par* $(\mathcal{M}, w)$ *es llamado pointed model; en general, omitimos los paréntesis y escribimos* $\mathcal{M}, w$ *para un pointed model.*

Los model variants son modelos de Kripke en los cuales se han realizado algunas actualizaciones por medio de operadores dinámicos. Esta notación será de gran utilidad para presentar la semántica de los operadores, y también más tarde para introducir otras construcciones como bisimulaciones o tableaux.

Definición 1.3.3 (Semántica)**.** *Sea* $\mathcal{M}, w$ *un pointed model, y sea* φ *una fórmula, decimos que* $\mathcal{M}, w$ *satisface* φ *y escribimos* $\mathcal{M}, w \models \varphi$*, cuando*

$\mathcal{M}, w \models \bot$		*nunca*
$\mathcal{M}, w \models p$	*sii*	$w \in V(p)$
$\mathcal{M}, w \models \neg\varphi$	*sii*	$\mathcal{M}, w \not\models \varphi$
$\mathcal{M}, w \models \varphi \wedge \psi$	*sii*	$\mathcal{M}, w \models \varphi$ *y* $\mathcal{M}, w \models \psi$
$\mathcal{M}, w \models \Diamond\varphi$	*sii*	*existe* $v \in W$ *t.q.* $(w,v) \in R, \mathcal{M}, v \models \varphi$
$\mathcal{M}, w \models \langle \mathsf{sb}\rangle\varphi$	*sii*	*existe* $v \in W$ *t.q.* $(w,v) \in R, \mathcal{M}^-_{wv}, v \models \varphi$
$\mathcal{M}, w \models \langle \mathsf{gsb}\rangle\varphi$	*sii*	*existe* $v, u \in W$*, t.q.* $(v,u) \in R, \mathcal{M}^-_{vu}, w \models \varphi$
$\mathcal{M}, w \models \langle \mathsf{sw}\rangle\varphi$	*sii*	*existe* $v \in W$ *t.q.* $(w,v) \in R, \mathcal{M}^*_{vw}, v \models \varphi$
$\mathcal{M}, w \models \langle \mathsf{gsw}\rangle\varphi$	*sii*	*existe* $v, u \in W$*, t.q.* $(v,u) \in R, \mathcal{M}^*_{uv}, w \models \varphi$
$\mathcal{M}, w \models \langle \mathsf{br}\rangle\varphi$	*sii*	*existe* $v \in W$ *t.q.* $(w,v) \notin R, \mathcal{M}^+_{wv}, v \models \varphi$
$\mathcal{M}, w \models \langle \mathsf{gbr}\rangle\varphi$	*sii*	*existenv*$, u \in W$*, t.q.* $(v,u) \notin R, \mathcal{M}^+_{vu}, w \models \varphi$*.*

φ es satisfactible si para algún pointed model $\mathcal{M}, w$ tenemos que $\mathcal{M}, w \models \varphi$.

Los operadores de cambio de accesibilidad pueden borrar, invertir o agregar nuevas aristas. La fórmula se evalúa en el model variant que corresponda. Extendemos la lógica modal básica con primitivas que nos permiten realizar estas operaciones en dos versiones: locales y globales. La versión local, borra, invierte o agrega aristas a nodos adyacentes, mientras que la versión global modifica aristas en cualquier parte del modelo.

1.4 ALGUNAS PROPIEDADES DE MODELOS

En esta sección, vamos a investigar algunas propiedades clásicas para las lógicas modales. Algunos resultados de expresividad se han demostrado gracias a que las fórmulas de un determinado lenguaje pueden o no forzar ciertos modelos con una estructura determinada. La *tree* y la *finite model property*, son ejemplos de este tipo de propiedades. La tree model property establece que cada fórmula satisfactible, también se satisface en la raíz de un árbol. La propiedad del modelo finito dice que ninguna fórmula puede forzar un modelo infinito. $\mathcal{ML}$ tiene ambas propiedades. Vamos a investigar lo que ocurre cuando añadimos operadores de cambio de accesibilidad.

Definición 1.4.1. *Sea $\mathcal{L}$ un lenguage, decimos que $\mathcal{L}$ tiene la* Tree Model Property *si para toda $\mathcal{L}$-fórmula φ satisfactible, φ se satisface también en la raíz de un árbol.*

Entonces:

Teorema 1.4.2. *$\mathcal{ML}$ tiene la tree model property.*

Demostración. En [Blackburn and van Benthem, 2006] una construcción llamada *unraveling* es introducida. Esta construcción fue introducida primera en [Dummet and Lemmon, 1959] y luego utilizada en [Sahlqvist, 1973]. Dicho unraveling genera un modelo bisimilar al original, por lo tanto satisface las mismas fórmulas. □

A continuación mostramos que agregando operadores cambio de accesibilidad, perdemos la tree model property.

Teorema 1.4.3. *$\mathcal{ML}(\blacklozenge)$ no tiene la tree model property, para todo $\blacklozenge \in \{\langle \mathsf{sb}\rangle, \langle \mathsf{gsb}\rangle, \langle \mathsf{sw}\rangle, \langle \mathsf{gsw}\rangle, \langle \mathsf{br}\rangle, \langle \mathsf{gbr}\rangle\}$.*

Demostración. Presentamos fórmulas tal que no puede ser stisfechas en un árbol. Para $\langle \mathsf{gsb}\rangle$, ya fue demostrado anteriormente en [Löding and Rohde, 2003a], para $\langle \mathsf{sw}\rangle$ en [Areces *et al.*, 2013b] y para $\langle \mathsf{sb}\rangle$ y $\langle \mathsf{br}\rangle$ en [Areces *et al.*, 2012]. Definamos primero algo de notación que resultará útil:

- $\blacksquare^0\varphi = \varphi$,
- $\blacksquare^{n+1}\varphi = \blacksquare\blacksquare^n\varphi$,
- $\blacksquare^{(n)}\varphi = \bigwedge_{1\leq i\leq n}\blacksquare^i\varphi$,

para $\blacksquare \in \{\Box, [\mathsf{sb}], [\mathsf{gsb}], [\mathsf{sw}], [\mathsf{gsw}], [\mathsf{br}], [\mathsf{gbr}]\}$.

Supongamos que las siguientes fórmulas se satisfacen en algún punto w en un modelo:

1. $\Diamond\Diamond\top \wedge [\mathsf{sb}]\Box\bot$, entonces w es reflexivo;
2. $\Diamond\Diamond\top \wedge [\mathsf{gsb}]\Box\bot$, ídem al anterior;
3. $p \wedge \Box^{(3)}\neg p \wedge \langle\mathsf{sw}\rangle\Diamond\Diamond p$, entonces w tiene un sucesor reflexivo;
4. $\Box\bot \wedge \langle\mathsf{gsw}\rangle\Diamond\top$, entonces w tiene una arista que lo apunta.
5. $\Box\bot \wedge \langle\mathsf{br}\rangle\langle\mathsf{br}\rangle\top$, entonces w y algún punto v distinto están desconectados;
6. $\Box\bot \wedge \langle\mathsf{gbr}\rangle\langle\mathsf{gbr}\rangle\top$, ídem al anterior.

En cada caso, la fórmula no puede satifacerse en la raíz de un árbol. Discutamos cada caso en detalle.

□

Los resultados anteriores nos dan las primeras intuiciones sobre la expresividad de las lógicas con operadores de cambio de accesibilidad. Hemos demostrado en el Teorema 1.4.3 que hay fórmulas satisfactibles de las seis lógicas que investigamos, que no pueden ser satisfechas en la raíz de un árbol. Esto demuestra que agregando dichos operadores obtenemos más expresividad. El reto es descubrir cuánto poder expresivo agregamos. Vamos a explorar ahora, otra propiedad clásica para las lógicas modales: la propiedad del modelo finito. Esta propiedad establece que toda fórmula que es satisfactible lo es también en un modelo finito. $\mathcal{ML}$ tiene la propiedad de modelo finito, y también satisface una variante más fuerte: la propiedad del modelo acotado. Se llama "acotada" porque además de la propiedad del modelo finito, podemos determinar una cota para tales modelos. Vamos a introducir la definición formal.

Definición 1.4.4. *Sea $\mathcal{L}$ un lenguage, decimos que $\mathcal{L}$ tiene la* Finite Model Property *(propiedad de modelo finito) si para toda $\mathcal{L}$-fórmula φ que es satisfactible, φ se satisface en un modelo finito.*

$\mathcal{L}$ tiene la Bounded Finite Model Property *(propiedad del modelo acotado) si para toda $\mathcal{L}$-formula φ satisfactible, φ se satisface en un modelo finito, cuya longitud es acotada por una función en el tamaño de la fórmula.*

Teorema 1.4.5. *$\mathcal{ML}$ tiene la bounded finite model property.*

Demostración (Sketch). Un demostración completa via filtraciones o selección puede ser encontrada en [Blackburn *et al.*, 2001]. □

Teorema 1.4.6. *$\mathcal{ML}(\blacklozenge)$ no tiene la finite model property, para todo $\blacklozenge \in \{\langle\mathsf{sb}\rangle, \langle\mathsf{gsb}\rangle, \langle\mathsf{sw}\rangle, \langle\mathsf{gsw}\rangle, \langle\mathsf{br}\rangle, \langle\mathsf{gbr}\rangle\}$.*

Para cada lógica introducimos una fórmula que fuerza modelos infinitos.

Demostración. Veamos las fórmulas para cada caso.

1. Para $\mathcal{ML}(\langle \mathsf{sw} \rangle)$, definimos la siguiente conjunción:

$$\begin{array}{lll} \varphi = & s \wedge \Box^{(9)} \neg s & (1) \\ & \wedge \Diamond \top & (2) \\ & \wedge \Box \Diamond \top & (3) \\ & \wedge [\mathsf{sw}] \Box \neg \Diamond s & (Irr) \\ & \wedge [\mathsf{sw}][\mathsf{sw}](\neg s \rightarrow \Diamond\Diamond\Diamond\Diamond\Diamond s) & (Spy) \\ & \wedge [\mathsf{sw}][\mathsf{sw}][\mathsf{sw}](\neg\Diamond s \rightarrow \Diamond\Diamond\Diamond(\neg s \wedge \Diamond\Diamond\Diamond s)) & (Trans) \end{array}$$

   El modelo de la Figura 4 es uno que satisface φ.

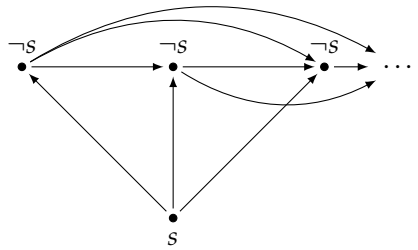


Figure 4: Modelo infinito para $\mathcal{ML}(\langle \mathsf{sw} \rangle)$.

2. Para $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$.

$$\begin{array}{lll} \varphi = & s \wedge \Box^{(9)} \neg s \wedge \Diamond \top \wedge \Box \Diamond \top & (1) \\ & \wedge \Box\Box[\mathsf{gsw}][\mathsf{gsw}]\Box\Box(s \rightarrow \Diamond\Diamond\Diamond s) & (Spy) \\ & \wedge \Box[\mathsf{gsw}](\Diamond s \rightarrow \Box\Box\neg s) & (Irr) \\ & \wedge \Box\Box\Box[\mathsf{gsw}][\mathsf{gsw}][\mathsf{gsw}](\Diamond\Diamond\Diamond s \rightarrow \Diamond\Diamond\Diamond(\neg s \wedge \Diamond\Diamond\Diamond s)) & (Trans) \end{array}$$

   La fórmula es satisfactible y sus modelos son infinitos. El modelo de la Figura 4 satisface φ.

3. Para $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ seguimos las mismas ideas, pero borrando aristas apuntando al spy point. La Figura 5 muestra la clase de modelos que queremos forzar.

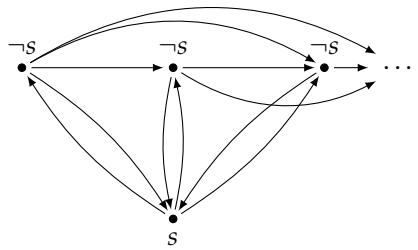


Figure 5: Modelo infinito para $\mathcal{ML}(\langle \mathsf{sb} \rangle)$.

Ahora introducimos una fórmula que fuerza modelos como el de la Figura 5.

$$
\begin{array}{lll}
\varphi = & s \wedge \Box\neg s \wedge \Diamond\top \wedge \Box\Diamond s & (1)\\
& \wedge \Box\Box(s \to \neg\Diamond s) & (2)\\
& \wedge [\mathsf{sb}][\mathsf{sb}](s \to \Box\Diamond s) & (3)\\
& \wedge \Box[\mathsf{sb}](s \to \Diamond\neg\Diamond s) & (4)\\
& \wedge \Box\Diamond\neg s & (5)\\
& \wedge \Box\Box(\neg s \to \Diamond(s \wedge \neg\Diamond s)) & (6)\\
& \wedge \Box[\mathsf{sb}](\neg s \to [\mathsf{sb}](s \to \Box\Box(\neg s \to \Diamond s))) & (7)\\
& \wedge \Box\Box(\neg s \to [\mathsf{sb}](s \to \Diamond\Diamond(\neg s \wedge \neg\Diamond s))) & (8)\\
& \wedge \Box\Box(\neg s \to [\mathsf{sb}](s \to \Diamond\neg\Diamond s)) & (Spy)\\
& \wedge \Box[\mathsf{sb}](s \to \Box(\neg\Diamond s \to \Box\Diamond s)) & (Irr)\\
& \wedge \neg\Diamond\langle\mathsf{sb}\rangle(s \wedge \Diamond(\neg\Diamond s \wedge \Diamond\Diamond(\neg s \wedge \Diamond s \wedge \Diamond\neg\Diamond s))) & (3cyc)\\
& \wedge \Box\langle\mathsf{sb}\rangle(s \wedge \Diamond(\neg\Diamond s \wedge \Diamond\Diamond(\neg s \wedge \langle\mathsf{sb}\rangle(s \wedge \Diamond(\neg\Diamond s \wedge \Diamond\neg\Diamond s))))) & (Trans)
\end{array}
$$

4. Para $\mathcal{ML}(\langle\mathsf{gsb}\rangle)$ definimos una fórmula que fuerza modelos tales como el de la Figura 6.

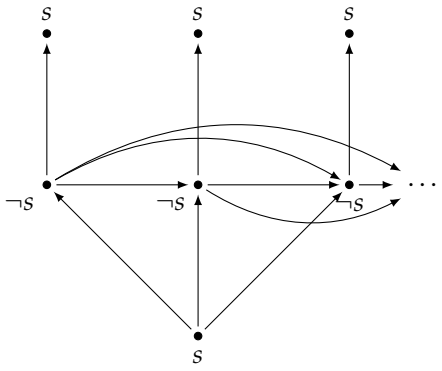


Figure 6: Modelo infinito para $\mathcal{ML}(\langle\mathsf{gsb}\rangle)$.

Ahora introducimos la fórmula que fuerza modelos como el anterior.

$$
\begin{array}{lll}
\varphi = & s \wedge \Box\neg s \wedge \Diamond\top & (1)\\
& \wedge \Box(\Diamond s \wedge \langle\mathsf{gsb}\rangle\neg\Diamond s) & (2)\\
& \wedge \Box\Diamond\neg s & (3)\\
& \wedge \Box\Box(\neg s \to \Diamond s \wedge \langle\mathsf{gsb}\rangle\neg\Diamond s) & (4)\\
& \wedge [\mathsf{gsb}](\Diamond(\Diamond s \wedge \Diamond(\neg s \wedge \neg\Diamond s)) \to \Diamond\neg\Diamond s) & (Spy)\\
& \wedge [\mathsf{gsb}]\Box(\neg\Diamond s \to \Box\Diamond s) & (Irr)\\
& \wedge [\mathsf{gsb}]\neg\Diamond(\neg\Diamond s \wedge \Diamond\Diamond(\neg s \wedge \Diamond s \wedge \Diamond\neg\Diamond s)) & (3cyc)\\
& \wedge [\mathsf{gsb}][\mathsf{gsb}](\Diamond(\neg\Diamond s \wedge \Diamond\Diamond(\neg s \wedge \neg\Diamond s)) \to \Diamond(\neg\Diamond s \wedge \Diamond\neg\Diamond s)) & (Trans)
\end{array}
$$

5. Para $\mathcal{ML}(\langle\mathsf{br}\rangle)$ forzamos otra estructura. La idea ahora es crear una "bolsa" de spy points, relacionados entre ellos. Uno de los modelos infinitos que forzamos es introducido en Figura 7.

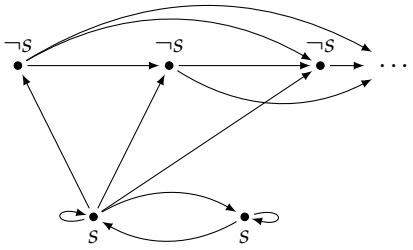


Figure 7: Modelo infinito para $\mathcal{ML}(\langle \mathsf{br} \rangle)$.

La fórmula que fuerza tales modelos es:

$$
\begin{array}{lll}
\varphi = & s \wedge [\mathsf{br}]\neg s & (1) \\
& \wedge \Box[\mathsf{br}]\neg s & (s - connex) \\
& \wedge \Box\Box s & (3) \\
& \wedge \langle \mathsf{br} \rangle \top & (4) \\
& \wedge \Box[\mathsf{br}]\Box\neg s & (5) \\
& \wedge [\mathsf{br}]\Diamond\top & (Ser) \\
& \wedge [\mathsf{br}][\mathsf{br}](s \rightarrow \Box(\neg s \rightarrow \Box(\neg s \rightarrow \Box\neg s))) & (Irr) \\
& \wedge [\mathsf{br}]\Box\Box[\mathsf{br}](s \rightarrow \Diamond(\neg s \wedge \Diamond\Diamond s)) & (Trans)
\end{array}
$$

6. Para $\mathcal{ML}(\langle \mathsf{gbr} \rangle)$ forzamos el mismo tipo de modelos que para $\mathcal{ML}(\langle \mathsf{br} \rangle)$.

$$
\begin{array}{lll}
\varphi = & s \wedge \Box\neg s \wedge \Diamond\top \wedge \Box\Box\neg s & (1) \\
& \Box\Diamond\neg s & (2) \\
& \Box\Box(\neg s \rightarrow \Box\neg s) & (3) \\
& [\mathsf{gbr}](\Diamond(\neg\Diamond s \wedge \Diamond\Diamond s) \rightarrow \Diamond\Diamond s) & (Spy) \\
& [\mathsf{gbr}]\Box(\Diamond s \rightarrow \Box\neg\Diamond s) & (Irr) \\
& [\mathsf{gbr}]\neg\Diamond(\Diamond s \wedge \Diamond(\neg s \wedge \Diamond(\neg s \wedge \neg\Diamond s \wedge \Diamond\Diamond s))) & (3cyc) \\
& [\mathsf{gbr}][\mathsf{gbr}](\Diamond(\Diamond s \wedge \Diamond(\neg s \wedge \Diamond(\neg s \wedge \Diamond s))) \rightarrow \Diamond(\Diamond s \wedge \Diamond\Diamond s)) & (Trans)
\end{array}
$$

□

Mostramos que las seis lógicas que investigamos son más expresivas que la lógica modal básica. Sin embargo, no tenemos cotas exactas. En los siguientes capítulos introduciremos las herramientas adecuadas para estudiar expresividad: *bisimulaciones*. Comenzaremos por introducir esas nociones para $\mathcal{ML}$ y luego las extenderemos para los operadores de cambio de accesibilidad.

2

Poder Expresivo

Anything that thinks logically can be fooled by something else that thinks at least as logically as it does.

from "The Hitchhiker's Trilogy", Douglas Adams.

2.1 DIFERENCIAS MODALES

El poder expresivo de un lenguaje se mide en términos de las distinciones que podemos expresar en el lenguaje. Para ello, las *bisimulaciones* introducidas por van Benthem en [van Benthem, 1984; van Benthem, 1985] serán útiles.

En teoría de modelos modal, la noción de bisimulación es una herramienta fundamental. En general, una bisimulación es una relación binaria relacionando elementos de los dominios que tienen la misma información atómica, y que preservan las mismas propiedades estructurales. Introduzcamos la noción de bisimulación para el lenguaje modal básico $\mathcal{ML}$.

Definición 2.1.1 ($\mathcal{ML}$-Bisimulaciones)**.** *Sean $\mathcal{M} = \langle W, R, V\rangle$, $\mathcal{M}' = \langle W', R', V'\rangle$ dos modelos. Una relación no vacía $Z \subseteq W \times W'$ es una $\mathcal{ML}$-bisimulación si satisface las siguientes condiciones. Si wZw' entonces*

(atomic harmony) *para todo $p \in \mathsf{PROP}$, $w \in V(p)$ sii $w' \in V'(p)$;*

(zig) *si $(w, v) \in R$ entonces existe v', $(w', v') \in R'$ y vZv';*

(zag) *si $(w', v') \in R'$ entonces existe v, $(w, v) \in R$ y vZv'.*

Sean $\mathcal{M}, w$ y $\mathcal{M}', w'$ dos pointed models, decimos que son $\mathcal{ML}$-bisimilares y lo denotamos $\mathcal{M}, w \leftrightarroweq_{\mathcal{ML}} \mathcal{M}', w'$ si existe una $\mathcal{ML}$-bisimulación Z tal que wZw'.

La importancia de la noción de bisimulación se puede ver en el siguiente teorema, que establece que las bisimulaciones relacionan modelos modalmente equivalentes.

Teorema 2.1.2 (Invarianza por Bisimulaciones)**.** *Sean $\mathcal{M}=\langle W, R, V\rangle$, $\mathcal{M}'=\langle W', R', V'\rangle$ dos modelos, $w \in W$ y $w' \in W'$. Si existe una $\mathcal{ML}$-bisimulación Z entre $\mathcal{M}, w$ y $\mathcal{M}', w'$ tal que wZw' entonces para toda fórmula $\varphi \in \mathcal{ML}$, $\mathcal{M}, w \models \varphi$ sii $\mathcal{M}', w' \models \varphi$.*

Demostración. La demostración es por inducción estructural en $\mathcal{ML}$-fómulas.

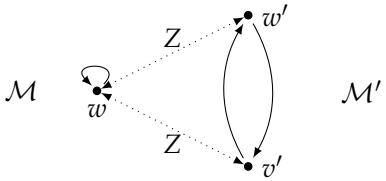
- $\boldsymbol{\varphi = p}$: Supongamos $\mathcal{M}, w \models p$. Entonces por definición de $\models$, $w \in V(p)$. Pero como wZw' y por (atomic harmony), $w' \in V'(p)$, entonces $\mathcal{M}', w' \models p$.
- $\boldsymbol{\varphi = \neg\psi}$: Supongamos $\mathcal{M}, w \models \neg\psi$. Entonces por definición de $\models$, $\mathcal{M}, w \not\models \psi$. Pero por H.I., $\mathcal{M}', w' \not\models \psi$ entonces $\mathcal{M}', w' \models \neg\psi$.

$\varphi = \psi \wedge \chi$: Supongamos $\mathcal{M}, w \models \psi \wedge \chi$. Entonces $\mathcal{M}, w \models \psi$ and $\mathcal{M}, w \models \chi$. Por H.I., $\mathcal{M}', w' \models \psi$ y $\mathcal{M}', w' \models \chi$, entonces $\mathcal{M}', w' \models \psi \wedge \chi$.

$\varphi = \Diamond\psi$: supongamos $\mathcal{M}, w \models \Diamond\psi$. Entonce existe v en W t.q. $R(w, v)$ y $\mathcal{M}, v \models \psi$. Por (zig) tenemos que v' en W' t.q. $R'(w', v')$ y vZv'. Por H.I., $\mathcal{M}', v' \models \psi$ y por definición de $\mathcal{M}, w' \models \Diamond\psi$. Para la otra dirección usar (zag).

□

Ejemplo 2.1.3. *Este es un ejemplo de modelos que no pueden ser distinguidos por ninguna fórmula de $\mathcal{ML}$. Las líneas punteadas representan la relación Z, que define una bisimulación.*



Para las lógicas modales con operadores de cambio las nociones presentadas hasta el momento no son suficientes. Las bisimulaciones como han sido presentadas en esta sección no captan el comportamiento de los operadores dinámicos. Afortunadamente, no necesitamos encontrar nuevas herramientas para estudiar los operadores de cambio de accesibilidad. En la siguiente sección, vamos a demostrar que adaptando las herramientas existentes, captamos exactamente el significado de las seis lógicas introducidas en esta tesis.

2.2 CÓMO DESCRIBIR CAMBIOS

En las lógicas que introdujimos en el Capítulo 1, no alcanza con bisimulaciones que solo relacionan elementos de los dominios. Debido a que estamos estudiando lógicas donde un modelo puede cambiar en cualquier momento mientras estamos evaluando una fórmula, tenemos que mantener un seguimiento de las modificaciones que el modelo ya sufrió. Para las lógicas con operadores de cambio de accesibilidad las bisimulaciones capturan el comportamiento dinámico relacionando estados junto a la actual relación de accesibilidad. En [Areces *et al.*, 2012; Areces *et al.*, 2013b] introdujios las siguientes nociones de bisimulación:

Definición 2.2.1 ($\mathcal{ML}(\blacklozenge)$-Bisimulaciones). *Sean $\mathcal{M} = \langle W, R, V\rangle$, $\mathcal{M}' = \langle W', R', V'\rangle$ dos modelos. Una relación no vacía $Z \subseteq (W \times \mathcal{P}(W^2)) \times (W' \times \mathcal{P}(W'^2))$ es una $\mathcal{ML}(\blacklozenge)$-bisimulación si satisface las siguientes condiciones* atomic harmony, zig *y* zag, *y las correspondientes condiciones de acuerdo a los operadores que la lógica contenga. Si $(w, S)Z(w', S')$ entonces*

(atomic harmony) *para todo $p \in \mathsf{PROP}$, $w \in V(p)$ sii $w' \in V'(p)$;*

(zig) *si $(w, v) \in S$ entonces existe v', $(w', v') \in S'$ y $(v, S)Z(v', S')$;*

(zag) *si $(w', v') \in S'$ entonces existe v, $(w, v) \in S$ y $(v, S)Z(v', S')$;*

($\langle\mathsf{sb}\rangle$-zig) *si $(w, v) \in S$ entonces existe v', $(w', v') \in S'$ y $(v, S^{-}_{vw})Z(v'S'^{-}_{v'w'})$;*

($\langle$sb$\rangle$-zag) *si* $(w', v') \in S'$ *entonces existe* v, $(w, v) \in S$ *y* $(v, S^{-}_{wv})Z(v' S'^{-}_{w'v'})$;

($\langle$gsb$\rangle$-zig) *si* $(u, v) \in S$ *entonces existe* u', v', $(u', v') \in S'$ *y* $(w, S^{-}_{uv})Z(w' S'^{-}_{u'v'})$;

($\langle$gsb$\rangle$-zag) *si* $(u', v') \in S'$ *entonces existe* u, v, $(u, v) \in S$ *y* $(w, S^{-}_{uv})Z(w' S'^{-}_{u'v'})$;

($\langle$sw$\rangle$-zig) *si* $(w, v) \in S$ *entonces existe* v', $(w', v') \in S'$ *y* $(v, S^{*}_{vw})Z(v' S'^{*}_{v'w'})$;

($\langle$sw$\rangle$-zag) *si* $(w', v') \in S'$ *entonces existe* v, $(w, v) \in S$ *y* $(v, S^{*}_{vw})Z(v' S'^{*}_{v'w'})$;

($\langle$gsw$\rangle$-zig) *si* $(u, v) \in S$ *entonces existe* u', v', $(u', v') \in S'$ *y* $(w, S^{*}_{vu})Z(w' S'^{*}_{v'u'})$;

($\langle$gsw$\rangle$-zag) *si* $(u', v') \in S'$ *entonces existe* u, v, $(w, v) \in S$ *y* $(w, S^{*}_{vu})Z(w' S'^{*}_{v'u'})$;

($\langle$br$\rangle$-zig) *si* $(w, v) \notin S$, *existe* $v' \in W'$ *t.q.* $(w', v') \notin S'$ *y* $(v, S^{+}_{wv})Z(v', S'^{+}_{w'v'})$;

($\langle$br$\rangle$-zag) *si* $(w', v') \notin S'$, *existe* $v \in W$ *t.q.* $(w, v) \notin S$ *y* $(v, S^{+}_{wv})Z(v', S'^{+}_{w'v'})$;

($\langle$gbr$\rangle$-zig) *si* $(u, v) \notin S$, *existen* $u', v' \in W'$ *t.q.* $(u', v') \notin S'$ *y* $(w, S^{+}_{uv})Z(w', S'^{+}_{u'v'})$;

($\langle$gbr$\rangle$-zag) *si* $(u', v') \notin S'$, *existen* $u, v \in W$ *t.q.* $(u, v) \notin S$ *y* $(w, S^{+}_{uv})Z(w', S'^{+}_{u'v'})$.

Sean $\mathcal{M}, w$ *y* $\mathcal{M}', w'$ *dos pointed models, decimos que son* $\mathcal{ML}(\blacklozenge)$*-bisimilares y lo denotamos* $\mathcal{M}, w \leftrightarroweq_{\mathcal{ML}(\blacklozenge)} \mathcal{M}', w'$ *si existe una* $\mathcal{ML}(\blacklozenge)$*-bisimulación* Z *tal que* $(w, R)Z(w, R')$ *donde* R *y* R' *son las relaciones de* $\mathcal{M}$ *y* $\mathcal{M}'$ *respectivamente.*

El siguiente teorema establece que dos modelos bisimilares no son distinguibles por ninguna fórmula del lenguaje correspondiente.

Teorema 2.2.2 (Invarianza por Bisimulaciones)**.** *Sean* $\mathcal{M}=\langle W, R, V\rangle$ *y* $\mathcal{M}'=\langle W', R', V'\rangle$ *dos modelos,* $w \in W$, $w' \in W'$, *y sea* $S \subseteq W^2, S' \subseteq W'^2$. *Si existe una* $\mathcal{ML}(\blacklozenge)$*-bisimulación* Z *entre* $\mathcal{M}, w$ *y* $\mathcal{M}', w'$ *tal que* $(w, S)Z(w', S')$ *entonces para toda fórmula* $\varphi \in \mathcal{ML}(\blacklozenge)$, $\langle W, S, V\rangle, w \models \varphi$ *sii* $\langle W', S', V'\rangle, w' \models \varphi$.

Demostración. Vamos a mostrar en caso para $\mathcal{ML}(\langle \mathsf{sw}\rangle)$. La demostración es por inducción estructural en las fórmula de $\mathcal{ML}(\langle \mathsf{sw}\rangle)$. El caso base vale por (atomic harmony), y los casos $\wedge$ y $\neg$ son triviales..

- $\boldsymbol{\varphi = \Diamond\psi}$: Supongamos $\langle W, S, V\rangle, w \models \Diamond\psi$. Entonces existe v en W t.q. $(w, v) \in S$ y $\langle W, S, V\rangle, v \models \psi$. Por (zig) existe v' en W' tal que $w' S' v'$ y $(v, S)Z(v', S')$. Por H.I., $\langle W', S', V'\rangle', v' \models \psi$ y por definición $\langle W', S', V'\rangle, w' \models \Diamond\psi$. Para la otra dirección usar (zag).
- $\boldsymbol{\varphi = \langle \mathsf{sw}\rangle\psi}$: Supongamos $\langle W, S, V\rangle, w \models \langle \mathsf{sw}\rangle\psi$. Entonces existe v en W t.q. $(w, v) \in S$ y $\langle W, S^{*}_{vw}, V\rangle, v \models \psi$. Por ($\langle$sw$\rangle$-zig) existe v' in W' t.q. $(w', v') \in S'$ y $(v, S^{*}_{vw})Z(v', S'^{*}_{v'w'})$. Por hipótesis inductiva, $\langle W', S'^{*}_{v'w'}, V'\rangle, v' \models \psi$ y por definición $\langle W', S', V'\rangle, w' \models \langle \mathsf{sw}\rangle\psi$. Para la otra dirección usar ($\langle$sw$\rangle$-zag). $\square$

Podemos observar que las bisimulaciones para las lógicas modales con operadores de cambio de accesibilidad, relaciones estados y relaciones de accesibildiad actuales de los modelos. Dependiendo de qué operador estamos considerando, diferentes condiciones de zig/zag son agregadas. Zig y zag para $\mathcal{ML}$-bisimulaciones son las

condiciones correspondientes para captar $\Diamond$: hablan de los sucesores de la situación actual. Condiciones para las lógicas modales con operadores de cambio de accesibilidad son las mismas, pero además se debe hacer el seguimiento de las modificaciones ya realizadas de acuerdo con la semántica de los operadores considerados. Por ejemplo, $\langle \mathsf{sb} \rangle$ -zig/zag establecen que hay sucesores del estado actual que están relacionados, y borra las aristas que los conectan. Para $\langle \mathsf{sw} \rangle$ son las mismas condiciones, pero intercambiando aristas en vez de borrarlas. Condiciones para $\langle \mathsf{br} \rangle$ requieren que existen puntos inalcanzables desde los estados actuales y agrega aristas hacia ellos en la relación de accesibilidad. Los casos globales son similares a los casos locales, pero sin cambiar el punto de evaluación.

Bisimulaciones y el Teorema 2.2.2 nos brindan las herramientas necesarias para investigaar el poder expresivo de las lógicas que introdujimos. Podemos usar dichas herramientas para comparar diferentes lógicas (las que introdujimos, y otras). La Definición 2.2.3 formaliza cómo comparar el poder expresivo de dos lógicas.

Definición 2.2.3 ($\mathcal{L} \leq \mathcal{L}'$)**.** *Decimos que $\mathcal{L}'$ es* al menos tan expresivo como *$\mathcal{L}$ (notación $\mathcal{L} \leq \mathcal{L}'$) si existe una function* Tr *entre fórmulas de $\mathcal{L}$ y $\mathcal{L}'$ tal que para todo modelo $\mathcal{M}$ y toda fórmula φ de $\mathcal{L}$ tenemos que*

$$\mathcal{M} \models_{\mathcal{L}} \varphi \text{ sii } \mathcal{M} \models_{\mathcal{L}'} \mathsf{Tr}(\varphi).$$

$\mathcal{M}$ es visto como un modelo de $\mathcal{L}$ en la parte izquierda y como un modelo de $\mathcal{L}'$ en la derecha, usando en cada caso la relación semántica $\models_{\mathcal{L}}$ o $\models_{\mathcal{L}'}$ según corresponda.

Decimos que $\mathcal{L}$ y $\mathcal{L}'$ son incomparables *(notación $\mathcal{L} \neq \mathcal{L}'$) si $\mathcal{L} \not\leq \mathcal{L}'$ y $\mathcal{L}' \not\leq \mathcal{L}$.*

Decimos que $\mathcal{L}'$ es estrictamente más expresivo *que $\mathcal{L}$ (notación $\mathcal{L} < \mathcal{L}'$) si $\mathcal{L} \leq \mathcal{L}'$ pero no $\mathcal{L}' \leq \mathcal{L}$.*

La relación $\leq$ indica que podemos "incrustar" un lenguaje en otro. Para hacer esto, necesitamos una traducción que preserve equivalencia del primer lenguaje en el segundo. Su versión estricta es $<$, que indica que el segundo lenguaje puede expresar estrictamente más que el primero. La relación de incomparabilidad indica que ninguno de los dos lenaguajes puede ser incrustado en el otro, es decir, ellos son capaces de expresar cosas diferentes. En la siguiente sección vamos a usar estas definiciones para comparar el poder expresivo de las lógicas con operadores de cambio de accesibilidad.

2.3 COMPARANDO LENGUAJES

Hemos introducido las nociones que necesitamos para estudiar la expresividad de un lenguaje. Pero también hemos definido una manera de relacionar diferentes lenguajes de acuerdo con lo que son capaces de describir. Este es el trabajo de esta sección: vamos a tomar un par de lenguajes, y utilizando la maquinaria introducida anteriormente vamos a ser capaces de decir si uno de ellos es más (o igual) expresivo que el otro, o si son incomparables.

El primer caso, es comparar las lógicas modales con operadores de cambio con la lógica modal básica. Su falta de la tree y la finite model property ya nos dicen que son más expresivas que $\mathcal{ML}$, pero en esta sección ofrecemos una prueba más simple de este resultado. La prueba es directa de las definiciones: tenemos que encontrar

modelos bisimilares para $\mathcal{ML}$ que se puedan distinguir por las otras lógicas, dado que podemos trivialmente traducir $\mathcal{ML}$ en cualquiera de ellos.

Teorema 2.3.1. $\mathcal{ML} < \mathcal{ML}(\blacklozenge)$, *donde* $\blacklozenge \in \{\langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle\}$.

Demostración. La traducción de fórmulas de $\mathcal{ML}$ en $\mathcal{ML}(\blacklozenge)$ es trivial, ya que $\mathcal{ML}$ es un fragmento sintáctico de $\mathcal{ML}(\blacklozenge)$. Para demostrar $\mathcal{ML}(\blacklozenge) \not\leq \mathcal{ML}$ necesitamos dos modelos que sean bisimilares en $\mathcal{ML}$ y fórmulas de $\mathcal{ML}(\blacklozenge)$ que los distingan. Los modelos de la Figura 8 son $\mathcal{ML}$-bisimilares: w y w' satisfacen los mismo símbolos proposicionales (sus valuaciones son vacías), y en cada modelo siempre podemos encontrar sucesores bisimilares. La bisimulación Z relaciona w con todos los estados de $\mathcal{M}'$.

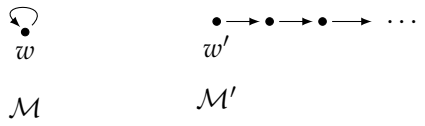


Figure 8: Modelos $\mathcal{ML}$-bisimilares.

Necesitamos encontrar un $\mathcal{ML}(\blacklozenge)$-fórmula φ, con $\blacklozenge \in \{\langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle\}$ tal que φ se satisface en uno de los modelos y no se satisface en el otro. Las fórmulas enumeradas a continuación se satisfacen en $\mathcal{M}', w'$ pero no en $\mathcal{M}, w$.

1. $\langle \mathsf{sb} \rangle \Diamond \top$. Claramente, después de borrar una arista adyacente desde w ya no tenemos sucesores, pero si tenemos desde w' (en $\mathcal{M}'$ tenemos una cadena infinita).

2. $\langle \mathsf{gsb} \rangle \Diamond \top$. Ídem que para el caso local.

3. $\langle \mathsf{sw} \rangle \Diamond \Box \bot$. Intercambiando aristas desde w, siempre obtenemos el mismo model variant, y w siempre tiene un sucesor. Empezando la evaluación desde w' e invirtiendo una arista, podemos volver a w' y alcanzar un dead end.

4. $\langle \mathsf{gsw} \rangle \Box \bot$. Ídem que para el caso local.

5. $\langle \mathsf{br} \rangle \top$. No hay componentes aisladas en el modelo de la izquiera, pero hay infinitas posibilidades donde agregar nuevas aristas en el modelo de la derecha.

6. $\langle \mathsf{gbr} \rangle \top$. Ídem que para el caso local.

□

La mayoría de las comparaciones han sido ya estudiadas en [Areces *et al.*, 2012], estableciendo que las lógicas modales con operadores de cambio son incomparables entre ellas. Esto no es trivial, teniendo en cuenta que podemos esperar algún tipo de conexión, al menos, entre las versiones locales y las versiones globales de cada operador. Sin embargo, veremos que la versiones locales y las globales nos permiten describir cosas diferentes. Algunos casos son fáciles de comprobar, pero hay otras en las que necesitamos estructuras más complejas para distinguir dos lenguajes.

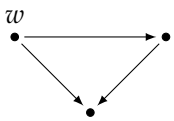
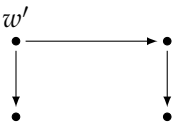
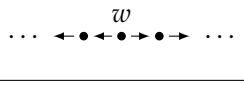
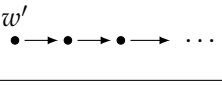
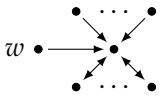
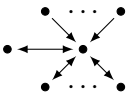
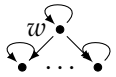
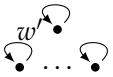
$\mathcal{M}$	$\mathcal{M}'$	Distinguibles por	Bisimilar para
		$\langle \mathsf{br} \rangle \langle \mathsf{br} \rangle \top$ $\langle \mathsf{gbr} \rangle \langle \mathsf{gbr} \rangle \top$ $\langle \mathsf{gsb} \rangle \top$ $\langle \mathsf{gsw} \rangle \top$	$\mathcal{ML}(\langle \mathsf{sb} \rangle)$ $\mathcal{ML}(\langle \mathsf{sw} \rangle)$
		$\langle \mathsf{sb} \rangle \Diamond \top$ $\langle \mathsf{gsb} \rangle \Diamond \top$	$\mathcal{ML}(\langle \mathsf{sw} \rangle)$ $\mathcal{ML}(\langle \mathsf{br} \rangle)$ $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$ $\mathcal{ML}(\langle \mathsf{gbr} \rangle)$
		$\langle \mathsf{sw} \rangle \Diamond \Diamond \Diamond \Box \bot$ $\Diamond \langle \mathsf{gsw} \rangle \Diamond \Diamond \Diamond \Box \bot$ $\langle \mathsf{br} \rangle \langle \mathsf{br} \rangle \top$ $\langle \mathsf{gbr} \rangle^6 \langle \mathsf{gbr} \rangle \top$	$\mathcal{ML}(\langle \mathsf{gsb} \rangle)$ $\mathcal{ML}(\langle \mathsf{sb} \rangle)$
		$\langle \mathsf{sw} \rangle \Diamond \Box \bot$ $\langle \mathsf{gsw} \rangle \Box \bot$	$\mathcal{ML}(\langle \mathsf{br} \rangle)$ $\mathcal{ML}(\langle \mathsf{gbr} \rangle)$
		$\langle \mathsf{sb} \rangle \Diamond \Box \bot$	$\mathcal{ML}(\langle \mathsf{gsb} \rangle)$
		$\langle \mathsf{br} \rangle^3 \top$ $\langle \mathsf{gbr} \rangle^3 \top$	$\mathcal{ML}(\langle \mathsf{gsw} \rangle)$
		$\langle \mathsf{br} \rangle \top$	$\mathcal{ML}(\langle \mathsf{gbr} \rangle)$

Figure 9: Modelos bisimilares y fórmulas que los distinguen.

Teorema 2.3.2. *Para todo* $\blacklozenge_1, \blacklozenge_2 \in \{\langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle\}$ *tal que* $\blacklozenge_1 \neq \blacklozenge_2$, $\mathcal{ML}(\blacklozenge_1)$ *y* $\mathcal{ML}(\blacklozenge_2)$, $\mathcal{ML}(\blacklozenge_1) \neq \mathcal{ML}(\blacklozenge_2)$ *(son incomparables), excepto por la comparación entre* $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$ *y* $\mathcal{ML}(\langle \mathsf{sw} \rangle)$.

Demostración. Para cada par de modelos $\mathcal{M}, w$ y $\mathcal{M}', w'$ en la Figura 9, y para toda correspondiente fórmula φ de la columna "Distinguibles por", tenemos que $\mathcal{M}, w \not\models \varphi$ y $\mathcal{M}', w' \models \varphi$. Además, para cada lenguaje $\mathcal{L}$ correspondiente de la columna "Bisimilares para", tenemos que (w, R) y (w', R') están relacionados por una $\mathcal{L}$-bisimulación, donde R y R' son las relacciones de accesibilidad de $\mathcal{M}$ y $\mathcal{M}'$ respectivamente. □

La única comparación que no ha sido obtenida es el caso $\mathcal{ML}(\langle \mathsf{sw} \rangle) \not\leq \mathcal{ML}(\langle \mathsf{gsw} \rangle)$. El resultado está todavía abierto, pero conjeturamos que estas dos lógicas también son incomparables.

Conjetura 2.3.3. $\mathcal{ML}(\langle \mathsf{sw} \rangle) \neq \mathcal{ML}(\langle \mathsf{gsw} \rangle)$.

2.4 UNA PERSPECTIVA MÁS GENERAL

Algunos trabajos relacionados con el poder expresivo de los lenguajes presentados en esta tesis quedan pendientes. Hemos definido una noción de bisimulación que captura exactamente el comportamiento de los operadores de cambio de accesibilidad. Para cada lenguaje, bisimulación se define como una relación que une los estados y la relación de accesibilidad actualizada de los modelos, y demostramos que dos modelos bisimilares satisfacen las mismas fórmulas del lenguaje correspondiente. Tenemos condiciones específicas en función de los operadores que estamos considerando, pero las condiciones se definen de manera uniforme. Es posible definir una noción más general de bisimulación que en lugar de considerar los casos particulares de las actualizaciones (borrar, intercambiar o añadir aristas) podría considerar funciones de actualización en general. Sería fácil mostrar que, bajo ciertas circunstancias, podemos utilizar las mismas ideas que para los operadores sabotage, swap y bridge para definir una noción de bisimulación adecuada para cada modificador.

Discutimos los resultados generales sobre bisimulación, pero las mismas ideas también se puede aplicar a casos particulares. Otros modificadores modales se han investigado, por ejemplo, en Lógica Dinámica Epistémica. En algunos casos, los lenguajes no aumentan el poder expresivo de $\mathcal{ML}$ (pueden ser traducidos a $\mathcal{ML}$ a través de *axiomas de reducción*. Como consecuencia de ello, la noción de bisimulación es la misma que para $\mathcal{ML}$. Sin embargo, hay otros casos en los que los cambios producidos por el operador no se pueden expresar en $\mathcal{ML}$. Para estos lenguajes, conjeturamos que podemos aplicar las técnicas generales como para lógicas de cambio de accesibilidad para obtener una noción adecuada de bisimulación.

Otro problema interesante para ser analizado es *Caracterización de van Benthem* [van Benthem, 1977]. El Teorema de caracterización original establece que una fórmula φ de $\mathcal{FOL}$ es equivalente a la traducción de una fórmula de $\mathcal{ML}$ si y sólo si φ es invariante por bisimulación. En [Areces *et al.*, 2013a] se mostró que bajo ciertas condiciones de adecuidad, el teorema de caracterización de van Benthem vale. Sería interesante verificar dichas condiciones de adecuidad e investigar caracterización para las lógicas con operadores que cambian la accesibilidad.

3

Satisfactibilidad

The Entscheidungsproblem is solved when we know a procedure that allows for any given logical expression to decide by finitely many operations its validity or satisfiability. (...) The Entscheidungsproblem must be considered the main problem of mathematical logic.

from "Grundzüge der theoretischen Logik", David Hilbert and Wilhelm Ackerman.

Un problema clásico en lógica a ser investigado es el *problema de satisfactibilidad*, es decir, dado una fórmula del lenguaje decidir si existe un modelo que la satisface. En este capítulo investigamos dicho problema para los casos locales de las lógicas que introdujimos. Vamos a mostrar que el problema de satisfactibilidad para estas lógicas es indecidible. Para $\mathcal{ML}(\langle \mathsf{sw} \rangle)$ esto ya fue demostrado en [Areces *et al.*, 2013b].

En primer lugar, traducimos modelos de lógicas con memoria a modelos de Kripke estándar. La idea de la traducción es que obliga a alguna restricción en la forma de los modelos en los que vamos a evaluar las fórmulas. Los modelos que queremos forzar, se basan en la estructura definida para los modelos infinitos de Sección 1.4. A continuación, traducimos fórmulas de $\mathcal{ML}(ⓡ, ⓚ)$ a las fórmulas con operadores de cambio. En las siguientes secciones se introducen las traducciones para las versiones locales de sabotage, bridge y swap, respectivamente. El comportamiento de swap es muy diferente y más difícil de capturar. Como veremos también en otros capítulos, las construcciones para swap son siempre más difíciles que para los demás operadores.

3.1 SABOTAGE LOGIC

Comenzamos demostrando que el problema de decidir satisfactibilidad para $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ es indecidible. En primer lugar, introducimos una traducción de las fórmulas de la lógica con memoria $\mathcal{ML}(ⓡ, ⓚ)$ a fórmulas de $\mathcal{ML}(\langle \mathsf{sb} \rangle)$. Con el fin de simular el comportamiento de los operadores ⓡ and ⓚ sin tener una memoria en el modelo, imponemos restricciones en los modelos en los que evaluamos la fórmula traducida. Luego probamos que una $\mathcal{ML}(ⓡ, ⓚ)$-fórmula es satisfactible si y sólo si la traducción de esta fórmula (además a las condiciones que definimos) es satisfactible.

Definición 3.1.1. *Sea $s \in \mathsf{PROP}$, definimos Conds como la conjunción de las siguientes fórmulas:*

$$
\begin{array}{ll}
(1) & s \wedge \Box\neg s \wedge \Box\Diamond s \\
(2) & \Box\Box(s \to \neg\Diamond s) \\
(3) & [\mathsf{sb}][\mathsf{sb}](s \to \Box\Diamond s) \\
(4) & \Box[\mathsf{sb}](s \to \Diamond\neg\Diamond s) \\
(5) & \Box\Box(\neg s \to \Diamond(s \wedge \neg\Diamond s)) \\
(6) & \Box[\mathsf{sb}](\neg s \to [\mathsf{sb}](s \to \Box\Box(\neg s \to \Diamond s))) \\
(7) & \Box\Box(\neg s \to [\mathsf{sb}](s \to \Diamond\Diamond(\neg s \wedge \neg\Diamond s))) \\
(Spy) & \Box\Box(\neg s \to [\mathsf{sb}](s \to \Diamond\neg\Diamond s)).
\end{array}
$$

Llamemos s (por *spy point*) a un estado que satisface *Conds* en un modelo arbitrario. Entonces el punto s satisface el símbolo proposicional s y está relacionado con todos los estados de la componente conexa del modelo en ambas direcciones. La fórmula (1) asegura que el símbolo proposicional s se satisface en el punto de evaluación, y no se satisface en ningún sucesor. Como consecuencia, s es irreflexivo. Esto también nos dice que todos los sucesores pueden ver un estado que satisface s.

(2) asegura que todos los estados s que son accesibles en dos pasos desde el punto de evaluación, no tienen sucesores satisfaciendo s.

(3) asegura que después de eliminar dos aristas y habiendo alcanzado un estado que satisface s, la propiedad de que todos los sucesores pueden ver un estado s se mantiene.

La fórmula (4) establece que para todos los sucesores, después de eliminar una arista y alcanzando un estado s, existe un sucesor que no puede ver ningún estado s (él era el único sucesor satisfaciendo s).

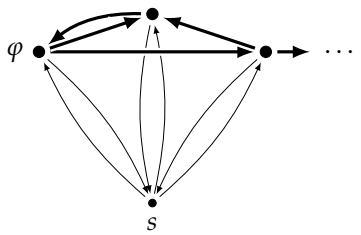
(5) dice que alcanzando en dos pasos algún estado que no satisface s, siempre existe un estado s que es alcanzable y que no tiene sucesores que satisfacen s.

(6) asgura que después de eliminar una arista desde una estado que no satisface s (que ya no es accesible desde el punto de evaluacón en dos pasos) hacia un estado que satisface s, los demás estados que no satisfacen s tienen todavía una arista apuntando hacia un estado que sí satisface s.

(7) asegura que todos los estados que son alcanzables en dos pasos y que no satisfacen s tiene sólo un sucesor etiquetado por s.

(Spy) establece que todos los estados que son accesibles en dos pasos son también alcanzables en un paso.

A continuación podemos ver un ejemplo del uso de *Conds*. La idea es tomar un modelo de $\mathcal{ML}(ⓡ, ⓚ)$, y agregar un spy point para satisfacer *Conds*. El siguiente es un modelo tal que $\mathcal{M}, s \models \mathit{Conds}$:



Proposición 3.1.2. *Sea $\mathcal{M} = \langle W, R, V\rangle$ un modelo, $w \in W$. Si $\mathcal{M}, w \models Conds$, entonces las siguientes propiedades se satisfacen:*

1. *w es el único estado en $\mathcal{M}$ que satisface s en la componente conexa generada por w.*

2. *Para todo $v \in W$ tal que $v \neq w$, tenemos que si $(w, v) \in R$ entonces $(v, w) \in R$, y si $(w, v) \in R^*$ entonces $(w, v) \in R$ (w es un spy point).*

Ahora introducimos la traducción de fórmulas de $\mathcal{ML}(ⓡ, ⓚ)$ a fórmulas de $\mathcal{ML}(\langle \mathsf{sb}\rangle)$.

Definición 3.1.3. *Sea φ una fórmula de $\mathcal{ML}(ⓡ, ⓚ)$ tal que no contiene el símbolo proposicional s. Definimos $\mathsf{Tr}(\varphi) = \Diamond(\varphi)'$, donde $(\ )'$ está definida como:*

$$
\begin{array}{lcl}
(p)' & = & p \quad \textit{para } p \in \textit{PROP que aparece in } \varphi \\
(ⓚ)' & = & \neg\Diamond s \\
(\neg\psi)' & = & \neg(\psi)' \\
(\psi \wedge \chi)' & = & (\psi)' \wedge (\chi)' \\
(\Diamond\psi)' & = & \Diamond(\neg s \wedge (\psi)') \\
(ⓡ\psi)' & = & (\Diamond s \rightarrow \langle \mathsf{sb}\rangle(s \wedge \langle \mathsf{sb}\rangle(\neg\Diamond s \wedge (\psi)'))) \ \wedge \ (\neg\Diamond s \rightarrow (\psi)').
\end{array}
$$

Los casos Booleanos y modales son obvios. ⓡ se representa eliminando las aristas desde el spy point hacia el estado que queremos memorizar, y desde dicho estado hacia el spy point. Si el punto ya ha sido memorizado ($\neg\Diamond s$) no queda nada por hacer; caso contrario ($\Diamond s$), hacemos que s sea inaccesible desde y hacia el estado corriente usando $\langle \mathsf{sb}\rangle$. ⓚ se representa verificando si existe una arista hacia el spy point o no.

Teorema 3.1.4. *Sea φ una fórmula de $\mathcal{ML}(ⓡ, ⓚ)$ que no contiene el símbolo proposicional s. Entonces, φ y $\mathsf{Tr}(\varphi) \wedge Conds$ son equisatisfactibles.*

Demostración. Vamos a demostrar que φ es satisfactible si y sólo si $\mathsf{Tr}(\varphi) \wedge Conds$ es satisfactible.

($\Leftarrow$) Supongamos que $\mathsf{Tr}(\varphi) \wedge Conds$ es satisfactible, es decir, existe un modelo $\mathcal{M} = \langle W, R, V\rangle$, y $s \in W$ tal que $\langle W, R, V\rangle, s \models \mathsf{Tr}(\varphi)$ y $\langle W, R, V\rangle, s \models Conds$. Entonces podemos definir el modelo $\mathcal{M}' = \langle W', R', V', \emptyset\rangle$ donde

$$
\begin{array}{ll}
W' & = \{v \mid (s, v) \in R\} \\
R' & = R \cap (W' \times W') \\
V'(p) & = V(p) \cap W' \quad \text{para } p \in \mathsf{PROP}.
\end{array}
$$

Sea $w' \in W'$ un estado t.q. $(s, w') \in R$ y $\langle W, R, V\rangle, w' \models (\varphi)'$ (dado que $\langle W, R, V\rangle, s \models \Diamond(\varphi)'$). Vamos a demostrar

$$\langle W', R', V', M'\rangle, v \models \psi \ \text{ sii } \ \langle W, R(M'), V\rangle, v \models (\psi)',$$

donde $v \in W'$, $M' \subseteq W'$, $\psi \in \mathsf{FORM}$, y $R(M') = R \backslash \{(s, t), (t, s) \mid t \in M'\}$. En particular, cuando $M' = \emptyset$ tenemos que $\langle W', R', V', \emptyset\rangle, w' \models \varphi$ sii $\langle W, R, V\rangle, w' \models (\varphi)'$.

Entonces hacemos inducción estructural en ψ. Tenemos dos casos base:

$\psi = p$: Supongamos que $\langle W', R', V', M' \rangle, v \models p$. Por definición de $\models$, $v \in V'(p)$, y esto es equivalente $v \in V(p) \cap W'$ por definición de V'. Dado que $v \in V(p)$, por $\models$ tenemos $\langle W, R(M'), V \rangle, v \models p$, y por definición de $(\ )'$ esto es equivalente a $\langle W, R(M'), V \rangle, v \models (p)'$.

$\psi = \textcircled{k}$: Supongamos que $\langle W', R', V', M' \rangle, v \models \textcircled{k}$. Por $\models$ tenemos $v \in M'$, y por Proposición 3.1.2 y definición de $R(M')$ tenemos $(v, s) \notin R(M')$ y $\langle W, R(M'), V \rangle, s \models s$. Entonces por $\models \langle W, R(M'), V \rangle, v \models \neg\Diamond s$, y por definición de $(\ )'$ esto es equivalente a $\langle W, R(M'), V \rangle, v \models (\textcircled{k})'$.

Ahora demostramos los casos inductivos.

$\psi = \neg\phi$: Supongamos $\langle W', R', V', M' \rangle, v \models \neg\phi$. Por definición de $\models$, tenemos que $\langle W', R', V', M' \rangle, v \not\models \phi$. Por H.I., tenemos $\langle W, R(M'), V \rangle, v \not\models (\phi)'$, sii $\langle W, R(M'), V \rangle, v \models \neg(\phi)'$. Entonces, por definición de $(\ )'$, $\langle W, R(M'), V \rangle, v \models (\neg\phi)'$.

$\psi = \phi \wedge \chi$: Supongamos $\langle W', R', V', M' \rangle, v \models \phi \wedge \chi$. Por $\models$, $\langle W', R', V', M' \rangle, v \models \phi$ y $\langle W', R', V', M \rangle, v \models \chi$. Por H.I., $\langle W, R(M'), V \rangle, v \models (\phi)'$ y $\langle W, R(M'), V \rangle, v \models (\chi)'$. Entonces tenemos $\langle W, R(M'), V \rangle, v \models (\phi)' \wedge (\chi)'$. Entonces por definición de $(\ )'$, $\langle W, R(M'), V \rangle, v \models (\phi \wedge \chi)'$.

$\psi = \Diamond\phi$: Supongamos $\langle W, R(M'), V \rangle, v \models (\Diamond\phi)'$. Por definición de $(\ )'$, tenemos $\langle W, R(M'), V \rangle, v \models \Diamond(\neg s \wedge (\phi)')$. Por $\models$, existe $v' \in W$ t.q. $(v, v') \in R(M')$ y $\langle W, R(M'), V \rangle, v'$
$\models \neg s \wedge (\phi)'$. Entonces tenemos $\langle W, R(M'), V \rangle, v' \models \neg s$ y $\langle W, R(M'), V \rangle, v' \models (\phi)'$. Por H.I., $\langle W', R', V', M' \rangle, v' \models \phi$, luego por $\models$ and Proposición 3.1.2, tenemos $\langle W', R', V', M' \rangle, v \models \Diamond\phi$.

$\psi = \textcircled{r}\phi$: Supongamos $\langle W, R(M'), V \rangle, v \models (\textcircled{r}\phi)'$. Por definición de $(\ )'$ y $\models$,

$$\langle W, R(M'), V \rangle, v \models \Diamond s \to \langle \mathsf{sb} \rangle (s \wedge \langle \mathsf{sb} \rangle (\neg\Diamond s \wedge (\phi)')) \text{ and}$$
$$\langle W, R(M'), V \rangle, v \models \neg\Diamond s \to (\phi)'.$$

Vamos a demostrar cada conjunto por separado. Primero supongamos que $\langle W, R(M'), V \rangle, v \models \Diamond s$. Entonces $(v, s) \in R(M')$ (por Proposición 3.1.2). Queremos demostrar que $\langle W, R(M'), V \rangle, v \models \langle \mathsf{sb} \rangle (s \wedge \langle \mathsf{sb} \rangle (\neg\Diamond s \wedge (\phi)'))$. Nosotros asumimos que $(v, s) \in R(M')$ entonces $(s, v) \in R(M')$, porque en $R(M')$ siempre eliminamos pares en las dos direcciones, y por Proposición 3.1.2. Solo necesitamos demostrar $\langle W, R(M')^{-}_{vs}, V \rangle, s \models s \wedge \langle \mathsf{sb} \rangle (\neg\Diamond s \wedge (\phi)')$. Es trivial que $\langle W, (R(M'))^{-}_{vs}, V \rangle, s \models s$. Veamos que $\langle W, (R(M'))^{-}_{vs}, V \rangle, s \models \langle \mathsf{sb} \rangle (\neg\Diamond s \wedge (\phi)')$. Dado que $(s, v) \in (R(M'))^{-}_{vs}$, es suficiente demostrar que $\langle W, R(M')^{-}_{vs,sv}, V \rangle, v \models \neg\Diamond s \wedge (\phi)'$. El primer conjunto es trivial ya que $(v, s) \notin R(M')^{-}_{vs,sv}$.

Por otro lado, sabemos que para todo t, $(t, s) \notin R(M')$ sii $(s, t) \notin R(M')$. Entonces por H.I., $\langle W, R(M')^{-}_{vs,sv}, V \rangle, v \models (\phi)'$ sii $\langle W', R', V', M \cup \{v\} \rangle, v \models \phi$. Luego, por $\models$ tenemos $\langle W', R', V', M' \rangle, v \models \textcircled{r}\phi$.

Ahora supongamos el otro caso, $\langle W, R(M'), V \rangle, v \models \neg\Diamond s$. Por Proposición 3.1.2, sabemos que $(v, s) \notin R(M')$. Por definición de $R(M')$, tenemos $(s, v) \notin R(M')$. Entonces $v \in M'$, y por H.I. tenemos $\langle W', R', V', M' \rangle, v \models \textcircled{r}\phi$.

($\Rightarrow$) Supongamos que φ es satisfactible, es decir, existe un modelo $\mathcal{M} = \langle W, R, V, \emptyset \rangle$ y $w \in W$ tal que $\langle W, R, V, \emptyset \rangle, w \models \varphi$.

Sea s un estado que no pertenece a W. Entonces podemos definir el modelo $\mathcal{M}' = \langle W', R', V' \rangle$ como:

$$\begin{array}{ll} W' & = W \cup \{s\} \\ R' & = R \cup \{(s,w) \mid w \in W\} \cup \{(w,s) \mid w \in W\} \\ V'(p) & = V(p) \quad \text{for } p \in \mathsf{PROP} \text{ appearing in } \varphi \\ V'(s) & = \{s\}. \end{array}$$

Por construcción de $\mathcal{M}'$, es fácil verificar que $\mathcal{M}', s \models \mathit{Conds}$. Entonces podemos verificar que

$$\langle W, R, V, M \rangle, w \models \varphi \text{ sii } \langle W', R'(M), V' \rangle, s \models \mathsf{Tr}(\varphi),$$

donde $R'(M)$ está definido de la misma manera que para la dirección ($\Leftarrow$) de la demostración.

Otra vez podemos hacer inducción estructural. Los casos Booleanos y Ⓚ son triviales. Si $\langle W, R, V, M \rangle, w \models \Diamond\psi$, entonces por construcción de $\mathcal{M}'$ es claro que $w \notin V'(s)$ y $\langle W', R'(M), V' \rangle, v \models (\psi)'$. Si $\langle W, R, V, M \rangle, w \models \textcircled{r}\psi$, podemos eliminar las aristas (w,s) y (s,w) para simular el almacenamiento de w en la memoria (si tales pares no están en R' significa que $w \in M$) y continuamos la evaluación del resto de la traducción $(\ )'$ (los pasos son similares a los de la dirección ($\Leftarrow$) de la demostración). □

Del teorema anterior inmediatamente tenemos:

Teorema 3.1.5. *El problema de decidir satisfactibilidad para $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ es indecidible.*

Hemos demostrado que podemos reducir el problema de satisfactibilidad de $\mathcal{ML}(\textcircled{r}, \textcircled{k})$ al mismo problema para $\mathcal{ML}(\langle \mathsf{sb} \rangle)$. Basados en este resultado, y en el hecho de que con fórmulas de $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$ podemos forzar modelos infinitos, conjeturamos que las mismas construcciones pueden ser realizadas para la versión global de sabotage, dejando esto como trabajo futuro.

3.2 BRIDGE LOGIC

En esta sección, vamos a reducir el problema satisfactibilidad de $\mathcal{ML}(\textcircled{r}, \textcircled{k})$ al problema satisfactibilidad de $\mathcal{ML}(\langle \mathsf{br} \rangle)$. La idea para simular el comportamiento de Ⓡ es mantener un "spy point" y agregar aristas en ambas direcciones hacia el punto a ser memorizado. Entonces, el comportamiento de Ⓚ es capturado consultando si el estado actual tiene una arista hacia el spy point. En este caso, vamos a permitir varios spy points, los cuales están desconectados del resto del modelo, formando una componente conexa entre ellos. Sin embargo, las restricciones que imponemos utilizan siempre el mismo spy point. Otra diferencia con $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ es que utilizamos un segundo conjunto de estados (llamados "bridge points") entre los spy points y el resto del modelo, también sin conexión con el resto, pero conectados entre ellos. Debido a que los spy point están aislados, tenemos que añadir nuevas aristas para llegar al resto del modelo. Si añadimos directamente una arista a un estado arbitrario,

cuando comenzamos la evaluación de una fórmula, estamos introduciendo alguna información no deseada (recordemos que las nuevas aristas representan estados visitados). Por esa razón utilizamos bridge points para iniciar la evaluación, sin introducir aristas no deseadas.

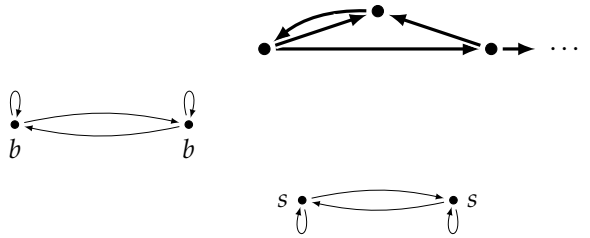
A continuación introducimos las condiciones para forzar los modelos que queremos:

Definición 3.2.1. *Sean $s, b \in \mathsf{PROP}$, definimos Conds como la conjunción de las siguientes fórmulas:*

$$
\begin{array}{ll}
(1) & s \wedge [\mathsf{br}]\neg s \\
(s-connex) & \Box[\mathsf{br}]\neg s \\
(3) & \Box\Box s \\
(4) & \Box[\mathsf{br}]\Box\neg s \\
(5) & \Box\neg b \\
(6) & \langle \mathsf{br}\rangle(b \wedge [\mathsf{br}]\neg b \\
 & \wedge \Box[\mathsf{br}]\neg b \\
 & \wedge \Box\Box b \\
 & \wedge \Box[\mathsf{br}]\Box\neg b \\
 & \wedge \Box\neg s).
\end{array}
$$

(1) establece que el punto de evaluación satisface s y es reflexivo. $(s-connex)$ asegura que no existe ningún estado s desconectado. (3) asegura que todos los sucesores tienen solo sucesores s. (4) dice que no hay aristas desde estados que satisfacen $\neg s$ hacia la componente conexa que satisface s. (5) asegura que no existe ningún sucesor de s que satisfaga b. Por último, la fórmula (6) establece que existe una componente conexa de estados que satisfacen b (descripto exactamente como para s) que es inalcanzable desde la componente s.

La idea es tomar un modelo de $\mathcal{ML}(ⓡ, ⓚ)$, y agregar una nube de spy points y una nube de bridge points para satisfacer *Conds*. Un modelo tal que $\mathcal{M}, s \models Conds$ es ilustrado a continuación:



La siguiente proposición explica la forma de los modelos que queremos forzar.

Proposición 3.2.2. *Sea $\mathcal{M} = \langle W, R, V\rangle$ un modelo, $w \in W$. Si $\mathcal{M}, w \models Conds$, entonces:*

1. *$w \in V(s)$ y $V(s)$ es una componente conexa (para todo $v, u \in V(s)$, tenemos $(v, u) \in R$ y $(u, v) \in R$).*
2. *Si $w \in V(s)$ y $v \notin V(s)$ entonces $(w, v) \notin R$ y $(v, w) \notin R$.*

3. *$w \notin V(b)$ y $V(b)$ es una componente conexa (para todo $v, u \in V(b)$, tenemos $(v,u) \in R$ y $(u,v) \in R$).*

4. *Si $w \notin V(b)$ y $v \in V(b)$ entonces $(w,v) \notin R$ y $(v,w) \notin R$.*

Ahora definimos la traducción de fórmulas de $\mathcal{ML}(ⓡ, ⓚ)$ a fórmulas de $\mathcal{ML}(\langle \mathsf{br} \rangle)$.

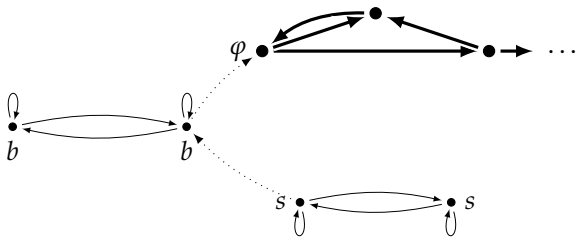
Definición 3.2.3. *Sea φ una fórmula de $\mathcal{ML}(ⓡ, ⓚ)$ que no contiene los símbolos proposicionales s y b. Definimos $\mathsf{Tr}(\varphi) = \langle \mathsf{br} \rangle (b \wedge \langle \mathsf{br} \rangle (\neg s \wedge (\varphi)'))$, donde $(\ )'$ está definida como:*

$$
\begin{array}{lcl}
(p)' & = & p \quad \text{para } p \in \mathit{PROP} \text{ que aparece en } \varphi \\
(ⓚ)' & = & \Diamond s \\
(\neg \psi)' & = & \neg (\psi)' \\
(\psi \wedge \chi)' & = & (\psi)' \wedge (\chi)' \\
(\Diamond \psi)' & = & \Diamond (\neg s \wedge \neg b \wedge (\psi)') \\
(ⓡ \psi)' & = & (\neg \Diamond s \rightarrow \langle \mathsf{br} \rangle (s \wedge \Diamond \neg s \wedge \langle \mathsf{br} \rangle (\Diamond s \wedge (\psi)'))) \ \wedge \ (\Diamond s \rightarrow (\psi)').
\end{array}
$$

Los casos Booleanos son triviales. $\Diamond \psi$ se satisface si existe un sucesor que satisfaga ψ, y verificamos también que tal sucesor no satisface s y b. ⓡ se representa agregando una arista entre el estado a ser memorizado y algún estado s, en las dos direcciones. De esta manera verificamos si un estado ha sido visitado verificando que existe una arista hacia un estado s.

Teorema 3.2.4. *Sea φ una fórmula de $\mathcal{ML}(ⓡ, ⓚ)$ que no contiene los símbolos proposicionales s y b. Entonces, φ y $\mathsf{Tr}(\varphi) \wedge Conds$ son equisatisfactibles.*

Un modelo tal que $\mathcal{M}, s \models Conds \wedge Tr(\varphi)$ es ilustrado a continuación:



Demostración. Vamos a demostrar que φ es satisfactible si y sólo si $\mathsf{Tr}(\varphi) \wedge Conds$ es satisfactible.

($\Leftarrow$) Supongamos que $\mathsf{Tr}(\varphi) \wedge Conds$ es satisfactible, es decir, existe un modelo $\mathcal{M} = \langle W, R, V \rangle$, y $s \in W$ tal que $\langle W, R, V \rangle, s \models \mathsf{Tr}(\varphi)$ y $\langle W, R, V \rangle, s \models Conds$. Entonces podemos definir el modelo $\mathcal{M}' = \langle W', R', V', \emptyset \rangle$, donde

$$
\begin{array}{ll}
W' & = \{v \mid (s,v) \notin R\} \\
R' & = R \cap (W' \times W') \\
V'(p) & = V(p) \cap W' \quad \text{para } p \in \mathsf{PROP}.
\end{array}
$$

Sea $w' \in W'$ un estado tal que existe $w'' \in W$ s.t $(s, w'') \in R$ y $(w'', w') \in R$ y $\langle W, R, V \rangle, w' \models (\varphi)'$ (dado que $\langle W, R, V \rangle, s \models \langle \mathsf{br} \rangle (b \wedge \langle \mathsf{br} \rangle (\neg s \wedge (\varphi)'))$). Vamos a demostrar

$$\langle W', R', V', M' \rangle, v \models \psi \ \text{ sii } \ \langle W, R(M'), V \rangle, v \models (\psi)'$$

donde $v \in W'$, $M' \subseteq W'$, $\psi \in \mathsf{FORM}$, y $R(M') = R \cup \{(s,t),(t,s) \mid t \in M'\}$. En particular, entonces $M' = \emptyset$ tenemos que $\langle W', R', V', \emptyset\rangle, w' \models \varphi$ sii $\langle W, R, V\rangle, w' \models (\varphi)'$.

La demostración es por inducción estructural en ψ. Tenemos dos casos base:

$\boldsymbol{\psi = p}$: Supongamos que $\langle W', R', V', M'\rangle, v \models p$. Por $\models$ tenemos $v \in V'(p)$, y esto es equivalente a $v \in V(p) \cap W'$ por definición de V'. Dado que $v \in V(p)$, por $\models$ tenemos $\langle W, R(M'), V\rangle, v \models p$, y Por definición de $(\ )'$ esto es equivalente a $\langle W, R(M'), V\rangle, v \models (p)'$.

$\boldsymbol{\psi = \textcircled{k}}$: Supongamos que $\langle W', R', V', M'\rangle, v \models \textcircled{k}$. Por $\models$ tenemos $v \in M'$, y por Proposición 3.2.2 y definición de $R(M')$ tenemos $(v,s) \in R(M')$ y $\langle W, R(M'), V\rangle, s \models s$. Luego por $\models \langle W, R(M'), V\rangle, v \models \Diamond s$, y por definición de $(\ )'$ esto es equivalente a $\langle W, R(M'), V\rangle, v \models (\textcircled{k})'$.

Ahora demostramos los casos inductivos.

$\boldsymbol{\psi = \neg\phi}$ and $\boldsymbol{\psi = \phi \wedge \chi}$: por H.I.

$\boldsymbol{\psi = \Diamond\phi}$: Supongamos $\langle W, R(M'), V\rangle, v \models (\Diamond\phi)'$. Por definición de $(\ )'$ tenemos $\langle W, R(M'), V\rangle, v \models \Diamond(\neg s \wedge \neg b \wedge (\phi)')$. Por $\models$, existe $v' \in W$ t.q. $(v,v') \in R(M')$ y $\langle W, R(M'), V\rangle, v' \models \neg s \wedge \neg b \wedge (\phi)'$. Entonces (por $\models$) $\langle W, R(M'), V\rangle, v' \models \neg s$, $\langle W, R(M'), V\rangle, v' \models \neg b$ y $\langle W, R(M'), V\rangle, v' \models (\phi)'$. Por H.I. tenemos $\langle W', R', V', M'\rangle, v' \models \phi$, luego por $\models$ y Proposición 3.2.2, $\langle W', R', V', M'\rangle, v \models \Diamond\phi$.

$\boldsymbol{\psi = \textcircled{r}\phi}$: Supongamos $\langle W, R(M'), V\rangle, v \models (\textcircled{r}\phi)'$. Por definición de $(\ )'$ y $\models$,

$$\langle W, R(M'), V\rangle, v \models \neg\Diamond s \to \langle \mathsf{br}\rangle(s \wedge \Diamond\neg s \wedge \langle \mathsf{br}\rangle(\Diamond s \wedge (\phi)')) \text{ and}$$
$$\langle W, R(M'), V\rangle, v \models \Diamond s \to (\phi)'.$$

Vamos a demostrar cada conjunto de manera separada. Primero, supongamos $\langle W, R(M'), V\rangle, v \models \neg\Diamond s$. Entonces $(v,s) \notin R(M')$ (por Proposición 3.2.2). Queremos demostrar que $\langle W, R(M'), V\rangle, v \models \langle \mathsf{br}\rangle(s \wedge \Diamond\neg s \wedge \langle \mathsf{br}\rangle(\Diamond s \wedge (\phi)'))$. Por asunción y por Proposición 3.2.2, sabemos que si $(v,s) \notin R(M')$ entonces $(s,v) \notin R(M')$ (porque en $R(M')$ siempre agregamos pares en las dos direcciones). Entonces sólo necesitamos demostrar que $\langle W, R(M')^+_{vs}, V\rangle, s \models s \wedge \Diamond\neg s \wedge \langle \mathsf{br}\rangle(\Diamond s \wedge (\phi)')$. Sabemos que $s \in V(s)$, entonces resulta obvio que $\langle W, (R(M'))^+_{vs}, V\rangle, s \models s$. Luego $\langle W, (R(M'))^+_{vs}, V\rangle, s \models \Diamond\neg s$, dado que la evaluación de la traducción empezó en s y agregando una arista hacia un estado que satisface b (el cual no satisface s, por Proposición 3.2.2). Queda por demostrar que $\langle W, (R(M'))^+_{vs}, V\rangle, s \models \langle \mathsf{br}\rangle(\Diamond s \wedge (\phi)')$. Dado que $(s,v) \notin (R(M'))^+_{vs}$, es suficiente demostrar $\langle W, R(M')^+_{vs,sv}, V\rangle, v \models \Diamond s \wedge (\phi)'$. Por un lado, el primer conjunto es trivial porque $(v,s) \in R(M')^+_{vs,sv}$. Por otro lado, sabemos que para todo t, $(t,s) \in R(M')$ sii $(s,t) \in R(M')$. Entonces, por H.I., $\langle W, R(M')^+_{vs,sv}, V\rangle, v \models (\phi)'$ sii $\langle W', R', V', M' \cup \{v\}\rangle, v \models \phi$. Luego, por $\models$ tenemos $\langle W', R', V', M'\rangle, v \models \textcircled{r}\phi$.

Ahora supongamos el otro caso, $\langle W, R(M'), V\rangle, v \models \Diamond s$. Entonces sabemos que $(v,s) \in R(M')$ (por Proposición 3.2.2). Por definición de $R(M')$, tenemos $(s,v) \in R(M')$. Luego $v \in M'$, and por H.I. tenemos $\langle W', R', V', M'\rangle, v \models \textcircled{r}\phi$.

($\Rightarrow$) Supongamos que φ es satisfactible,es decir, existe un modelo $\mathcal{M}=\langle W, R, V, \emptyset\rangle$ y $w \in W$ tal que $\langle W, R, V, \emptyset\rangle, w \models \varphi$.

Sean s y b dos estados que no pertenecen a W. Entonces podemos definir el modelo $\mathcal{M}' = \langle W', R', V'\rangle$ como:

$$\begin{array}{ll} W' & = W \cup \{s\} \cup \{b\} \\ R' & = R \cup \{(s,s),(b,b)\} \\ V'(p) & = V(p) \quad \text{para } p \in \mathsf{PROP} \text{ que aparece en } \varphi \\ V'(s) & = \{s\} \\ V'(b) & = \{b\}. \end{array}$$

Por construcción de $\mathcal{M}'$, es fácil verificar que $\mathcal{M}', s \models \mathit{Conds}$. Entonces podemos ver que

$$\langle W, R, V, M\rangle, w \models \varphi \text{ sii } \langle W', R'(M), V'\rangle, s \models \mathsf{Tr}(\varphi),$$

donde $R'(M)$ está definido de la misma manera que para la dirección ($\Leftarrow$).

De nuevo necesitamos inducción estructural. Los casos Booleanos y Ⓚ son triviales. Si $\langle W, R, V, M\rangle, w \models \Diamond\psi$, entonces por construcción de $\mathcal{M}'$ es claro que $w \notin V'(s)$ y $w \notin V'(b)$ y $\langle W', R'(M), V'\rangle, v \models (\psi)'$. Si $\langle W, R, V, M\rangle, w \models ⓡ\psi$, podemos agregar las aristas (w,s) y (s,w) para simular el almacenamiento de w en la memoria (si dichos pares están en R' significa que $w \in M$) y continuamos la evaluación del resto de la traducción $(\ )'$ (los pasos son similares a los de la dirección ($\Leftarrow$) de la demostración). □

Como consecuencia obtenemos:

Teorema 3.2.5. *El problema de decidir satisfactibilidad para $\mathcal{ML}(\langle \mathsf{br}\rangle)$ es indecidible.*

Las dos demostraciones presentadas hasta el momento siguen las mismas ideas. Como mencionamos en la sección previa para el caso de $\mathcal{ML}(\langle \mathsf{sb}\rangle)$, conjeturamos que podemos aplicar las mismas técnicas para demostrar indecidibilidad para la versión global ($\mathcal{ML}(\langle \mathsf{gbr}\rangle)$). Ésto es dejado también como trabajo futuro.

3.3 SWAP LOGIC

Una vez mas vamos a simular el comportamiento de $\mathcal{ML}(ⓡ, Ⓚ)$, es decir, la capacidad de memorizar y verificar estados sin tener una memoria externa. Lo que tenemos en cambio es la capacidad de invertir aristas en el modelo. Vamos a construir models que contengan *switches*, o sea, aristas especiales cuya posición –"apagado" por defecto, y "encendido" si la dirección de la arista ha sido invertida – representará cuando un punto del modelo ha sido memorizado con el operador ⓡ. Vamos a simular el predicado Ⓚ consultando la posición de los switches. Introduzcamos la traducción de $\mathcal{ML}(ⓡ, Ⓚ)$ a $\mathcal{ML}(\langle \mathsf{sw}\rangle)$.

Definición 3.3.1. *Sea φ una fórmula de $\mathcal{ML}(ⓡ, Ⓚ)$ que no contiene los símbolos proposicionales s y sw. Sea $\mathsf{Tr}(\varphi)$ la siguiente fórmula:*

$$\begin{array}{lll} (1) & & s \wedge \Box^{(4)} \neg s \\ (2) & \wedge & \neg sw \ \wedge \ \Box \neg sw \ \wedge \ \Box(\ \Diamond(sw \wedge \Box\bot) \ \wedge \ [\mathsf{sw}](sw \rightarrow \Box\neg\Diamond sw)\ ) \\ (3) & \wedge & [\mathsf{sw}][\mathsf{sw}]\Box\Box\Box\Box(sw \rightarrow \Box\bot) \\ (4) & \wedge & \langle \mathsf{sw}\rangle[\mathsf{sw}](\ \neg s \wedge \neg sw \ \rightarrow (ⓡ\Diamond\Diamond(s \wedge \Diamond Ⓚ))'\ ) \\ (5) & \wedge & \Diamond(\varphi)'. \end{array}$$

Donde $(\ )'$ *se define como:*

$$
\begin{array}{lcl}
(\textcircled{r}\psi)' & = & (\ \Diamond sw \ \rightarrow\ \langle \mathsf{sw} \rangle (sw \wedge \Diamond(\psi)')\ ) \quad \wedge \quad (\ \neg\Diamond sw \ \rightarrow\ (\psi)'\ ) \\
\textcircled{k}' & = & \neg\Diamond sw \\
(\psi \otimes \chi)' & = & (\psi)' \otimes (\chi)' \quad \textit{para} \otimes \in \{\vee, \wedge\} \\
(\neg\psi)' & = & \neg(\psi)' \\
(\otimes\psi)' & = & \otimes(\neg s \wedge \psi)' \quad \textit{para} \otimes \in \{\Diamond, \langle \mathsf{sw} \rangle\} \\
(\otimes\psi)' & = & \otimes(\neg s \rightarrow \psi)' \quad \textit{para} \otimes \in \{\Box, [\mathsf{sw}]\}.
\end{array}
$$

En $\mathsf{Tr}(\varphi)$, el símbolo proposicional s es usado para identificar el punto de evaluación, que será el spy point, es decir, un punto que tiene un acceso directo a todos los otros punto en la componente conexa del modelo. sw representa los "switch points", los cuales serán usados para codificar los operadores de memoria.

(2) inicializa los interruptores, representada por aristas a los estados donde la fórmula (1) asegura de que el símbolo proposicional s vale en el punto de evaluación y no vale en cualquier punto accesible entre 1 y 4 pasos desde allí. (2) inicializa los switches, representados por aristas a los estados donde sw se satisface.

(3) asegura que los switches sean accesibles desde el punto de evaluación por un camino único. De hecho, si este no fuera el caso, entonces sería posible invertir dos aristas alcanzando algún switch, luego volver al punto de evaluación en dos pasos por este nuevo camino, y volver al mismo switch en dos pasos, donde la fórmula $(sw \wedge \neg\Box\bot)$ sería satisfecha.

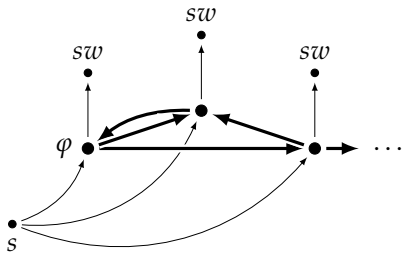
(4) asegura que el punto de evaluación está conectado a todo punto del modelo excepto con sí mismo y con los switch points. Notar que aunque $(\textcircled{k})'$ se satisfaga en el punto de evaluación, éste es irreflexivo por (1), luego $(s \wedge \Diamond\textcircled{k})'$ asegura que una arista conecte el punto de evaluación con el punto donde $\textcircled{r}$ se satisface.

(5) sitúa la traducción de la fórmula de lógica con memoria junto después del punto de evaluación.

Por definición de $(\textcircled{r}\psi)'$, la acción de recordar un punto en un modelo de φ se realiza en el modelo correspondiente de $\mathsf{Tr}(\varphi)$ girando la arista entre el correspondiente punto y su switch point. En el caso donde el punto ya ha sido memorizado, es decir, $(\textcircled{k})' = \neg\Diamond sw$ se satisface, entonces no es necesario hacer ninguna acción.

Es importante que los switches points no tengan sucesores y que ellos tengan exactamente un predecesor. Esto asegura que el camino tomado por $(\textcircled{r}\psi)'$ retorne correctamente al mismo punto del modelo.

Un modelo de $\mathsf{Tr}(\varphi)$ para algún φ es ilustrado a continuación:



Por ejemplo, un modelo de la fórmula $\mathsf{Tr}(\textcircled{r}\Diamond\textcircled{k})$ podría ser:

Teorema 3.3.2. *Sea φ una fórmula de $\mathcal{ML}(ⓡ, ⓚ)$ que no contiene los símbolos proposicionales s y sw. Entonces, φ y $\mathsf{Tr}(\varphi)$ son equisatisfactibles.*

Demostración. ($\Rightarrow$) Supongamos que φ es satisfactible, es decir, existe un modelo $\mathcal{M} = \langle W, R, V, \emptyset \rangle$ y $w \in W$ tal que $\langle W, R, V, \emptyset \rangle, w \models \varphi$.

Sea *sw* una función biyectiva entre W y un conjunto S tal que $S \cap W = \emptyset$, y *eval* un punto que no pertenece a $S \cup W$. Entonces podemos definir el modelo $\mathcal{M}' = \langle W', R', V' \rangle$ como:

$$\begin{array}{ll} W' & = W \cup \{s\} \cup S \\ R' & = R' \cup \{(eval, w) \mid w \in W\} \cup \{(w, sw(w)) \mid w \in W\} \\ V'(p) & = V(p) \quad \text{para } p \in \mathsf{PROP} \text{ que aparece en } \varphi \\ V'(s) & = \{s\} \\ V'(sw) & = \{sw(w) \mid w \in W\}. \end{array}$$

Siguiendo la descripción de $\mathsf{Tr}(\varphi)$ dada en esta sección, podemos verificar que $\mathcal{M}', s \models \mathsf{Tr}(\varphi)$.

($\Leftarrow$) Para la otra dirección, supongamos $\mathsf{Tr}(\varphi)$ es satisfactible, es decir, existe un modelo $\mathcal{M} = \langle W, R, V \rangle$, y $w \in W$ tal que $\langle W, R, V \rangle, w \models \mathsf{Tr}(\varphi)$. Entonces podemos definir el modelo $\mathcal{M}' = \langle W', R', V', \emptyset \rangle$ donde

$$\begin{array}{ll} W' & = \{\ v \mid (w, v) \in R\ \} \\ R' & = R \cap (W' \times W') \\ V'(p) & = V(p) \cap W' \quad \text{para } p \in \mathsf{PROP}. \end{array}$$

Podemos verificar fácilmente que existe $w \in W'$ tal que $\mathcal{M}', w \models \varphi$. □

Entonces obtenemos inmediatamente:

Teorema 3.3.3. *El problema de decidir satisfactibilidad para $\mathcal{ML}(\langle \mathsf{sw} \rangle)$ es indecidible.*

Como hicimos para los otros operadores, conjeturamos que ideas similares pueden ser usadas para demostrar indecidibilidad para la versión global del operador de swap.

La técnica que utilizamos para probar que el problema satisfactibilidad de las lógicas modales con operadores de cambio de modelo es indecidible, es una reducción del problema de decidir satisfactibilidad para la lógica con memoria $\mathcal{ML}(ⓡ, ⓚ)$. La clave es el hecho de que podemos simular la memorización de un elemento, y la consulta de pertenencia a la memoria mediante la definición de un modelo con spy points y utilizando los operadores de cambio de accesibilidad. El poder expresivo obtenido por tener comportamiento dinámico en el lenguaje hace que sea posible forzar estructuras complejas tales como modelos infinitos o modelos con spy points.

Esta es la razón por la que podemos conjeturar que podemos utilizar las mismas técnicas usadas en este capítulo para demostrar indecidibilidad de los operadores globales.

4

Model Checking

The biggest difference between time and space is that you can't reuse time.

Merrick Furst.

4.1 El problema de model checking

En esta sección vamos a establecer cotas computacionales para el problema de model checking de las lógicas que hemos introducido. Para ello vamos a reducir el problema de satisfactibilidad de Quantified Boolean Formulas (QBF) [Papadimitriou, 1994] al problema de model checking de cada lógica. PSpace-hardness para la versión global de sabotage fue demostrado en [Löding and Rohde, 2003b; Löding and Rohde, 2003a], pero vamos a introducir una demostración más directa. Para $\langle \mathsf{sw} \rangle$, ya ha sido demostrado este resultado en [Areces *et al.*, 2013b], y para $\langle \mathsf{sb} \rangle$ y $\langle \mathsf{br} \rangle$, en [Areces *et al.*, 2012].

Primero vamos a introducir QBF, un problema clásico de complejidad PSpace-completo. Introducimos su sintaxis y semántica.

Definición 4.1.1 (Sintaxis de QBF). *Sea* PROP *un conjunto infinito contable de símbolos de proposición. El conjunto de fórmulas de QBF* FORM *sobre* PROP *se define como:*

$$\mathsf{FORM} ::= x \mid \neg\varphi \mid \varphi \wedge \psi \mid \exists x\varphi,$$

donde $x \in \mathsf{PROP}$ *y* $\varphi, \psi \in \mathsf{FORM}$.

Satisfactibilidad depende solo de asignaciones de verdad a variables proposicionales, por lo tanto los modelos de QBF son simplemente funciones de valuación.

Definición 4.1.2 (Semántica de QBF). *Sea* $v : \{x_1, \ldots, x_k\} \to \{0, 1\}$ *una valuación, la relación* $\models_{\text{qbf}}$ *se define como:*

$$\begin{array}{lll} v \models_{\text{qbf}} x_i & \textit{sii} & v(x_i) = 1 \\ v \models_{\text{qbf}} \neg\varphi & \textit{sii} & v \not\models_{\text{qbf}} \varphi \\ v \models_{\text{qbf}} \varphi \wedge \psi & \textit{sii} & v \models_{\text{qbf}} \varphi \textit{ y } v \models_{\text{qbf}} \psi \\ v \models_{\text{qbf}} \exists x_i\varphi & \textit{sii} & v[x_i \mapsto 0] \models_{\text{qbf}} \varphi \textit{ or } v[x_i \mapsto 1] \models_{\text{qbf}} \varphi \end{array}$$

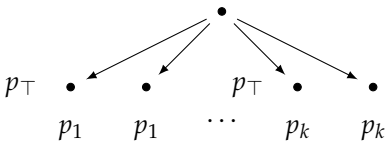
La valuación $v[x_i \mapsto n]$ *se define como* $v[x_i \mapsto n](x_i) = n$ *y* $v[x_i \mapsto n](x) = v(x)$, *para* $x \neq x_i$. φ *es satisfactible si existe una valuación* v *tal que* $v \models_{\text{qbf}} \varphi$.

Satisfactibilidad para QBF es PSpace-completo [Papadimitriou, 1994]. Vamos a usar este resultado para demostrar que model checking para las lógicas con operadores que cambian la accesibilidad es PSpace-hard.

Teorema 4.1.3. *Sea* $\blacklozenge \in \{\langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle\}$, *model checking para todas las lógicas* $\mathcal{ML}(\blacklozenge)$ *es* PSPACE*-hard.*

Demostración. Vamos a reducir el problema de satisfactibilidad para QBF (que es PSPACE-completo) al problema de model checking de cada una de las seis lógicas. Mostramos la demostración completa para el caso de $\langle \mathsf{sw} \rangle$; para los demás operadores se utilizan estrategias similares.

- Consideremos $\mathcal{ML}(\langle \mathsf{sw} \rangle)$. Sea α una fórmula de QBF formula con variables $\{x_1, \ldots, x_k\}$. Sin pérdida de generalidad, podemos asumir que α no tiene variables libres y que ninguna variable está cuantificada dos veces. Podemos construir en tiempo polinomial la estructura relacional $\mathcal{M}_k$ introducida a continuación. El punto de evaluación tiene dos sucesores por cada variable en α, pero solo uno de cada tipo está etiquetado con $p_\top$.



  $\mathcal{M}_k = \langle W, R, V \rangle$ es construido sobre una signatura con un símbolo relacional y proposiciones $\{p_\top, p_1, \ldots, p_k\}$, donde:

$$\begin{array}{ll} W & = \{w\} \cup \{w_i^1, w_i^0 \mid 1 \leq i \leq k\} \\ V(p_i) & = \{w_i^1, w_i^0\} \\ V(p_\top) & = \{w_i^1 \mid 1 \leq i \leq k\} \\ R & = \{(w, w_i^1), (w, w_i^0) \mid 1 \leq i \leq k\}. \end{array}$$

  Ahora necesitamos traducir QBF a fórmulas de $\mathcal{ML}(\langle \mathsf{sw} \rangle)$. La idea es que cada vez que tenemos una fórmula con un existencial sobre una variable x_i, invertimos la dirección de una de las aristas apuntando a estados p_i en $\mathcal{M}_k$. Continuamos la evaluación del resto de la fórmula en dicho estado p_i en el nuevo model variant. Si invertimos la arista que apunta a $p_\top$, representa el hecho de que a x_i le asignamos el valor de verdad 1. En caso contrario, se asigna 0. Los otros operadores se traducen de la manera obvia.

  Sea $(\ )'$ la siguiente traducción lineal de QBF en $\mathcal{ML}(\langle \mathsf{sw} \rangle)$:

$$\begin{array}{ll} (\exists x_i.\alpha)' & = \langle \mathsf{sw} \rangle (p_i \wedge \Diamond(\alpha)') \\ (x_i)' & = \neg\Diamond(p_i \wedge p_\top) \\ (\neg\alpha)' & = \neg(\alpha)' \\ (\alpha \wedge \beta)' & = (\alpha)' \wedge (\beta)'. \end{array}$$

  Resta demostrar que α es satisfactible sii $\mathcal{M}_k, w \models (\alpha)'$. Para un modelo $\mathcal{M}$ con una relación R definimos $v_R : \{x_1, \ldots, x_k\}$ como "$v_R(x_i) = 1$ iff $(w, w_i^1) \notin R$", en el presente caso, sii la arista entre w y w_i^1 ha sido invertida.

  Sea β una subfórmula de α. Vamos a demostrar por inducción en β que $\mathcal{M}, w \models (\beta)'$ sii $v_R \models_{\text{qbf}} \beta$. La primera observación es que R satisface i) si x_i

aparece libre en β, entonces $(w, w_i^1) \notin R$ o $(w, w_i^0) \notin R$ pero no ambas, y ii) si x_i no aparece libre en β entonces $(w, w_i^1) \in R$ y $(w, w_i^0) \in R$. Luego $\mathcal{M}_k, w \models (\alpha)'$ sii $v \models_{\text{qbf}} \alpha$ para toda v dado que α no tiene variables libres, sii α es satisfactible.

Para el caso base, $v_R \models_{\text{qbf}} x_i$ sii $(w, w_i^1) \notin R$ lo que implica (por definición de $\mathcal{M}_k$) $\mathcal{M}, w \models (x_i)'$. Para la otra dirección supongamos $\mathcal{M}, w \not\models (x_i)'$. Luego $\mathcal{M}, w \models \Diamond(p_i \wedge p_\top)$ lo que implica $(w, w_i^1) \in R$ y $v_R \not\models_{\text{qbf}} x_i$.

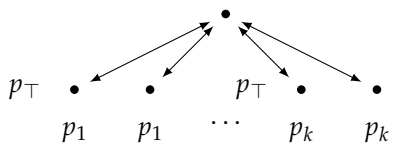
Los casos Booleanos son directos de la hipótesis inductiva.

Consideremos el caso $\beta = \exists x_i.\gamma$. Dado que no hay variables ligadas dos veces en α sabemos que $(w, w_i^1) \in R$ y $(w, w_i^0) \in R$. Tenemos que $v_R \models_{\text{qbf}} \beta$ sii $(v_R[x_i \mapsto 0] \models_{\text{qbf}} \gamma$ o $v_R[x_i \mapsto 1] \models_{\text{qbf}} \gamma)$ sii $(v_{R^*_{w_i^0 w}} \models_{\text{qbf}} \gamma$ o $v_{R^*_{w_i^1 w}} \models_{\text{qbf}} \gamma)$.

Por H.I., esto vale si y solo si $(\mathcal{M}^*_{w_i^0 w}, w_i^0 \models \Diamond(\gamma)'$ o $\mathcal{M}^*_{w_i^1 w}, w_i^1 \models \Diamond(\gamma)')$ sii $\mathcal{M}, w \models \langle \mathsf{sw} \rangle (p_i \wedge \Diamond(\gamma)')$ sii $\mathcal{M}, w \models (\exists x_i.\gamma)'$.

Con esto mostramos que model checking para $\mathcal{ML}(\langle \mathsf{sw} \rangle)$ es PSPACE-hard.

- Para $\mathcal{ML}(\langle \mathsf{sb} \rangle)$ y $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$, usamos el siguiente modelo $\mathcal{M}_k$:



$\mathcal{M}_k = \langle W, R, V \rangle$ es idéntico a la construcción para swap, excepto que:

$$\begin{array}{ll} W & = \{w\} \cup \{w_i^1, w_i^0 \mid 1 \leq i \leq k\} \\ V(p_i) & = \{w_i^1, w_i^0\} \\ V(p_\top) & = \{w_i^1 \mid 1 \leq i \leq k\} \\ R & = \{(w, w_i^1), (w, w_i^0), \\ & \qquad (w_i^1, w), (w_i^0, w) \mid 1 \leq i \leq k\} \end{array}$$

Sea $(\ )'$ la siguiente traducción de QBF en $\mathcal{ML}(\langle \mathsf{sb} \rangle)$:

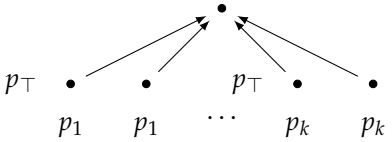
$$\begin{array}{ll} (\exists x_i.\alpha)' & = \langle \mathsf{sb} \rangle (p_i \wedge \Diamond(\alpha)') \\ (x_i)' & = \neg\Diamond(p_i \wedge p_\top) \\ (\neg\alpha)' & = \neg(\alpha)' \\ (\alpha \wedge \beta)' & = (\alpha)' \wedge (\beta)'. \end{array}$$

De QBF a $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$, introducimos la siguiente traducción:

$$\begin{array}{ll} (\exists x_i.\alpha)' & = \langle \mathsf{gsb} \rangle ((\neg\Diamond(p_i \wedge p_\top) \vee \neg\Diamond(p_i \wedge \neg p_\top)) \wedge \Diamond(p_i \wedge \Diamond(\alpha)')) \\ (x_i)' & = \neg\Diamond(p_i \wedge p_\top) \\ (\neg\alpha)' & = \neg(\alpha)' \\ (\alpha \wedge \beta)' & = (\alpha)' \wedge (\beta)'. \end{array}$$

Una fórmula QBF α es satisfactible sii $\mathcal{M}_k, w \models (\alpha)'$, en ambos casos.

- Para mostrar PSPACE-hardness para $\mathcal{ML}(\langle \mathsf{br} \rangle)$ y $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$, construimos el siguiente $\mathcal{M}_k$:



$\mathcal{M}_k = \langle W, R, V \rangle$ se define como:

$$\begin{array}{ll} W & = \{w\} \cup \{w_i^1, w_i^0 \mid 1 \leq i \leq k\} \\ V(p_i) & = \{w_i^1, w_i^0\} \\ V(p_\top) & = \{w_i^1 \mid 1 \leq i \leq k\} \\ R & = \{(w_i^1, w), (w_i^0, w) \mid 1 \leq i \leq k\} \end{array}$$

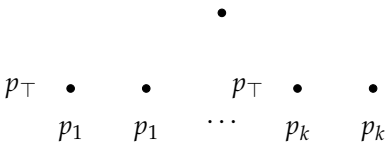
Para $\mathcal{ML}(\langle \mathsf{br} \rangle)$ usamos la siguiente traducción lineal:

$$\begin{array}{l} (\exists x_i.\alpha)' = \langle \mathsf{br} \rangle (p_i \wedge \Diamond(\alpha)') \\ (x_i)' \quad = \Diamond(p_i \wedge p_\top) \\ (\neg\alpha)' \quad = \neg(\alpha)' \\ (\alpha \wedge \beta)' = (\alpha)' \wedge (\beta)'. \end{array}$$

Para $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$:

$$\begin{array}{l} (\exists x_i.\alpha)' = \langle gsw \rangle (\Diamond p_i \ \wedge \ (\alpha)') \\ (x_i)' \quad = \Diamond(p_i \wedge p_\top) \\ (\neg\alpha)' \quad = \neg(\alpha)' \\ (\alpha \wedge \beta)' = (\alpha)' \wedge (\beta)'. \end{array}$$

- Finalmente, para demostrar PSPACE-hardness para$\mathcal{ML}(\langle \mathsf{gbr} \rangle)$, construimos el siguiente modelo $\mathcal{M}_k$:



$\mathcal{M}_k = \langle W, R, V \rangle$ se define como:

$$\begin{array}{ll} W & = \{w\} \cup \{w_i^1, w_i^0 \mid 1 \leq i \leq k\} \\ V(p_i) & = \{w_i^1, w_i^0\} \\ V(p_\top) & = \{w_i^1 \mid 1 \leq i \leq k\} \\ R & = \emptyset \end{array}$$

La traducción $(\ )'$ de QBF a $\mathcal{ML}(\langle \mathsf{gbr} \rangle)$:

$$\begin{array}{l} (\exists x_i.\alpha)' = \langle gbr \rangle (\Diamond p_i \ \wedge \ (\alpha)') \\ (x_i)' \quad = \Diamond(p_i \wedge p_\top) \\ (\neg\alpha)' \quad = \neg(\alpha)' \\ (\alpha \wedge \beta)' = (\alpha)' \wedge (\beta)'. \end{array}$$

Hemos reducido satisfactibilidad para QBF a model checking para las lógicas introducidas en Definición 1.3.1. Entonces, model checking para todas ellas es PSPACE-hard. □

Para demostrar PSPACE-completitud de las seis lógicas, tenemos que mostrar que verificar si una fórmula se satisface en un modelo puede hacerse usando espacio polinomial.

Teorema 4.1.4. *Model checking for* $\mathcal{ML}(\langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle)$ *pertenece a la clase* PSPACE.

Demostración. La evaluación de una fórmula en un modelo puede hacerse en un algoritmo que use espacio polinomial, siguiendo la Definición 1.3.3.

□

Entonces obtenemos:

Teorema 4.1.5. *Para todo* $\blacklozenge \in \{\langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle, \langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle, \langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle\}$, *model checking para* $\mathcal{ML}(\blacklozenge)$ *es* PSPACE*-completo.*

En la siguiente sección establecemos algunos problemas tratables para las seis lógicas que introdujimos. Vamos a estudiar model checking con una fórmula fija, y con un modelo fijo.

4.2 COMPLEJIDAD DE FÓRMULA Y DE PROGRAMA

Además de considerar model checking en función del modelo y la fórmula, podemos considerarlo fijando el modelo (complejidad de *fórmula*) o fijando la fórmula (complejidad de *programa* o de *dato*). Para $\mathcal{FOL}$ se demostró en [Vardi, 1982] que ambos problemas son PSPACE-completos. Para LTL y CTL*, la complejidad de model checking y de fórmula son PSPACE-completos, pero la complejidad de programa es NLOGSPACE-completo (ver detalles en [Vardi and Wolper, 1986; Kupferman *et al.*, 2000; Schnoebelen, 2002].)

Ha sido demostrado en [Löding and Rohde, 2003a; Rohde, 2006] que la complejidad de fórmula y de programa de $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$ son lineal y polinomial, respectivamente. A continuación vamos a generalizar esos resultados para $\mathcal{ML}(\langle \mathsf{sb} \rangle)$, $\mathcal{ML}(\langle \mathsf{sw} \rangle)$, $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$, $\mathcal{ML}(\langle \mathsf{br} \rangle)$ y $\mathcal{ML}(\langle \mathsf{gbr} \rangle)$. En particular, lo vamos a hacer para una familia más amplia de operadores dinámicos.

Sea W un conjunto de estados de un modelo y sea f una función que toma un elemento w de W y la relación de accesibilidad actual R sobre W, y retorna un conjunto de pares (v, S), donde $v \in W$ es el nuevo punto de evaluación y S es la nueva relación de acecesibilidad (aquí solo discutimos relaciones binarias, luego, $f : W \times 2^{W^2} \mapsto 2^{W \times 2^{W^2}}$ pero podría generalizarse para modalidades de aridad arbitraria). Cada función f diferente define un nuevo operador. Por ejemplo, $\langle \mathsf{sw} \rangle$ estaría definido por una función f tal que $f(w, R) = \{(v, R \backslash \{(w, v)\} \cup \{(v, w)\}) \mid (w, v) \in R\}$.

Esta definición no cubre todos los posibles operadores dinámicos, por ejemplo algunos definidos en [van Ditmarsch *et al.*, 2007], pero cubre un amplio rango de operadores, como los definidos en [Rohde, 2006; Areces *et al.*, 2012; Areces *et al.*, 2013b;

Areces *et al.*, 2013c] (ver [Areces and Gorín, 2010] para una presentación mas general del framework).

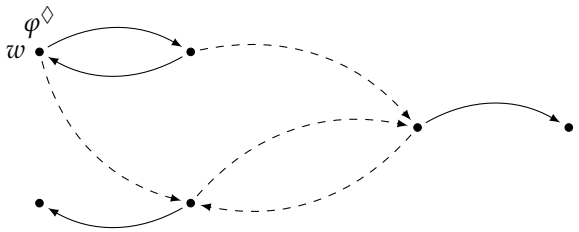
Definición 4.2.1 (Funciones de Actualización de Modelos). *Sea $\mathcal{C}$ una clase de modelos. Decimos que $F = \{f_W : W \times 2^{W^2} \mapsto 2^{W \times 2^{W^2}} \mid \mathcal{M} = \langle W, R, V\rangle \in \mathcal{C}\}$ es una familia de* funciones de actualización de modelos. *Decimos que $\mathcal{C}$ es cerrada bajo una familia de funciones de actualización F si se cumple que $\mathcal{M} = \langle W, R, V\rangle \in \mathcal{C}$, entonces $\{\langle W, R', V\rangle \mid f_W \in F, w \in W, (v, R') \in f_W(w, R)\} \subseteq \mathcal{C}$.*

Ahora podemos usar familias de funciones de actualización para definir los operadores introducimos en el Capítulo 1:

$$
\begin{array}{lll}
F_{sw} & = \{f_W^{sw}\}, \text{ donde para todo } \mathcal{M} = \langle W, R, V\rangle, f_W^{sw}(w, R) & = \{(v, R^*_{vw}) \mid wv \in R\}. \\
F_{gsw} & = \{f_W^{gsw}\}, \text{ donde para todo } \mathcal{M} = \langle W, R, V\rangle, f_W^{gsw}(w, R) & = \{(w, R^*_{vu}) \mid uv \in R\}. \\
F_{sb} & = \{f_W^{sb}\}, \text{ donde para todo } \mathcal{M} = \langle W, R, V\rangle, f_W^{sb}(w, R) & = \{(v, R^-_{wv}) \mid wv \in R\}. \\
F_{gsb} & = \{f_W^{gsb}\}, \text{ donde para todo } \mathcal{M} = \langle W, R, V\rangle, f_W^{gsb}(w, R) & = \{(w, R^-_{uv}) \mid uv \in R\}. \\
F_{br} & = \{f_W^{br}\}, \text{ donde para todo } \mathcal{M} = \langle W, R, V\rangle, f_W^{br}(w, R) & = \{(v, R^+_{wv}) \mid wv \notin R\}. \\
F_{gbr} & = \{f_W^{gbr}\}, \text{ donde para todo } \mathcal{M} = \langle W, R, V\rangle, f_W^{gbr}(w, R) & = \{(w, R^+_{uv}) \mid uv \notin R\}.
\end{array}
$$

Para investigar complejidad de fórmula y de programa, vamos a usar una técnica de unfolding de modelos. Partimos de un modelo $\mathcal{M}$ y una fórmula φ de una lógica $\mathcal{ML}(\blacklozenge)$:

Empezando de $\mathcal{M}$, construimos explícitamente los model variants obtenidos aplicando sucesivamente el operador $\blacklozenge$.



En este nuevo modelo, usamos dos relaciones: una representa la relación interna correspondiente a cada model variant, y una relación exterrna que relaciona model variants entre sí. A nivel sintáctico, vamos a reescribir φ en una fórmula $\varphi^{\Diamond}$ de la lógica modal básica con dos modalidades. Siguiendo esta idea reducimos el problema de model checking de un lenguaje dinámico a model checking de la lógica modal básica. Este problema es polinomial con respecto al tamaño de la fórmula y del modelo, más concretamente se puede resolver en tiempo $\mathcal{O}(|\varphi| \cdot |\mathcal{M}|)$, donde $|\varphi|$ es el tamaño de la fórmula $\mathcal{ML}$ dada φ y $|\mathcal{M}|$ es el tamaño del modelo dado $\mathcal{M}$.

Definimos la siguiente traducción:

Definición 4.2.2. *Consideremos el lenguaje $\mathcal{ML}(\blacklozenge_F)$ para alguna familia F de funciones de actualización. Definimos la siguiente traducción desde este lenguaje al lenguaje modal básico con dos modalidades:*

$$\begin{array}{rcl}(p)^{\Diamond} & = & p \\ (\varphi \wedge \psi)^{\Diamond} & = & \varphi^{\Diamond} \wedge \psi^{\Diamond} \\ (\neg\varphi)^{\Diamond} & = & \neg\varphi^{\Diamond} \\ (\Diamond\varphi)^{\Diamond} & = & \Diamond_1\varphi^{\Diamond} \\ (\blacklozenge_F\varphi)^{\Diamond} & = & \Diamond_2\varphi^{\Diamond}.\end{array}$$

Antes de completar la construcción del unfolding, necesitamos una medida de complejidad para las lógicas con operadores de cambio de accesibilidad.

Definición 4.2.3. *El número máximo de modalidades dinámicas anidadas es llamada* dynamic depth*, y se define como:*

$$\begin{array}{rcl}dd(p) & = & 0 \\ dd(\neg\varphi) & = & dd(\varphi) \\ dd(\varphi \wedge \varphi') & = & max(dd(\varphi), dd(\varphi')) \\ dd(\Diamond\varphi) & = & dd(\varphi) \\ dd(\blacklozenge_F\varphi) & = & 1 + dd(\varphi),\end{array}$$

con $\blacklozenge_F$ el operador definido por la familia de funciones de actualización F.

Medir las fórmulas de acuerdo con el número de modalidades de cambio de accesibilidad anidadas ayudará a establecer algunos límites en el tamaño de los modelos "unfolded". La idea general es que la profundidad dinámica será el número de model variants que pondremos en el unfolding. Estos modelos se construyeron utilizando la siguiente definición de todas las posibles relaciones de accesibilidad a partir de una función de actualización:

Definición 4.2.4. *Sea W un dominio, $R \subseteq W^2$ una relación de accesibilidad, f una función de actualización para W, y n un número natural. Definimos inductivamente $Vars(R, f, n)$, el conjunto de todas las posibles relaciones obtenidas aplicando n veces la función f a la relación inicial R como:*

$$\begin{array}{rcl}Vars(R, f, 0) & = & \{R\} \\ Vars(R, f, n+1) & = & \{T \mid (v, T) \in f(w, S), S \in Vars(R, f, n), w \in W\}.\end{array}$$

Sea

$$Vars(R, f) = \bigcup_{n<\omega} Vars(R, f, n)$$

el conjunto de todas las relaciones obtenidas aplicando f a la relación inicial R.

Introducimos ahora la traducción de modelos de las lógicas con cambios de accesibilidad en modelos estáticos basados en funciones de actualización. Luego vamos a usarlos para realizar model checking con fórmulas de la Definición 4.2.2.

Definición 4.2.5. *Sea $\mathcal{M} = \langle W, R, V\rangle$. Sea $f : W \times 2^{W^2} \mapsto 2^{(W \times 2^{W^2})}$.*

Definimos $\mathcal{M}_{f,0} = \langle W', \{R'_1, R'_2\}, V' \rangle$ *como:*

$$\begin{array}{lcl} W' & = & W \times \{R\} \\ R'_1 & = & \{((s,R),(t,R)) \mid (s,t) \in R\} \\ R'_2 & = & \emptyset \\ V'(p) & = & \{(s,S) \mid s \in V(p), S \in Vars(R,f)\}. \end{array}$$

Definimos ahora $\mathcal{M}_{f,n} = \langle W', \{R'_1, R'_2\}, V' \rangle$ *(para* $n \geq 1$*) como:*

$$\begin{array}{lcl} W' & = & W \times Vars(R,f,n) \\ R'_1 & = & \bigcup\{((s,S),(t,S)) \mid (s,t) \in S, S \in Vars(R,f)\} \\ R'_2 & = & \{((s,S),(t,T)) \mid (t,T) \in f(s,S), S \in Vars(R,f,n-1)\} \\ V'(p) & = & \{(s,S) \mid s \in V(p), S \in Vars(R,f)\}. \end{array}$$

Finalmente, definimos $\mathcal{M}_f = \langle W', \{R'_1, R'_2\}, V' \rangle$ *como:*

$$\begin{array}{lcl} W' & = & W \times Vars(R,f) \\ R'_1 & = & \{((s,S),(t,S)) \mid (s,t) \in S, S \in Vars(R,f)\} \\ R'_2 & = & \{((s,S),(t,T)) \mid (t,T) \in f(s,S), S \in Vars(R,f)\} \\ V'(p) & = & \{(s,S) \mid s \in V(p), S \in Vars(R,f)\}. \end{array}$$

Es fácil verificar lo siguiente, usando las definiciones anteriores:

Lema 4.2.6. *Sea* $\mathcal{M} = \langle W, R, V \rangle$*, y* φ *una fórmula. Entonces*

$$\mathcal{M}, w \models \varphi \text{ iff } \mathcal{M}_{f,dd(\varphi)}, (w,R) \models \varphi^{\Diamond}.$$

Ahora el resultado complejidad. En primer lugar, observamos que si un operador de cambio de accesibilidad sólo puede crear un número polinómico de relaciones posibles en un solo paso, entonces el número de posibles relaciones después de n pasos también es polinomial:

Lema 4.2.7. *Sea* $f : W \times 2^{W^2} \mapsto 2^{(W \times 2^{W^2})}$*. Supongamos que para todo* $s \in W$*,* $S \subseteq W \times W$*, existe* $c > 0$ *tal que* $|f(s,S)| \in \mathcal{O}(|W|^c)$*. Entonces existe* $d > 0$ *tal que* $|Vars(R,f,n)| \in \mathcal{O}(|W|^{dn})$*.*

En particular, dado un modelo $\mathcal{M}$*, es posible construir* $\mathcal{M}_{f,n}$ *en tiempo polinomial en términos de* n*, donde* n *es el número de aplicaciones de* f *en* $\mathcal{M}$*.*

Esto nos lleva al resultado general de complejidad de fórmula y de programa, para lógicas modales equipadas con operadores "polinomiales" de cambio de accesibilidad:

Teorema 4.2.8. *Sea* $\mathcal{L}$ *la lógica modal básica equipada con un operador de cambio cuya semántica está definida por una familia* F *de funciones de actualización tal que para todo* $\mathcal{M} = \langle W, R, V \rangle$*, todo* $s \in W$*, y todo* $S \subseteq W \times W$*,* $|f(s,S)| \in \mathcal{O}(|W|^c)$*.*

1. *Model checking para* $\mathcal{L}$ *con un modelo finito fijo se puede resolver en tiempo lineal con respecto al tamaño de la fórmula (complejidad de fórmula).*
2. *Model checking para* $\mathcal{L}$ *con una fórmula fija se puede resolver en tiempo polinomial con respecto al tamaño del modelo finito (complejidad de programa).*

Demostración. Ambas partes siguen de la complejidad de model checking para $\mathcal{ML}$. □

Observar que las modalidades $\langle \mathsf{sw} \rangle$, $\langle \mathsf{gsw} \rangle$, $\langle \mathsf{sb} \rangle$, $\langle \mathsf{gsb} \rangle$, $\langle \mathsf{br} \rangle$ y $\langle \mathsf{gbr} \rangle$ satisfacen las condiciones del Lema 4.2.7, luego el Teorema 4.2.8 se aplica a cada lógica correspondiente. Más aún: el resultado se extiende para la lógica modal básica equipada con cualquier combinación de este tipo de operadores.

Un simple ejemplo de un operador que no satisface las condiciones del teorema puede ser la modalidad "universal-universal", que puede explotar el número de posibles relaciones a 2^{W^2} en solo un paso:

$$f_W : (s, S) \mapsto \{(t, T) \mid T \subseteq W \times W, t \in W\}.$$

Evaluar esta modalidad "universal-universal" consiste en considerar todas los modelos con dominio W, alguna valuación fija V y todas las posibles relaciones binarias sobre W.

5

Tableaux

In solving a problem of this sort, the grand thing is to be able to reason backwards. That is a very useful accomplishment, and a very easy one, but people do not practise it much. In the every-day affairs of life it is more useful to reason forwards, and so the other comes to be neglected. There are fifty who can reason synthetically for one who can reason analytically... Let me see si I can make it clearer. Most people, if you describe a train of events to them, will tell you what the result would be. They can put those events together in their minds, and argue from them that something will come to pass. There are few people, however, who, si you told them a result, would be able to evolve from their own inner consciousness what the steps were which led up to that result. This power is what I mean when I talk of reasoning backwards, or analytically.

-Sherlock Holmes.

from "A Study in Scarlet" , Sir Arthur Conan Doyle.

5.1 CÁLCULO DE TABLEAU

Ya investigamos tareas de razonamiento desde un punto de vista teórico, ahora es el momento de implementar algún procedimiento. Un algoritmo de tableau es un procedimiento que decide si una fórmula es satisfactible o no explorando la satisfactibilidad de las subfórmulas. En caso de ser satisfactible, el algoritmo también retorna el modelo. Tableaux para lógica de primer orden fue introducida en [Beth, 1955] y estudiada más tarde también en [Smullyan, 1968]. Los primeros resultados acerca de tableaux para lógicas modales fueron introducidos en [Fitting, 1972]. Para más información consultar [D'Agostino *et al.*, 1999].

Vamos a presentar definiciones básicas para diferentes algoritmos de tableau para operadores que cambian la accesibilidad del modelo, y nos vamos a basar en los resultados introducidos en [Areces *et al.*, 2013c].

Definición 5.1.1 (Fórmulas de tableau)**.** *Sea* NOM *un conjunto infinito y bien ordenado de símbolos llamados* nominales*. Un* fórmula de tableau *es una* fórmula con prefijo*, una* fórmula ecuacional *o una* fórmula relacional*. Una fórmula con prefijo es de la forma* $(n, X) : \varphi$*, con* $n \in \mathsf{NOM}$*,* $X \subseteq \mathsf{NOM}^2$*, y* φ *una fórmula del lenguaje objeto. Una fórmula ecuacional es una combinación Booleana de fórmulas de la forma* $n \doteq m$ *or* $n \dot{\neq} m$ *con* $n, m \in \mathsf{NOM}$*. Usamos la siguiente notación:*

$$
\begin{array}{lcllcl}
nm \doteq xy & := & n \doteq x \wedge m \doteq y & nm \dot{\in} X & := & \bigvee\limits_{xy \in X} nm \doteq xy \\
nm \dot{\neq} xy & := & n \dot{\neq} x \vee m \dot{\neq} y & nm \dot{\notin} X & := & \bigwedge\limits_{xy \in X} nm \dot{\neq} xy.
\end{array}
$$

En particular $nm\dot{\in}\emptyset$ es una notación para $\bot$ y $nm\dot{\notin}\emptyset$ es una notación para $\top$. Una fórmula relacional es de la forma $\dot{R}nm$ o $\neg\dot{R}nm$, con $n, m \in \mathsf{NOM}$.

El conjunto X de una fórmula con prefijo $(n, X) : \varphi$ es usado para describir el model variant en el cual la fórmula φ va a ser interpretada. De acuerdo con la lógica con la cual estamos trabajando, este conjunto será interpretado de una manera diferente. Esto se hace fijando una función f que toma dos relaciones $R, S \subseteq W \times W$ y retorna otra relación $R' = f(R, S)$.

Definición 5.1.2 (Ramas e interpretaciones)**.** *Una* rama *es un conjunto no vacío de fórmulas de tableau. Sea $\mathcal{M} = \langle W, R, V\rangle$ un modelo, $f : W^2 \times W^2 \mapsto W^2$ una función de cambio de relación, y $\sigma : \mathsf{NOM} \mapsto W$ un mapeo de nominales en estados de $\mathcal{M}$. Definimos $X^\sigma = \{\sigma(a)\sigma(b) \mid ab \in X\}$, para $X \subseteq \mathsf{NOM}^2$.*

Sea $\mathcal{M} = \langle W, R, V\rangle$, definamos $\mathcal{M}^f_{X^\sigma} = \langle W, f(R, X^\sigma), V\rangle$. Es decir, $\mathcal{M}^f_{X^\sigma}$ es el modelo $\mathcal{M}$ actualizado por la función de cambio de relación f de acuerdo al conjunto de pares de nominales X con el mapeo σ.

Una rama Θ es satisfactible *si existe un modelo $\mathcal{M} = \langle W, R, V\rangle$ y un mapeo σ tal que todas las fórmula de Θ son satisfactibles en el modelo $\mathcal{M}$ y el mapeo σ. Es decir, deben satisfacer las siguientes condiciones:*

- *si $(n, X) : \varphi \in \Theta$ entonces $\mathcal{M}^f_{X^\sigma}, \sigma(n) \models \varphi$,*
- *si $n\dot{=}m \in \Theta$ entonces $\sigma(n) = \sigma(m)$,*
- *si $n\dot{\neq}m \in \Theta$ entonces $\sigma(n) \neq \sigma(m)$,*
- *combinaciones Booleanas de fórmulas ecuacionales son interpretadas de la manera esperada,*
- *si $\dot{R}nm \in \Theta$ entonces $R\sigma(n)\sigma(m)$,*
- *si $\neg\dot{R}nm \in \Theta$ entonces $\neg R\sigma(n)\sigma(m)$.*

Una rama es insatisfactible *si no es satisfactible.*

Ahora vamos a introducir el cálculo formalmente.

Definición 5.1.3 (Tableau)**.** *Un* cálculo de tableau *es un conjunto de reglas tal que cada regla se aplica a una rama y produce una o más ramas, bajo ciertas condiciones, llamadas condiciones de saturación. Se estipula que ninguna regla puede ser aplicada dos veces en las mismas premisas, y que ninguna fórmula puede ser introducida dos veces en la misma rama.*

Un tableau *es un árbol en el cual cada nodo define una rama de tableau, y las aristas son aplicaciones de las reglas. Un tableau es completamente expandido tanto como es posible por las reglas del sistema. Un tableau completamente expandido es llamado* saturado*.*

Una rama es cerrada *si contiene a $\bot$, caso contrario es* abierta*. Un tableau es cerrado si todas sus ramas están cerradas, sino es abierto.*

Sea Θ una rama, $\sim_\Theta$ denota la clausura por equivalencia de la relación $\{nm \mid n\dot{=}m \in \Theta\}$, y escribimos $\bar{n}$ para el nominal más pequeño x tal que $x \sim_\Theta n$. Para $X \subseteq \mathsf{NOM}^2$ escribimos $\bar{X} = \{\bar{n}\bar{m} \mid nm \in X\}$. La Figura 10 presenta las reglas comunes a todos los cálculos de tableau presentados en este capítulo. Nos referimos a las

reglas Booleanas $(\wedge)$ y $(\vee)$, las reglas de clash $(\bot_{atom})$ y $(\bot_{\neq})$, las reglas ecuacionales $(R\sim)$ y (Id), y la regla de unrestricted blocking (ub) [Schmidt and Tishkovsky, 2007]. Usamos la regla de unrestricted blocking como una manera de saturar ramas con fórmulas ecuacionales. Estas fórmulas pueden aparecer como premisas en las reglas que vamos a introducir más adelante.

$$\frac{(n, X) : \varphi \wedge \psi}{\begin{array}{l}(n, X) : \varphi \\ (n, X) : \psi\end{array}} (\wedge) \qquad \frac{(n, X) : \varphi \vee \psi}{(n, X) : \varphi \mid (n, X) : \psi} (\vee)$$

$$\frac{\begin{array}{l}(n, X_1) : p \\ (n, X_2) : \neg p\end{array}}{\bot} (\bot_{atom})^1 \qquad \frac{\begin{array}{c}n \sim_\Theta m \\ n \dot{\neq} m\end{array}}{\bot} (\bot_{\neq})$$

$$\frac{\dot{R}nm}{\dot{R}\bar{n}\bar{m}} (R\sim) \qquad \frac{(n, X) : \varphi}{(\bar{n}, X) : \varphi} (Id) \qquad \frac{}{n \dot{=} m \mid n \dot{\neq} m} (ub)^2$$

[1] $p \in \mathsf{PROP}$.
[2] n y m son dos nominales diferentes en la rama.

Figure 10: Reglas comunes.

Este resultado es una consecuencia de las reglas del tableau:

Lema 5.1.4. *Sea Θ una rama saturada abierta. Si $nm \dot{\in} S$ aparece en Θ entonces $\bar{n}\bar{m} \in \bar{S}$. Si $nm \dot{\notin} S$ aparece en Θ entonces $\bar{n}\bar{m} \notin \bar{S}$.*

Para hablar de adecuidad, necesitamos considerar dos propiedades: completitud y correctitud. Dado un cálculo de tableau $\mathcal{T}$, escribimos $\mathcal{T}(\varphi)$ para referirnos al tableau obtenido corriendo $\mathcal{T}$ en la fórmula de entrada $(n_0, \emptyset) : \varphi$, donde n_0 es el menor nominal en NOM. Definimos:

Definición 5.1.5 (Completitud)**.** *Un cálculo de tableau $\mathcal{T}$ es* completo *si para toda fórmula φ, si $\mathcal{T}(\varphi)$ es abierto entonces φ es satisfactible.*

Definición 5.1.6 (Correctitud)**.** *Un cálculo de tableau $\mathcal{T}$ es* correcto *si para toda fórmula φ, si φ es satisfactible entonces $\mathcal{T}(\varphi)$ es abierto.*

Entonces definimos los modelos inducidos a partir de ramas abiertas.

Definición 5.1.7 (Modelos Inducidos)**.** *Sea Θ una rama abierta. Definimos el* modelo inducido para Θ *como $\mathcal{M}^\Theta = \langle W^\Theta, R^\Theta, V^\Theta \rangle$, donde:*

$$\begin{array}{lcl} W^\Theta & = & \{\bar{n} \mid n \in \Theta\} \\ R^\Theta & = & \{(\bar{n}, \bar{m}) \mid \dot{R}nm \in \Theta\} \\ V^\Theta(p) & = & \{\bar{n} \mid n : p \in \Theta\}. \end{array}$$

En las siguientes subsecciones vamos a presentar cálculos de tableau para $\mathcal{ML}(\langle \mathsf{sb} \rangle)$, $\mathcal{ML}(\langle \mathsf{br} \rangle)$ y $\mathcal{ML}(\langle \mathsf{sw} \rangle)$.

5.1.1 *Sabotage*

La Figura 11 introduce las reglas que, en combinación con las de la Figura 10, forman un cálculo de tableau completo y correcto para $\mathcal{ML}(\langle \mathsf{sb} \rangle)$. Este cálculo, una fórmula $(n, S) : \varphi$ se interpreta como "φ se satisface en el estado nombrado por n en el model variant descripto por el conjunto S de pares removidos".

$$\frac{(n,S) : \Diamond\varphi}{\begin{array}{c}\dot{R}nm \\ nm\dot{\notin}S \\ (m,S) : \varphi\end{array}}\ (\Diamond)^1 \qquad \frac{(n,S) : \Box\varphi \quad \dot{R}nm \quad nm\dot{\notin}S}{(m,S) : \varphi}\ (\Box)$$

$$\frac{(n,S) : \langle \mathsf{sb} \rangle\varphi}{\begin{array}{c}\dot{R}nm \\ nm\dot{\notin}S \\ (m, S \cup nm) : \varphi\end{array}}\ (\langle \mathsf{sb} \rangle)^1 \qquad \frac{(n,S) : [\mathsf{sb}]\varphi \quad \dot{R}nm \quad nm\dot{\notin}S}{(m, S \cup nm) : \varphi}\ ([\mathsf{sb}])$$

[1] m es nuevo.

Figure 11: Reglas de tableau para $\mathcal{ML}(\langle \mathsf{sb} \rangle)$.

Interpretamos las ramas de este cálculo con la siguiente función de cambio: $f : (R, S) \mapsto R \setminus S$. Esto significa que una fórmula $(n, S) : \varphi$ en una rama Θ debería satisfacerse en el model variant inducido $\mathcal{M}^\Theta_S$ definido como $\mathcal{M}^\Theta_S = \langle W^\Theta, R^\Theta_S, V^\Theta \rangle$, donde $R^\Theta_S = R^\Theta \setminus \bar{S}$.

Las reglas usan la notación $nm\dot{\notin}S$. $nm\dot{\notin}S$ especifica que la arista correspondiente al par de nominas (n, m) no debería ser removida en el presente model variant descripto por S. Cuando esta condición está presente como premisa de una regla, se requiere que unos de los disyuntos de $nm\dot{\notin}S$ esté presente en la rama, lo que significa que $n\dot{\neq}x$ o $m\dot{\neq}y$ pertenece a la rama, para todo $xy \in S$.

La regla $(\Diamond)$ captura el comportamiento estándar del operador $\Diamond$, pero agrega una restricción que especifica que el sucesor no ha sido eliminado hasta este punto de la rama. Lo mismo se tiene en cuenta para $(\Box)$. Para cada sucesor m de n en el modelo inicial ($\dot{R}nm$), y solo si la arista entre n y m no ha sido eliminada ($nm\dot{\notin}S$), φ se debe satisfacer en m en el mismo model variant. La regla $([\mathsf{sb}])$ es similar a $(\Box)$, pero φ se debe satisfacer en m en el model variant donde la arista nm es eliminada. La regla $(\langle \mathsf{sb} \rangle)$ se corresponde de la misma manera con $(\Diamond)$.

Ahora vamos a demostrar completitud y correctitud del cálculo para $\mathcal{ML}(\langle \mathsf{sb} \rangle)$.

Lema 5.1.8. *Sea Θ una rama abierta saturada y sea φ una fórmula de $\mathcal{ML}(\langle \mathsf{sb} \rangle)$. Si $(n, S) : \varphi \in \Theta$ entonces $\mathcal{M}^\Theta_S, \bar{n} \models \varphi$.*

Demostración. Sea $(n, S) : \varphi \in \Theta$. Procedemos por inducción estructural en φ.

$\varphi = p$: Por definición, $\bar{n} \in V^\Theta(p)$, entonces $\mathcal{M}^\Theta, \bar{n} \models p$ y $\mathcal{M}^\Theta_S, \bar{n} \models p$.

$\varphi = \neg p$: Por saturación de (Id), $\bar{n} : \neg p \in \Theta$. Dado que Θ es abierta, $\bar{n} : p \notin \Theta$. Por definición, $\bar{n} \notin V^\Theta(p)$, entonces $\mathcal{M}^\Theta, \bar{n} \not\models p$ y $\mathcal{M}^\Theta_S, \bar{n} \not\models p$.

$\varphi = \psi \wedge \chi$ y $\varphi = \psi \vee \chi$: Trivial por hipótesis inductiva.

$\varphi = \Diamond\psi$: Por $(\Diamond)$, Θ contiene $\dot{R}nm$, $nm\dot{\notin}S$ y $(m, S) : \psi$. Queremos ver que $\bar{n}\bar{m} \in R_S^\Theta$. Verificamos lo siguiente:

1. $\bar{n}\bar{m} \in R^\Theta$: esto vale dado que $\dot{R}nm \in \Theta$.
2. $\bar{n}\bar{m} \notin \bar{S}$: verdadero dado que $(nm\dot{\notin}S) \in \Theta$ por Lema 5.1.4.

A partir de $\bar{n}\bar{m} \in R_S^\Theta$, y (por (Id)) $(\bar{m}, S) : \psi \in \Theta$, tenemos que $\mathcal{M}_S^\Theta, \bar{n} \models \Diamond\psi$.

$\varphi = \langle \mathbf{sb} \rangle \psi$: Necesitamos mostrar que $\mathcal{M}_S^\Theta, \bar{n} \models \langle \mathsf{sb} \rangle \psi$, es decir, existe $x \in V^\Theta$ t.q. $\mathcal{M}_{S \cup pq}^\Theta, x \models \psi$, donde $\bar{p} = \bar{n}$ y $\bar{q} = x$. Esto puede ser verificado considerando $(\langle \mathsf{sb} \rangle)$ en vez de $(\Diamond)$ como en el caso anterior.

$\varphi = \Box\psi$: Solo consideramos los estados $x \in W^\Theta$ tal que $\bar{n}x \in R_S^\Theta$. Esto es, existen a, b tal que $\dot{R}ab \in \Theta$ y $\bar{n}x = \bar{a}\bar{b}$, y $\bar{n}x \notin \bar{S}$. La condición de la regla $(\Box)$ $(nm\dot{\notin}S)$ permite aplicarla en esos pares de nominales.

Por (Id), $(\bar{n}, S) : \Box\psi \in \Theta$, es decir, $(\bar{a}, S) : \Box\psi \in \Theta$, y también por $(R\sim)$, $\dot{R}\bar{a}\bar{b} \in \Theta$.

Por $(\Box)$ tenemos $(\bar{b}, S) : \psi \in \Theta$. Sabemos, $\bar{b} = x$, entonces $\mathcal{M}_S^\Theta, x \models \psi$. Luego para todo $x \in V^\Theta$ tal que $\bar{n}x \in R_S^\Theta$, $\mathcal{M}_S^\Theta, x \models \psi$, es decir, $\mathcal{M}_S^\Theta, \bar{n} \models \Box\psi$.

$\varphi = [\mathbf{sb}]\psi$: Necesitamos mostrar que $\mathcal{M}_S^\Theta, \bar{n} \models [\mathsf{sb}]\psi$, es decir, para todo $x \in W^\Theta$ tal que $(\bar{n}, x) \in R_S^\Theta$, $\mathcal{M}_{S \cup pq}^\Theta, x \models \psi$, donde $\bar{p}\bar{q} = \bar{n}x$. Esto puede ser verificado considerando la regla $([\mathsf{sb}])$ en vez de $(\Box)$ como en el caso anterior.

□

Del lema anterior obtenemos:

Teorema 5.1.9 (Completitud). *Si $\mathcal{T}(\varphi)$ es abierta, entonces φ es satisfactible.*

Ahora mostramos correctitud del cálculo para $\mathcal{ML}(\langle \mathsf{sb} \rangle)$.

Lema 5.1.10. *Sea Γ un conjunto de fórmulas de tableau satisfactibles, y sea $\varphi \in \mathcal{ML}(\langle \mathsf{sb} \rangle)$. Si existe un tableau cerrado $\mathcal{T}(\Gamma')$ para $\Gamma' = (\Gamma \cup \{\neg\varphi\})$, entonces φ es satisfactible.*

Demostración. Sea Θ una rama satisfactible. Siguiendo la Definición 5.1.2, Θ se satisface en un modelo $\mathcal{M} = \langle W, R, V \rangle$ y un mapeo $\sigma : \mathsf{NOM} \mapsto W$. Escribimos $\sigma[m \mapsto v]$ para referirnos al mapeo igual a σ excepto, quizás, $\sigma(m) = v$.

Asumamos que existe un tableau cerrado $\mathcal{T}(\Gamma')$ tal que $\Gamma' = (\Gamma \cup \{\neg\varphi\})$. Vamos a demostrar que Γ' es insatisfactible, por inducción en la estructura del tableau.

- $(\bot_{atom})$: Si podemos aplicar la regla, entonces $n : a \in \Gamma'$ and $n : \neg a \in \Gamma'$, para algún n, a. Entonces Γ' es trivialmente insatisfactible.
- Las reglas comunes $(\bot_{\neq})$, $(\wedge)$, $(\vee)$, $(R\sim)$, (Id) y (ub) son fáciles de verificar.

Resta por verificar que para cada regla restante, la aplicación de ellas nos genera al menos una rama satisfactible.

- $(\Diamond)$: Supongamos $(n, S) : \Diamond\varphi \in \mathcal{T}(\Gamma')$. Sabemos que $(n, S) : \Diamond\varphi$ es satisfactible, entonces existe un modelo $\mathcal{M} = \langle W, R, V\rangle$, y un mapeo $\sigma : \mathsf{NOM} \mapsto W'$ t.q. $\mathcal{M}^-_{S^\sigma}, \sigma(n) \models \Diamond\varphi$. Por definición de $\models$, existe $v \in W$ t.q. $\sigma(n)v \in R \setminus S^\sigma$ y $\mathcal{M}^-_{S^\sigma}, v \models \varphi$. La regla $(\Diamond)$ genera $\dot{R}nm$, $nm\dot{\notin}S$ y $(m, S) : \varphi$, con m nuevo en la rama. Necesitamos verificar que la rama que contiene esas nuevas tres fórmulas es satisfactible. Es decir, que existe un modelo y un mapeo que los satisfacen.

  Consideremos el mapeo $\sigma' = \sigma[m \mapsto v]$ y veamos que la interpretación $\mathcal{M}, \sigma'$ satisface la nueva rama:

  - $\dot{R}nm$ se satisface dado que $R\sigma'(n)\sigma'(m)$, es decir, $R\sigma(n)v$, se satisface.
  - Consideremos $nm\dot{\notin}S$. Es suficiente ver que para todo $xy \in S$, $\sigma'(n)\sigma'(m) \neq \sigma'(x)\sigma'(y)$, es decir, $\sigma(n)v \neq \sigma'(x)\sigma'(y)$. Pero $\sigma(n)v = \sigma'(x)\sigma'(y)$ estaría en contradicción con $\sigma(n)v \in R \setminus S^\sigma$.
  - $\mathcal{M}, \sigma'$ satisface $(m, S) : \varphi$ dado que $\mathcal{M}^-_{S^{\sigma'}}, \sigma'(m) \models \varphi$ se satisface.

- $(\langle\mathbf{sb}\rangle)$: Similar a $(\Diamond)$, excepto que necesitamos ver que la nueva fórmula de tableau $(m, S \cup nm) : \varphi$ se satisface. Esto se hace considerando el nuevo mapeo $\sigma' = \sigma[m \mapsto v]$ y observando que $\mathcal{M}^-_{S\cup nm^{\sigma'}}, \sigma'(m) \models \varphi$.

- $(\Box)$: Supongamos que $(n, S) : \Box\varphi$ y $\dot{R}nm$ aparecen en Θ, y que la condición $nm\dot{\notin}S$ se satisface. Esto implica que existe $\mathcal{M} = \langle W, R, V\rangle$ y un mapeo σ tal que $\mathcal{M}^-_{S^\sigma}, \sigma(n) \models \Box\varphi$, y $R\sigma(n)\sigma(m)$, y no existe par de nominales $xy \in S$ tal que $nm = xy$. Esto nos dice que para todo $v \in W$ t.q. $\sigma(n)v \in (R \setminus S^\sigma)$, $\mathcal{M}^-_{S^\sigma}, v \models \varphi$ y existe $v \in W$ t.q. $R\sigma(n)v$. Vamos a verificar que $(m, S) : \varphi$ es satisfecha por $\mathcal{M}, \sigma$. Dado que $\sigma(n)\sigma(m) \in (R \setminus S^\sigma)$, entonces $\mathcal{M}^-_{S^\sigma}, \sigma(m) \models \varphi$. Luego $(m, S) : \varphi$ es satisfecha por $\mathcal{M}, \sigma$.

- $([\mathbf{sb}])$: Es similar al caso $(\Box)$, pero tenemos que mostrar que $\mathcal{M}, \sigma$ satisface $(m, S \cup nm) : \varphi$. Esto lo podemos hacer observando que si $\mathcal{M}^-_{S^\sigma}, \sigma(n) \models [\mathsf{sb}]\varphi$ y $R\sigma(n)\sigma(m)$ entonces $\mathcal{M}^-_{S\cup nm^\sigma}, \sigma(m) \models \varphi$.

□

Del lema anterior obtenemos:

Teorema 5.1.11 (Correctitud). *Si φ es satisfactible, entonces $\mathcal{T}(\varphi)$ es abierto.*

Podemos adaptar las reglas anteriores para $\mathcal{ML}(\langle\mathsf{gsb}\rangle)$. Las adaptaciones se muestran en la Figura 12.

5.1.2 *Bridge*

La Figura 13 presenta reglas para el cálculo correspondiente a $\mathcal{ML}(\langle\mathsf{br}\rangle)$ que deben ser combinadas con las reglas comunes de la Figura 10. La principal diferencia con las reglas para sabotage es que aquí el prefijo B indica los pares que deben ser agregados al modelo original.

La función de interpretación es $f : (R, B) \mapsto R \cup B$. Es decir,'que una fórmula $(n, B) : \varphi$ en una rama Θ debería ser satisfecha en el model variant inducido $\mathcal{M}^\Theta_B$ definido como $\mathcal{M}^\Theta_B = \langle W^\Theta, R^\Theta_B, V^\Theta\rangle$, donde $R^\Theta_B = R^\Theta \cup \bar{B}$.

$$\frac{(n,S) : \langle \mathsf{gsb} \rangle \varphi}{\begin{array}{c} \dot{R}pq \\ pq \dot{\notin} S \\ (n, S \cup pq) : \varphi \end{array}} (\langle \mathsf{gsb} \rangle)^1 \qquad \frac{\begin{array}{c} (n,S) : [\mathsf{gsb}] \varphi \\ \dot{R}pq \\ pq \dot{\notin} S \end{array}}{(n, S \cup pq) : \varphi} ([\mathsf{gsb}])$$

[1] p y q son nuevos en la rama.

Figure 12: Reglas de tableau para $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$.

$$\frac{(n,B) : \Diamond \varphi}{\begin{array}{c|c} \dot{R}nm & nm \dot{\in} B \\ (m,B) : \varphi & (m,B) : \varphi \end{array}} (\Diamond)^1 \qquad \frac{\begin{array}{c} (n,B) : \Box \varphi \\ \dot{R}nm \end{array}}{(m,B) : \varphi} (\Box) \qquad \frac{\begin{array}{c} (n,B) : \Box \varphi \\ nm \dot{\in} B \end{array}}{(m,B) : \varphi} (\Box_2)$$

$$\frac{\begin{array}{c} \dot{R}nm \\ (a,B) : \varphi \\ nm \dot{\in} B \end{array}}{\bot} (R_\bot) \qquad \frac{(n,B) : \langle \mathsf{br} \rangle \varphi}{\begin{array}{c} nm \dot{\notin} B \\ (m, B \cup nm) : \varphi \end{array}} (\langle \mathsf{br} \rangle)^1 \qquad \frac{\begin{array}{c} (n,B) : [\mathsf{br}] \varphi \\ nm \dot{\notin} B \end{array}}{(m, B \cup nm) : \varphi \mid \dot{R}nm} ([\mathsf{br}])^2$$

[1] m es nuevo.
[2] m aparece en la rama.

Figure 13: Reglas de tableau para $\mathcal{ML}(\langle \mathsf{br} \rangle)$.

La regla $(\Diamond)$, cuando es aplicada a una fórmula $(n, B) : \Diamond \varphi$, tiene que asegurar que en el model variant descripto por B, el estado nombrado por el nominal n tiene un sucesor donde φ se satisfaga. Este model variant tiene una relación que es la unión de la relación en el modelo inicial y B.

La regla $(\Box)$ es la regla estándar de box para lógica modal básica. Se completa con la regla $(\Box_2)$ que asegura que tengamos en cuenta las aristas nuevas que agregamos.

La nueva regla de clash $(R_\bot)$ asegura que cada vez que una arista nm aparece en B representando algún model variant, la misma arista no aparezca en el modelo original, es decir, está prohibido que $\dot{R}nm$ aparezca en la rama.

La regla $(\langle \mathsf{br} \rangle)$ es diferente a $(\Diamond)$. Esto se debe a que $\langle \mathsf{br} \rangle$ mueve la evaluación a un estado que *no* es accesible desde el estado actual, por eso introduce $nm \dot{\notin} B$ y $(m, B \cup nm) : \varphi$ a la rama. Esta última fórmula, junto con la regla $(R_\bot)$, asegura que la arista nm no pertenece al modelo inicial.

La regla $([\mathsf{br}])$ genera dos ramas cuando se aplica a una fórmula $(n, B) : [\mathsf{br}] \varphi$. Esta decide, para todo nominal m tal que $nm \dot{\notin} B$, si $(m, R \cup nm) : \varphi$ se satisface, o $\dot{R}nm$ se satisface. En el primer caso, junto con la regla $(R_\bot)$, asegura que la arista nm no está en el modelo original. En el segundo caso, asegura lo contrario, que no es posible agregar una arista hacia m y φ no necesita ser satisfecha en m.

Completitud y correctitud pueden ser demostrados de manera similar a sabotage.

De la misma manera que fue realizado para $\mathcal{ML}(\langle \mathsf{gsb} \rangle)$, es posible adaptar las reglas para el caso $\mathcal{ML}(\langle \mathsf{gbr} \rangle)$. En la Figura 14 se muestran dichas adaptaciones.

$$\frac{(n,B) : \langle \mathsf{gbr} \rangle \varphi}{\begin{array}{c} pq \dot{\notin} B \\ (n, B \cup pq) : \varphi \end{array}} (\langle \mathsf{gbr} \rangle)^1 \qquad \frac{\begin{array}{c}(n,B) : [\mathsf{gbr}]\varphi \\ pq \dot{\notin} B \end{array}}{(n, B \cup pq) : \varphi \ \mid \ \dot{R}pq} ([\mathsf{gbr}])$$

[1] p y q son nuevas en la rama.

Figure 14: Reglas de tableau para $\mathcal{ML}(\langle \mathsf{gbr} \rangle)$.

5.1.3 *Swap*

Las reglas para swap son introducidas en la Figura 15, y son usadas en combinación con la reglas de la Figura 10.

$$\frac{(n,S) : \Diamond \varphi}{\begin{array}{c|c} \dot{R}nm & nm \dot{\in} S^{-1} \\ nm \dot{\notin} S & (m,S) : \varphi \\ (m,S) : \varphi & \end{array}} (\Diamond)^1 \qquad \frac{\begin{array}{c}(n,S) : \Box \varphi \\ \dot{R}nm \\ nm \dot{\notin} S \end{array}}{(m,S) : \varphi} (\Box) \qquad \frac{\begin{array}{c}(n,S) : \Box \varphi \\ nm \dot{\in} S^{-1} \end{array}}{(m,S) : \varphi} (\Box_2)$$

$$\frac{(n,S) : [\mathsf{sw}]\varphi}{\begin{array}{c} \dot{R}nn \\ (n,S) : \varphi \end{array}} ([\mathsf{sw}]) \qquad \frac{\begin{array}{c}(n,S) : [\mathsf{sw}]\varphi \\ \dot{R}nm \\ n \dot{\neq} m \\ nm \dot{\notin} (S \cup S^{-1}) \end{array}}{(m, S \cup nm) : \varphi} ([\mathsf{sw}]_2) \qquad \frac{\begin{array}{c}(n,S) : [\mathsf{sw}]\varphi \\ xy \in S \\ n \doteq y \end{array}}{(x, S \backslash xy \cup yx) : \varphi} ([\mathsf{sw}]_3)$$

$$\frac{(n,S) : \langle \mathsf{sw} \rangle \varphi}{\begin{array}{c|c|c} \dot{R}nn & \dot{R}nm & \bigvee_{xy \in S} (n \doteq y \wedge (x, S \backslash xy \cup yx) : \varphi) \\ (n,S) : \varphi & n \dot{\neq} m & \\ & nm \dot{\notin} (S \cup S^{-1}) & \\ & (m, S \cup nm) : \varphi & \end{array}} (\langle \mathsf{sw} \rangle)^1$$

[1] m es nuevo.

Figure 15: Reglas de tableau para $\mathcal{ML}(\langle \mathsf{sw} \rangle)$.

Estas reglas tienen que manejar el hecho de que intercambiando aristas en un modelo puede hacer que algunas aristas del modelo original ya no sean utilizables (como cuando se utiliza sabotage), y se pueden general nuevas aristas (como con bridge).

Un prefijo S es el conjunto de aristas que ya no pertenecen al model variant. S^{-1} contiene las aristas que tienen que ser agregadas en el modelo.

La función de interpretación de este cálculo es $f : (R, S) \mapsto (R \setminus S) \cup S^{-1}$. Esto significa que la fórmula $(n, S) : \varphi$ en una rama Θ se debería satisfacer en el model variant inducido $\mathcal{M}_S^\Theta$ definido como $\mathcal{M}_S^\Theta = \langle W^\Theta, R_S^\Theta, V^\Theta \rangle$, donde $R_S^\Theta = (R^\Theta \setminus \bar{S}) \cup \bar{S}^{-1}$.

En este cálculo, S se mantiene irreflexivo y asimétrico. Más aún, no debe contener dos pares diferentes de nominales que se refieran al mismo eje del modelo inducido. Esto garantiza que los nombres en S puede ser manipulados por el cálculo de la manera esperada, en particular cuando una arista ya invertida, tiene que ser invertida otra vez. $nm\dot{\in}S$ indica que nm ya no está presente en el model variant representado por S. $nm\dot{\in}S^{-1}$ indica que nm tiene que ser agregada al model variant representado por S.

Vamos a examinar las reglas. $(\Diamond)$ es una combinación de las reglas $(\Diamond)$ para sabotage y bridge. $(n, S) : \varphi$ en un estado que es accesible por la relación de accesibilidad inicial, o por una arista invertida (como hacemos en el cálculo para bridge).

La regla $(\Box)$, tal como en el cálculo para sabotage, trabaja sobre todos los estados accesibes desde n en el model variant inicial, excepto cuando se han hecho inaccesibles en el model variant actual. La regla $(\Box_2)$, tal como en el cálculo para bridge, asegura que los nuevos estados accessibles reciben la fórmula φ.

Las restantes reglas merecen una explicación más detallada. Las tres reglas que se encargan de fórmulas de la forma $[\mathsf{sw}]\varphi$ tienen en cuenta invertir una arista reflexiva, invertir una arista irreflexiva que nunca ha sido invertida (ni su inversa), e invertir una arista otra vez. $([\mathsf{sw}])$ invierte aristas reflexivas, por lo tanto el conjunto S no necesita ser modificado. $([\mathsf{sw}]_2)$ invierte aristas irreflexivas que nunca han sida invertidas con anterioridad. Esta regla asegura que S es irreflexivo ($n\dot{\neq}m$), asimétrico ($nm\dot{\notin}S^{-1}$) y que no contiene dos pares de nominales que se refieran a la misma arista ($nm\dot{\notin}S$). Finalmente, $([\mathsf{sw}]_3)$ atraviesa e invierte aristas de S^{-1}. Si $n\dot{=}y$ pertenece a la rama y $xy \in S$ entonces invertimos otra vez la arista yx y terminamos en x. Luego removemos xy de S y agregamos yx. Esta preserva las tres propiedades del conjunto S (irreflexividad, asimetría y nombres no redundantes).

Hay solo una regla $(\langle \mathsf{sw} \rangle)$ pero se encarga de todas las posibilidades tal como las reglas para $[\mathsf{sw}]$.

Vamos a demostrar completitud para el cálculo de $\mathcal{ML}(\langle \mathsf{sw} \rangle)$. Correctitud puede ser mostrado similarmente a sabotage.

Lema 5.1.12. *Sea Θ una rama abierta y saturada, y sea φ una fórmula de $\mathcal{ML}(\langle \mathsf{sw} \rangle)$. Si $(n, S) : \varphi \in \Theta$ entonces $\mathcal{M}_S^\Theta, \bar{n} \models \varphi$.*

Demostración. Sea $(n, S) : \varphi \in \Theta$, procedemos por inducción estructural en φ. Los casos proposicionales y Booleanos son exactamente los mismos que para $\mathcal{ML}(\langle \mathsf{sb} \rangle)$.

$\boldsymbol{\varphi = \Diamond\psi}$: Tenemos dos casos:

1. $\dot{R}nm \in \Theta$, $nm\dot{\notin}S \in \Theta$ y $(m, S) : \psi \in \Theta$. Como $\dot{R}nm \in \Theta$, tenemos que $(\bar{n}, \bar{m}) \in R^\Theta$. Por otro lado, como $nm\dot{\notin}S \in \Theta$ y la rama es abierta y saturada, por Lema 5.1.4, $\bar{n}\bar{m} \notin \bar{S}$. Entonces $\bar{n}\bar{m} \in R_S^\Theta$ y (por (Id)) $(\bar{m}, S) : \psi \in \Theta$. Luego, $\mathcal{M}_S^\Theta, \bar{n} \models \Diamond\psi$.
2. $nm\dot{\in}S^{-1} \in \Theta$ y $(m, S) : \psi \in \Theta$. A partir de la primera sentencia, y por Lema 5.1.4, tenemos $\bar{n}\bar{m} \in \bar{S}$, luego $\bar{n}\bar{m} \in R_S^\Theta$. Con el mismo argumento obtenemos $\mathcal{M}_S^\Theta, \bar{n} \models \Diamond\psi$.

$\varphi = \langle \mathbf{sw} \rangle \psi$: La regla ($\langle \mathsf{sw} \rangle$) tiene tres ramas:

1. $\dot{R}nn \in \Theta$ y $(n, S) \in \Theta$. En este caso $\bar{n}\bar{n} \in R_S^\Theta$, y por (Id) $(\bar{n}, S) : \psi \in \Theta$, entonces tenemos $\mathcal{M}_S^\Theta, \bar{n} \models \langle \mathsf{sw} \rangle \psi$.
2. En la segunda rama, las siguientes fórmulas pertencen a Θ: *a*) $\dot{R}nm$, *b*) $n \dot{\neq} m$, *c*) $nm \dot{\notin} (S \cup S^{-1})$ y *d*) $(m, S \cup nm) : \psi$. *b*) se satisface porque no estamos en el caso anterior. Por *a*) y *c*) (y Lema 5.1.4), tenemos $\bar{n}\bar{m} \in R_S^\Theta$. Por (Id) y *d*), $(\bar{m}, S \cup nm) : \psi \in \Theta$. Luego, $\mathcal{M}_S^\Theta, \bar{n} \models \langle \mathsf{sw} \rangle \psi$.
3. En la tercera rama, existen $x, y \in W^\Theta$, tal que $y \doteq n \in \Theta$ y $(x, S \setminus xy \cup yx) \in \Theta$. Entonces $\bar{y} \doteq \bar{n} \in \Theta$ y por definición $\bar{y}\bar{x} \in R_S^\Theta \otimes$. Pero $(\bar{x}, S \setminus xy \cup yx) : \psi \in \Theta$, entonces $\mathcal{M}^\Theta_{S \setminus xy \cup yx}, \bar{x} \models \psi$. Luego, a partir de esta última condición y por $\otimes$, tenemos $\mathcal{M}_S^\Theta, \bar{n} \models \langle \mathsf{sw} \rangle \psi$.

$\varphi = \Box \psi$: para todo $m \in W^\Theta$ tal que $\dot{R}nm$ and $nm \dot{\notin} S \in \Theta$, tenemos $(m, S) : \psi \in \Theta$. Como Θ es abierta y saturada, por Lema 5.1.4 se satisface que $\bar{n}\bar{m} \notin \bar{S}$, lo cual implica $\bar{n}\bar{m} \in R_S^\Theta$. En otro caso, si $nm \in S^{-1}$, entonces (por definición) $\bar{n}\bar{m} \in R_S^\Theta$. En ambos casos tenemos $(\bar{m}, S) : \psi \in \Theta$. Luego, $\mathcal{M}_S^\Theta, \bar{n} \models \Box \psi$.

$\varphi = [\mathbf{sw}] \psi$: el caso reflexivo es igual al caso para $\Box$. Si en Θ tenemos que $\dot{R}nm$, $n \dot{\neq} m$ y $nm \dot{\notin} (S \cup S^{-1})$, entonces $\bar{n}\bar{m} \in R_S^\Theta$. También tenemos que $(\bar{m}, S \cup nm) : \psi \in \Theta$. Por otro lado, si $xy \dot{\in} S$ y $n \doteq y$ están los dos en Θ, (por definición) $\bar{y}\bar{x} \in R_S^\Theta$, y $(\bar{x}, S \setminus xy \cup yx) : \psi \in \Theta$. Con los tres casos, obtenemos $\mathcal{M}_S^\Theta, \bar{n} \models [\mathsf{sw}] \psi$.

□

Por el lema anterior obtenemos:

Teorema 5.1.13 (Completitud). *Si $\mathcal{T}(\varphi)$ es abierto, entonces φ es satisfactible.*

Las reglas para un cálculo completo y correcto para $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$ son introducidas en la Figura 16).

Todas las lógicas que consideramos pueden forzar modelos infinitos. Como resultado, los cálculos de tableau no necesariamente terminan, ya que no implementan ningún tipo de "loop checking". Sin embargo, los métodos de tableau resultan útiles para diferentes propósitos. En las siguientes secciones, veremos dos aplicaciones: tableaux como procedimiento de construcción de modelos, que junto con model checking sirven como un procedimiento de terminación y verificación para el problema satisfactibilidad, y tableau como un método constructivo para calcular interpolantes para una versión híbrida de $\mathcal{ML}(\langle \mathsf{sb} \rangle)$.

5.2 COMBINANDO PROCEDIMIENTOS

Una pregunta natural es si es posible combinar estos cálculos en un único cálculo que sirva para lógica modal equipada con todos los operadores de cambio a la vez. Podemos obtener fácilmente combinaciones locales-globales de los cálculos para los operadores del mismo tipo: $\mathcal{ML}(\langle \mathsf{sb} \rangle, \langle \mathsf{gsb} \rangle)$, $\mathcal{ML}(\langle \mathsf{br} \rangle, \langle \mathsf{gbr} \rangle)$ y $\mathcal{ML}(\langle \mathsf{sw} \rangle, \langle \mathsf{gsw} \rangle)$, mediante la combinación de las reglas correspondientes de la Sección 5.1. Sin embargo, la otras combinaciones parecen requerir cambios profundos dado que cada tipo de

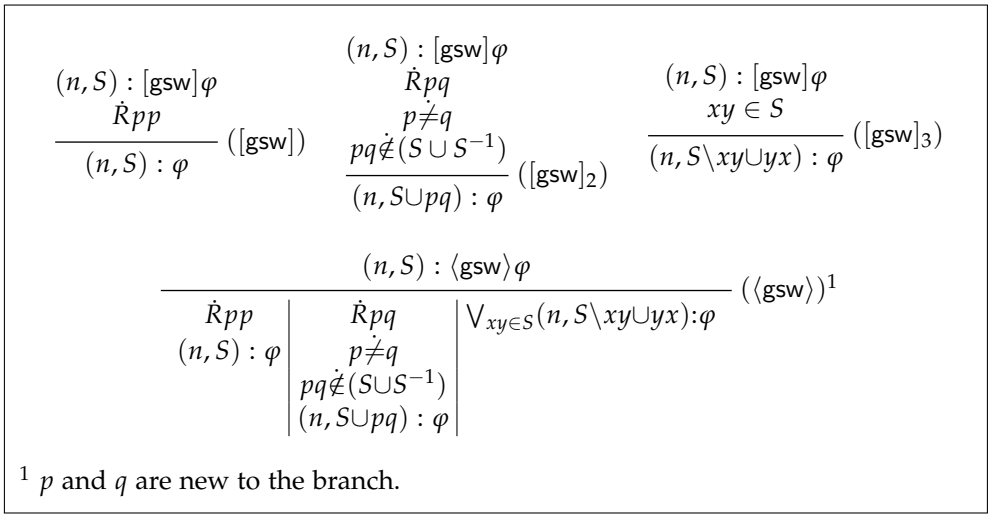


Figure 16: Tableau rules for $\mathcal{ML}(\langle\mathsf{gsw}\rangle)$.

lógica con operadores de cambio (sabotage, bridge, swap) requiere reglas diferentes para los conectores $\Diamond$ and $\Box$.

Cada lógica tiene ciertas de reglas tableau que implican forzar igualdad entre los estados de los modelos inducidos. En particular, las macros de la forma $nm \dot{\in} S$ implican igualdad, y es obvio a partir de la semántica de la lógicas que el razonamiento ecuacional es inevitable. Un resultado de esto es que estas lógicas son lo suficientemente expresivas para describir estructuras complejas. De hecho, es posible encontrar fórmulas que sólo son verdaderas en los modelos que contienen series irreflexivas y transitivas de estados, es decir una cantidad infinita de estados.

Sin embargo, podemos aplicar una técnica simple de terminación y verificación (ver [Areces *et al.*, 2009] para más detalles) a los procedimientos de tableau establecidos en la Sección 5.1. El precio a pagar es que los cálculos resultantes no son completos. La idea es usar tableaux como un procedimiento de construcción de modelos, darlo por terminado después de un cierto número, predefinido de pasos, y entonces hacer model checking de la fórmula de entrada en el modelo inducido. Si el model checking tiene éxito, podemos responder a SAT, si no la respuesta es NOT-KNOWN.

Este procedimento es descripto en el Algoritmo 1 para $\mathcal{ML}(\blacklozenge)$, con $\blacklozenge \in \{\langle\mathsf{sb}\rangle, \langle\mathsf{gsb}\rangle, \langle\mathsf{sw}\rangle, \langle\mathsf{gsw}\rangle, \langle\mathsf{br}\rangle, \langle\mathsf{gbr}\rangle\}$.

En la Sección 4.1 demostramos que model checking para todas las lógicas que investigamos es decidibile y PSPACE-completo, luego el algoritmo siempre termina.

Cada vez que el algoritmo retorna NOT-KNOWN, podemos incrementar el valor del parámetro k y reusar las computaciones realizadas con el valor k más pequeño. De esta manera, podemos aproximar una solución para satisfactibilidad en este tipo de lógicas.

Algorithm 1 SAT Incompleto φ

procedure ISAT(φ, $k \geq 0$)
  Construir $\mathcal{T}(\varphi)$, un tableau con raíz $(n_0, \emptyset) : \varphi$, usando a lo sumo k aplicaciones de las reglas de tableau por rama.
  if $\mathcal{T}(\varphi)$ es cerrado **then**
    return UNSAT
  else
    para $\Theta \in \mathcal{T}(\varphi)$ una rama abierta
    computar el correspondiente $\mathcal{M}^{\Theta}$
    if $\mathcal{M}^{\Theta}, n_0 \models \varphi$ **then**
      return SAT
    else
      return NOT-KNOWN
    end if
  end if
end procedure

5.3 TABLEAUX PARA CALCULAR INTERPOLANTES

El *Lema de Interpolación de Craig* [Craig, 1957] es una propiedad de teoría de modelos que ha sido investigada para muchas lógicas. Esta establece que si $\varphi \rightarrow \psi$ es una validez, entonces $\varphi \rightarrow \theta$ y $\theta \rightarrow \psi$ son válidas, para algún θ en el lenguaje común de φ y ψ. La fórmula θ es llamada un *interpolante*. En [Fitting, 1996; Fitting, 2002; Blackburn and Marx, 2003], el lema de Interpolación es demostrado constructivamente mostrando como extraer un interpolante de una prueba por tableaux de la validez $\varphi \rightarrow \psi$. El interpolante es computado por algoritmos basados en tableau, adaptando las reglas para guardar información adicional. Para $\mathcal{HL}(@, \downarrow)$, las fórmulas de tableau contienen prefijos, que indican en que punto del modelo tenemos que evaluar la fórmula. Con alguna información adicional acerca de la polaridad de las fórmulas, es posible computar un interpolante sistemáticamente.

Las fórmulas de tableau de la Definición 5.1.1 contienen refijos que indican en que punto del modelo tenemos que evaluar la fórmula, y también el model variant actual. Esta información sintáctica contenida en los prefijos es de ayuda para construir interpolantes. Pero los prefijos no son directamente expresables en lógicas con operadores de cambio. Una alternativa es extender estas lógicas con operadores híbridos, y adaptar las reglas de tableau para computar los interpolantes. Tenemos algunos resultados preliminares acerca de como obtener interpolación constructiva para una versión híbrida de sabotage ($\mathcal{HL}(@, \downarrow)+\langle \mathsf{sb} \rangle$), pero queda como trabajo futuro continuar esta investigación.

6

Operadores de Cambio en Lógica Dinámica Epistémica

The problem that we thought was a problem was, indeed, a problem,
but not the problem we thought was the problem.

Mike Smith.

Las lógicas dinámicas epistémicas son casos particulares de lógicas modales con cambios en la relación. El conocimiento de un agente se puede representar por su información accesible, y cambiando el acceso resulta en cambio de la información. Para consultar los orígenes de esta familia de lógicas ver [von Wright, 1951; Hintikka, 1962; Kripke, 1963]. Para conocer acerca de estos leguajes se puede consultar [Plaza, 2007; van Ditmarsch *et al.*, 2007], y para trabajos relacionados a esta tesis ver [Aucher *et al.*, 2009; Kooi and Renne, 2011a; Kooi and Renne, 2011b]. Para resultados de complejidad sobre esta familia de lógicas, ver [Lutz, 2006; French *et al.*, 2013; Aucher and Schwarzentruber, 2013].

Por esta razón, la comunicación de ciertos anuncios o la ejecución de acciones (que produce cambios de información) se puede modelar como modificaciones a la relación de accesibilidad de un modelo. La única diferencia entre $\mathcal{DEL}$ y las lógicas que estudiamos anteriormente en esta tesis, es que en $\mathcal{DEL}$ le damos un significado al modelo (información) y a las fórmulas (conocimientos), mientras que las lógicas modales con operadores de cambio de accesibilidad son más abstractas, y las investigamos desde un punto de vista puramente matemático. Estamos interesados en conocer qué tienen en común utilizando la experiencia obtenida anteriormente.

En este capítulo vamos a introducir un nuevo lenguaje con operadores de cambio de accesibilidad que codifica $\mathcal{DEL}$. Este lenguaje es una extensión de la lógica modal básica $\mathcal{ML}$ con dos operadores: $[\![\ ]\!]$ es una modalidad que elimina aristas del modelo bajo ciertas circunstancias, y $\mathsf{cp}_{\bar{p}}$ es un operador que crea diferentes copias del modelo y etiqueta cada copia con exactamente un símbolo proposicional p de la sequencia $\bar{p}$. Claramente este lenguaje (que llamamos $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$ es una ejemplo de la clase de lógicas que estudiamos en esta tesis. Vamos a mostrar que podemos codificar una versión restringida de $\mathcal{AML}$ con $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$, y vamos a estudiar algunas propiedades del nuevo lenguaje.

6.1 Una Lógica con Borrado y Copia

Hemos investigado varias primitivas para cambiar la relación de accessibilidad. Definimos operadores para borrar, invertir y agregar aristas en un modelo, local y globalmente, pero nunca las hemos combinado. En esta sección, intrucimos $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$,

un lenguaje que puede borrar aristas y crear copias de un modelo. El borrado se realiza de acuerdo a un parámetro caracterizado por una expresión de camino π. Introducimos a continuación la sintaxis formal de $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$.

Definición 6.1.1 (Sintaxis)**.** *Sea* PROP*, un conjunto infinito y contable de símbolos de proposición, y sea* AGT *un conjunto finito de agentes, definimos* FORM *el conjunto de fórmulas de* $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$*, junto con el conjunto* PATH *de expresiones de camino.*

$$\mathsf{FORM} ::= \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \Diamond_a \varphi \mid [\![\pi]\!]\varphi \mid \mathsf{cp}_{\bar{p}}\varphi,$$

donde $\bar{p} \in \mathsf{PROP}^+$ *no aparece en ninguna ocurrencia de* cp *en* φ*,* $a \in \mathsf{AGT}$*,* $\varphi, \varphi' \in \mathsf{FORM}$*, y* $\pi \in \mathsf{PATH}$*.*

$$\mathsf{PATH} ::= a \mid \pi; \pi' \mid \varphi?,$$

donde $a \in \mathsf{AGT}$*,* $\pi, \pi' \in \mathsf{PATH}$ *y* φ *es una fórmula Booleana.*

$[\![\pi]\!]\varphi$ es de la clase del operador de sabotage e intuitivamente siginifica que φ se satisface después de eliminar las aristas que aparecen en un camino que se corresponda con π. Definimos $[\![\pi \odot \pi']\!]\varphi$ una abreviación para $[\![\pi]\!][\![\pi']\!]\varphi$. El operador $\mathsf{cp}_{\bar{p}}\varphi$ indica que φ es verdadera en el modelo obtenido replicando el modelo inicial. Definimos la semántica.

Definición 6.1.2 (Modelos)**.** *Un modelo multimodal (o multiagente)* $\mathcal{M}$ *es una terna* $\mathcal{M} = \langle W, R, V\rangle$*, donde* W *es un conjunto no vacío;* $R \subseteq \mathsf{AGT} \times W^2$ *es una relación de accesibilidad; (dado* $a \in \mathsf{AGT}$*, escribimos* R_a *para referirnos a* $\{(w, v) \in W^2 \mid (a, w, v) \in R\}$*); y* $V : \mathsf{PROP} \to \mathcal{P}(W)$ *es una valuación.*

Representamos un camino como una secuncia $w_0a_0w_1a_1 \ldots w_{n-1}a_{n-1}w_n$ donde w_i son estados y a_i son agentes. Definamos el conjunto $\mathcal{P}^{\mathcal{M}}_{\pi}$ de π-caminos en el modelo $\mathcal{M}$ por inducción en π. $\mathcal{P}^{\mathcal{M}}_{a}$ contiene caminos representando a-aristas. $\mathcal{P}^{\mathcal{M}}_{\pi;\pi'}$ contiene concatenaciones de un π-camino y un π'-camino. En tal concatenación, el último estado w del π-camino tiene que ser el primer estado del π'-camino. $\mathcal{P}^{\mathcal{M}}_{\varphi?}$ contiene caminos de longitud o, compuestos por un estado que satisface φ.

Definición 6.1.3 (Caminos)**.** *Sea* $\mathcal{M} = \langle W, R, V\rangle$ *un modelo multiagente,* $\pi \in \mathsf{PATH}$*. Definimos el conjunto de* π*-caminos* $\mathcal{P}^{\mathcal{M}}_{\pi}$ *de* $\mathcal{M}$ *inductivamente:*

$$\begin{array}{lcl} \mathcal{P}^{\mathcal{M}}_{a} & = & \{wau \mid (w, u) \in R_a\} \\ \mathcal{P}^{\mathcal{M}}_{\pi;\pi'} & = & \{SwS' \mid Sw \in \mathcal{P}^{\mathcal{M}}_{\pi} \text{ y } wS' \in \mathcal{P}^{\mathcal{M}}_{\pi'}\} \\ \mathcal{P}^{\mathcal{M}}_{\varphi?} & = & \{w \mid \mathcal{M}, w \models \varphi\}. \end{array}$$

Sea $a \in \mathsf{AGT}$*, definimos* $edges_a(P)$ *al conjunto de todas las a-aristas del camino P. Formalmente:*

$$edges_a(P) = \{(a, w, u) \mid wau \text{ es una subsecuencia de } P\}.$$

Definición 6.1.4 (Model Variants)**.** *Dados un modelo* $\mathcal{M} = \langle W, R, V\rangle$ *y un* π*-camino, definimos el model variant donde eliminamos las aristas que corresponden con un* π*-camino en la relación de accesibilidad:*

$$\mathcal{M}_{[\![\pi]\!]} = \langle W, R_{[\![\pi]\!]}, V\rangle,$$

donde

$$R_{[\![\pi]\!]} = R \setminus \bigcup_{a \in \mathsf{AGT}, P \in \mathcal{P}_\pi^{\mathcal{M}}} edges_a(P).$$

Sea $\bar{p} \in \mathsf{PROP}^+$*, definimos* $\mathcal{M}_{\bar{p}} = \langle W_{\bar{p}}, R_{\bar{p}}, V_{\bar{p}} \rangle$*, donde*

$$\begin{array}{lcl} W_{\bar{p}} & = & W \times \{1, \ldots, |\,\bar{p}\,|\} \;\; (\textit{llamamos } w_i \textit{ a cada } (w,i)) \\ R_{\bar{p}} & = & \{(a, w_i, v_j) \mid (a,w,v) \in R \wedge i, j \leq |\,\bar{p}\,|\} \\ V_{\bar{p}}(\bar{p}(i)) & = & \{w_i \in W_{\bar{p}} \mid i \leq |\,\bar{p}\,|\} \\ V_{\bar{p}}(p) & = & \{w_i \in W_{\bar{p}} \mid w \in V(p)\} \textit{ si } p \notin \bar{p}. \end{array}$$

Definimos al semántica de los operadores de la Definición 6.1.1.

Definición 6.1.5 (Semántica)**.** *Sea un pointed model* $\mathcal{M}, w$ *y una fórmula* φ *decimos que* $\mathcal{M}, w$ *satisface* φ*, y escribimos* $\mathcal{M}, w \models \varphi$*, donde*

$$\begin{array}{lll} \mathcal{M}, w \models p & \textit{sii} & w \in V(p) \\ \mathcal{M}, w \models \neg\varphi & \textit{sii} & \mathcal{M}, w \not\models \varphi \\ \mathcal{M}, w \models \varphi \wedge \psi & \textit{sii} & \mathcal{M}, w \models \varphi \textit{ y } \mathcal{M}, w \models \psi \\ \mathcal{M}, w \models \Diamond_a \varphi & \textit{sii} & \textit{existe } v \in W \textit{ t.q. } (w,v) \in R_a,\ \mathcal{M}, v \models \varphi \\ \mathcal{M}, w \models [\![\pi]\!]\varphi & \textit{sii} & \mathcal{M}_{[\![\pi]\!]}, w \models \varphi \\ \mathcal{M}, w \models \mathsf{cp}_{\bar{p}}\varphi & \textit{sii} & \mathcal{M}_{\bar{p}}, w_1 \models \varphi. \end{array}$$

φ es satisfactible si para algún pointed model $\mathcal{M}, w$ tenemos que $\mathcal{M}, w \models \varphi$.

La Figura 6.1 muestra como trabaja $\mathsf{cp}_{\bar{p}}$, replicando el modelo original y etiquetando cada copia con un símbolo de proposición particular de la secuencia $\bar{p}$. Para cada estado se mantienen los sucesores del modelo original, incluso entre diferentes copias.

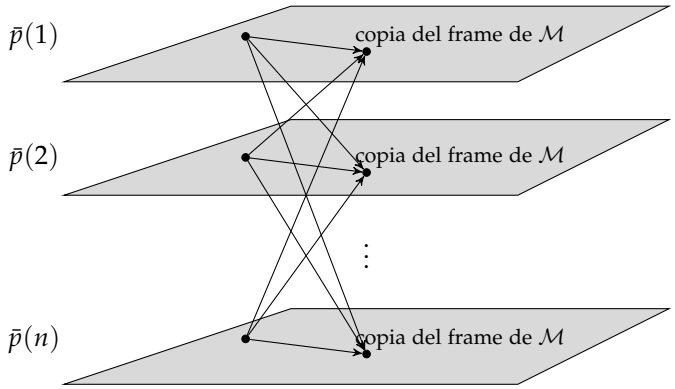


Figure 17: Semántica del operador de copia.

Vamos a estudiar algunas propiedades del nuevo lenguaje, como por ejemplo que $[\![\ ]\!]$ es self-dual.

Proposición 6.1.6 (Self-dualitdad)**.** *Sea* $\mathcal{M}$ *un modelo,* φ *una fórmula de* $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$ *y* $\pi \in \mathsf{PATH}$*. Entonces*

$$\mathcal{M}, w \models \neg[\![\pi]\!]\neg\varphi \textit{ sii } \mathcal{M}, w \models [\![\pi]\!]\varphi.$$

Demostración. Supongamos $\mathcal{M}, w \models \neg[\![\pi]\!]\neg\varphi$. Por $\models$ tenemos $\mathcal{M}, w \not\models [\![\pi]\!]\neg\varphi$, sii $\mathcal{M}_{[\![\pi]\!]}, w \not\models \neg\varphi$. Entonces, tenemos $\mathcal{M}_{[\![\pi]\!]}, w \models \neg\neg\varphi$, sii (por $\models$ y doble negación) $\mathcal{M}, w \models [\![\pi]\!]\varphi$. □

Una herramienta importante es bisimulación. Las nuevas condiciones para bisimilaridad necesitan hablar de información en el pasado. Tomando la Definición 2.1.1, tenemos que agregar las condiciones agregadas para el operador $\Diamond^{-1}$ en [Blackburn *et al.*, 2001]:

(zig^{-1}) si $(v, w) \in R_a$ entonces existe v', $(v', w') \in R'_a$ y vZv';

(zag^{-1}) si $(v', w') \in R'_a$ entonces existe v, $(v, w) \in R_a$ y vZv'.

Denotamos $\leftrightarrows_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])}$ cuando nos referimos a bisimulaciones para $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$. Sin pérdida de generalidad asumimos que cada borrado es de la forma

$$[\![\varphi_1?; a_1; \varphi_2?; a_2; \ldots; a_{n-1}; \varphi_n?]\!]\psi,$$

donde $\varphi_i?$ son fórmulas Booleanas arbitrarias, y $a_i \in \mathsf{AGT}$ (podemos introducir $\top?$ y conjugar sucesivas fórmulas Booelenas para transformar cualquier expresión en esta forma normal). El siguiente lema establece que bisimulación preserva caminos.

Lema 6.1.7. *Sea $\mathcal{M} = \langle W, R, V\rangle$ y $\mathcal{M}' = \langle W', R', V'\rangle$ dos modelos, $w \in W$ y $w' \in W'$, tal que $\mathcal{M}, w \leftrightarrows_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}', w'$. Sea $\pi = \varphi_1?; a_1; \varphi_2?; a_2; \ldots; a_{n-1}; \varphi_n?$ arbitrario. Entonces, para todo $P \in \mathcal{P}^{\mathcal{M}}_{\pi}$ tal que $P = w_0 a_0 \ldots w a_i \ldots w_n$, existe $P' \in \mathcal{P}^{\mathcal{M}'}_{\pi}$, con $P' = w'_0 a_0 \ldots w' a_i \ldots w'_n$ y para todo $j \in \{1, \ldots, n\}$ tenemos $\mathcal{M}, w_j \leftrightarrows_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}', w'_j$.*

Demostración. Sea $P \in \mathcal{P}^{\mathcal{M}}_{\pi}$, necesitamos encontrar $P' \in \mathcal{P}^{\mathcal{M}'}_{\pi}$ satisfaciendo el lema. Construiremos tal P'.

Supongamos $P = w_0 a_0 \ldots w a_i \ldots w_n$. Notar que tenemos el subcamino $w a_i w_{i+1}$, lo cual significa $(w, w_{i+1}) \in R_{a_i}$. Dado que $\mathcal{M}, w \leftrightarrows_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}', w'$, por (zig) existe w'_{i+1} tal que $(w', w'_{i+1}) \in R'_{a_i}$ y $\mathcal{M}, w_{i+1} \leftrightarrows_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}', w'_{i+1}$. Por esta razón, $\mathcal{M}, w_{i+1} \models \psi$ si y sólo si $\mathcal{M}', w'_{i+1} \models \psi$, para todo ψ (en particular φ_{i+1}). Entonces, w_{i+1} es una buena elección para construir P'. Podemos repetir este proceso para construir el subcamino $w' a_i w'_{i+1} \ldots w'_n$. Para elegir w_{i-1}, podemos proceder de la misma manera pero usando (zig^{-1}), y repitiendo el proceso hasta alcanzar w'_1. Uniendo todo, tenemos P'.

Para la otra dirección usar (zag) y (zag^{-1}). □

Otra propiedad es que replicando modelos bisimilares con cp, obtenemos modelos bisimilares.

Lema 6.1.8. *Sea $\mathcal{M} = \langle W, R, V\rangle$ y $\mathcal{M}' = \langle W', R', V'\rangle$ dos modelos, $w \in W$ y $w' \in W'$. Entonces $\mathcal{M}, w \leftrightarrows_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}', w'$ implica $\mathcal{M}_{\bar{p}}, w_1 \leftrightarrows_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}'_{\bar{p}}, w'_1$.*

Demostración. Tenemos que definir una bisimulación $Z \subseteq W_{\bar{p}} \times W'_{\bar{p}}$. Dado que $\mathcal{M}, w \leftrightarrows_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}', w'$, definimos:

$$Z = \{(v_i, v'_i) \mid v_i \in W_{\bar{p}},\ v'_i \in W_{\bar{p}},\ i \in \{1, \ldots, |\bar{p}|\} \text{ t.q. } \mathcal{M}, v \leftrightarrows_{\mathcal{ML}(\mathsf{cp},[\![\ ]\!])} \mathcal{M}', v'\}.$$

(atomic harmony) vale proque $v_i Z v'_i$ si y sólo si v y v' satisfacen (atomic harmony) en el modelo original, y v_i y v'_i están ambos etiquetados por el símbolo $\bar{p}(i)$. Para (zig), supongamos qe tenemos $v_j Z v'_j$ y $(v_j, u_k) \in (R_{\bar{p}})_a$. Entonces sabemos que $(v, u) \in R_a$. Como $\mathcal{M}, v \leftrightarrows_{\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])} \mathcal{M}', v'$, por (zig) existe u' tal que $(v', u') \in R'_a$. Luego, tenemos $(v'_j, u'_k) \in (R'_{\bar{p}})_a$. (zag) es similar. □

Ahora podemos demostrar que modelos $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$-bisimilares satisfacen las mismas fórmulas.

Teorema 6.1.9 (Invarianza por bisimulación.)**.** *Para toda fórmula φ de $\mathcal{ML}(cp, [\![\ ]\!])$, tenemos $\mathcal{M}, w \leftrightarrows_{\mathcal{ML}(cp, [\![\ ]\!])} \mathcal{M}', w'$ implica $\mathcal{M}, w \models \varphi$ sii $\mathcal{M}', w' \models \varphi$.*

Demostración. La demostración es por inducción estructural. Sea $\mathcal{M} = \langle W, R, V \rangle$ y $\mathcal{M}' = \langle W', R', V' \rangle$, tal que $\mathcal{M}, w \leftrightarrows_{\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])} \mathcal{M}', w'$.

Los casos Booleanos y modales son iguales a $\mathcal{ML}$. Resta demostrar los casos inductivos para $[\![\ ]\!]$ y cp.

$[\![\pi]\!]\varphi$: Supongamos $\mathcal{M}, w \models [\![\pi]\!]\varphi$, entonces $\mathcal{M}_{[\![\pi]\!]}, w \models \varphi$, $\mathcal{M}_{[\![\pi]\!]} = \langle W, R_{[\![\pi]\!]}, V \rangle$, $R_{[\![\pi]\!]} = R \setminus \bigcup_{P \in \mathcal{P}^{\mathcal{M}}_{\pi}, a \in \mathsf{AGT}} edges_a(P)$. Como $\mathcal{M}, w \leftrightarrows_{\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])} \mathcal{M}', w'$, por Lema 6.1.7 existe $P \in \mathcal{P}^{\mathcal{M}}_{\pi}$ sii existe $P' \in \mathcal{P}^{\mathcal{M}'}_{\pi}$. Luego $\mathcal{M}_{[\![\pi]\!]}, w \leftrightarrows_{\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])} \mathcal{M}'_{[\![\pi]\!]}, w'$, y por H.I. $\mathcal{M}'_{[\![\pi]\!]}, w' \models \varphi$. Como resultado, $\mathcal{M}', w' \models [\![\pi]\!]\varphi$.

$\mathsf{cp}_{\bar{p}}\varphi$: Supongamos $\mathcal{M}, w \models \mathsf{cp}_{\bar{p}}\varphi$. Entonces tenemos $\mathcal{M}_{\bar{p}}, w_1 \models \varphi$. Como tenemos que $\mathcal{M}, w \leftrightarrows_{\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])} \mathcal{M}', w'$, por Lema 6.1.8 tenemos $\mathcal{M}_{\bar{p}}, w_1 \leftrightarrows_{\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])} \mathcal{M}'_{\bar{p}}, w'_1$. Luego, (por H.I.) $\mathcal{M}'_{\bar{p}}, w'_1 \models \varphi$. Luego, $\mathcal{M}', w' \models \mathsf{cp}_{\bar{p}}\varphi$.

□

El siguiente teorema es una consecuencias de las nuevas distinciones que podemos hacer con esta noción de bisimulación:

Teorema 6.1.10. *El lenguaje $\mathcal{ML}(cp, [\![\ ]\!])$ tiene la tree model property.*

Demostración. $\mathcal{ML}(\Diamond^{-1})$ tiene la tree model property [Blackburn *et al.*, 2001]. Como las condiciones para bisimilaridad de $\mathcal{ML}(\Diamond^{-1})$ y $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$ son las mismas, por Teorema 6.1.9 tenemos que $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$ tiene la tree model property. □

6.2 CODIFICANDO ACTION MODELS CON OPERADORES DE CAMBIO DE ACCESIBILIDAD

Nuestro objetivo es demostrar que podemos definir $\mathcal{DEL}$ en términos de una lógica con operadores de cambio de accesibilidad como los que se investigaron en capítulos previos. Para ello, tomamos una versión restringida de $\mathcal{AML}$ y la codificamos con $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$. La restricción que imponemos a las fórmulas de $\mathcal{AML}$ es que las precondiciones de las acciones en los action models sean fórmulas booleanas.

Definición 6.2.1 (De $\mathcal{AML}$ a $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$)**.** *Sea $\mathcal{E} = \langle E, \to, \mathsf{pre} \rangle$ un action model con $E = \alpha_1, \ldots, \alpha_n$. Definimos la traducción* Tr *de fórmulas de $\mathcal{AML}$ a fórmulas de $\mathcal{ML}(cp, [\![\ ]\!])$ como:*

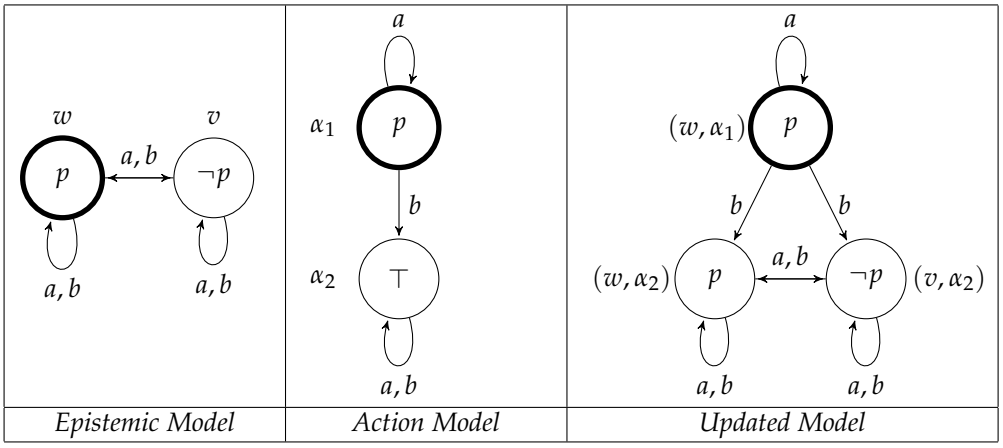
$$\mathsf{Tr}([\mathcal{E}, \alpha_1]\varphi) = \mathsf{pre}(\alpha_1) \to cp_{p_{\alpha_1} \ldots p_{\alpha_n}} [\![\rho]\!][\![\sigma]\!]\mathsf{Tr}(\varphi),$$

donde

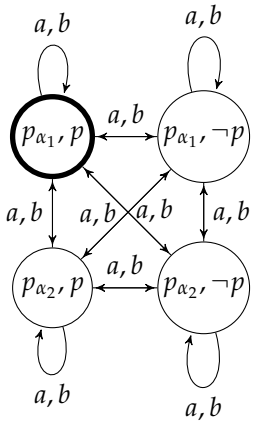
$$\rho \equiv \bigodot_{\alpha_i \in E, a \in \mathsf{AGT}} a; (p_{\alpha_i} \wedge \neg \mathsf{pre}(\alpha_i))?$$
$$\sigma \equiv \bigodot_{\alpha_i, \alpha_j \in E, a \in \mathsf{AGT}} p_{\alpha_i}?; a; p_{\alpha_j}? \quad \text{if } \alpha_i \not\to_a \alpha_j.$$

El Ejemplo 6.2.2 muestra como funciona la codificación en un escenario concreto.

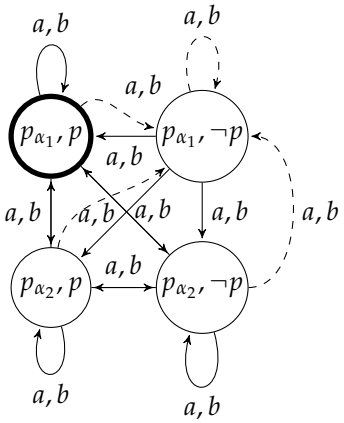
Ejemplo 6.2.2. *Abajo podemos ver un epistemic model $\mathcal{M}$ en la primer columna, un action model $\mathcal{E}$ en la segunda, y el modelo correspondiente después de evaluar $[\mathcal{E}, \alpha_1]$ en $\mathcal{M}, w$.*



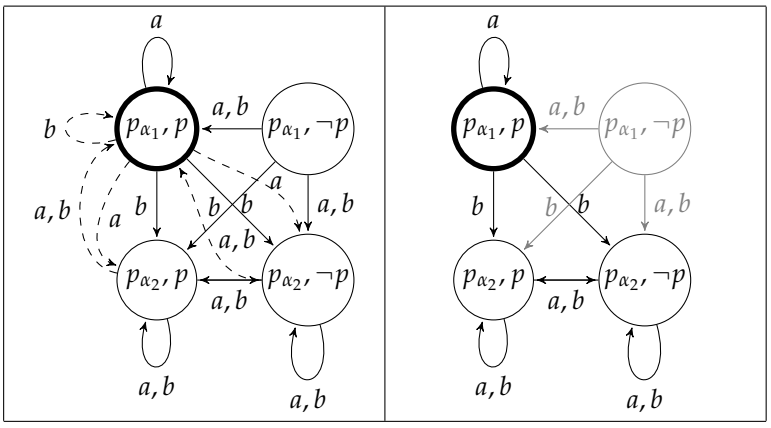
Ahora vamos a ver como aplicando la traducción Tr*, obtenemos el mismo modelo (módulo $\mathcal{AML}$-bisimulación). Vamos a ejecutar paso a paso la transformación, evaluando cada parte de la traducción. El primer paso replica tantas copias del modelo epistémico original, como acciones pertenecientes al dominio del modelo de acción. Hacemos eso con una operación de copia.*



Después evaluamos $[\![\rho]\!]$ (definido como en la Definición 6.2.1), eliminamos todas las aristas apuntando a estados donde p_{α_1} vale y $\mathsf{pre}(\alpha_1)$ *no al mismo tiempo (las aristas eliminadas están representadas por líneas de puntos).*



Finalmente, debemos evaluar $[\![\sigma]\!]$. Esto elimina las aristas que han sido agregadas por la operación de copia, pero que no estaban conectadas en el modelo de acción original. Luego removemos todos los accesos no deseados, obteniendo un modelo bisimilar al de la tercer columna de la primer figura de este ejemplo (el estado etiquetado por $\{p_{\alpha_1}, \neg p\}$ ya no es accesible).



En la siguiente sección vamos a estudiar algunas propiedades computacionales del nuevo lenguaje.

6.3 COMPORTAMIENTO COMPUTACIONAL

Estudiaremos la complejidad de diferentes framentos de $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$.

6.3.1 *Complejidad del fragmento $\mathcal{ML}(\mathsf{cp})$*

En esta subsección, investigamos el fragmento de $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$ donde solo se permiten los operadores $\Box_a$ y $\mathsf{cp}_{\bar{p}}$. A este fragmento lo denotamos $\mathcal{ML}(\mathsf{cp})$.

Sea Σ una secuencia arbitraria de símbolos de proposición. Definimos la traducción Tr_Σ que tranforma fórmulas $\varphi \in \mathcal{ML}(\mathsf{cp})$ en fórmulas de $\mathcal{ML}$ de la siguiente manera:

$$\begin{array}{lcl}
\mathsf{Tr}_\Sigma(p) & = & p \\
\mathsf{Tr}_\Sigma(\neg\varphi) & = & \neg\mathsf{Tr}_\Sigma(\varphi) \\
\mathsf{Tr}_\Sigma(\varphi \wedge \psi) & = & \mathsf{Tr}_\Sigma(\varphi) \wedge \mathsf{Tr}_\Sigma(\psi) \\
\mathsf{Tr}_\Sigma(\Box_a\varphi) & = & \Box_a(\bigwedge_{\text{donde } p_i \text{ es el primer símbolo prop. de una secuencia en } \Sigma} p_i \to \mathsf{Tr}_\Sigma(\varphi)) \\
\mathsf{Tr}_\Sigma(\mathsf{cp}_{\bar{p}}\varphi) & = & \mathsf{Tr}_{\Sigma\setminus\{\bar{p}\}}(\varphi).
\end{array}$$

Definimos por inducción:

$$\begin{array}{lcl}
\mathcal{M}_{\{\bar{p}\}\cup\Sigma} & := & (\mathcal{M}_\Sigma)_{\bar{p}} \\
\mathcal{M}_\emptyset & := & \mathcal{M}.
\end{array}$$

En realidad, la definición de $\mathcal{M}_\Sigma$ no depende del orden de Σ.

Teorema 6.3.1. *Para todo $\varphi \in \mathcal{ML}(\mathsf{cp})$, sea $\Sigma(\varphi)$ el conjunto de todas las secuencias de símbolos proposicionales que aparecen en un operador de copia en φ. Tenemos:*

$$\mathcal{M}, w \models \varphi \;\; sii \;\; \mathcal{M}_{\Sigma(\varphi)}, w_{\Sigma(\varphi)} \models \mathsf{Tr}_{\Sigma(\varphi)}(\varphi),$$

donde $w_{\Sigma(\varphi)}$ es el estado correspondiente al punto de evaluación después de $|\Sigma(\varphi)|$ aplicaciones consecutivas de cp desde w.

Demostración. La demostración es por inducción en la estructura de $\varphi \in \mathcal{ML}(\mathsf{cp})$.

$\varphi = p$: $\mathcal{M}, w \models p$ si y sólo si $\mathcal{M}_\emptyset, w \models \mathsf{Tr}_\emptyset(p)$ (por definición de $\mathcal{M}_\emptyset$ y Tr) si y sólo si (por definición de $\Sigma(\varphi)$) $\mathcal{M}_{\Sigma(p)}, w \models \mathsf{Tr}_{\Sigma(p)}(p)$

$\varphi = \neg\psi$ y **$\varphi = \psi \wedge \chi$:** Son triviales por hipótesis inductiva.

$\varphi = \Box_a\psi$: por definición de $\models$, $\mathcal{M}, w \models \Box_a\psi$ sii para todo v tal que $(w, v) \in R_a$, $\mathcal{M}, v \models \psi$. $\Sigma(\Box_a\psi) = \Sigma(\psi)$. Por H.I., tenemos $\mathcal{M}_{\Sigma(\psi)}, v_{\Sigma(\psi)} \models \mathsf{Tr}_{\Sigma(\psi)}(\psi)$. Esto es lo mismo que $\mathcal{M}_{\Sigma(\Box_a\psi)}, v_{\Sigma(\Box_a\psi)} \models \mathsf{Tr}_{\Sigma(\Box_a\psi)}(\psi)$. Tenemos $\mathcal{M}_{\Sigma(\Box_a\psi)}, w_{\Sigma(\Box_a\psi)} \models \Box_a(\bigwedge_{\text{donde } p_i \text{ es el primer símbolo proposicional de una secuencia en } \Sigma(\Box_a\psi)} p_i \to \mathsf{Tr}_\Sigma(\psi))$, (porque $v_{\Sigma(\Box_a\psi)}$ es una copia arbitraria de los sucesores de w), si y sólo si $\mathcal{M}_\Sigma, w_{\Sigma(\Box_a\psi)} \models \mathsf{Tr}_{\Sigma(\Box_a\psi)}(\Box_a\psi)$.

$\varphi = \mathsf{cp}_{\bar{p}}\psi$: $\mathcal{M}, w \models \mathsf{cp}_{\bar{p}}\psi$ sii $\mathcal{M}_{\bar{p}}, w_1 \models \psi$. por H.I., $(\mathcal{M}_{\bar{p}})_{\Sigma(\psi)}, w_{\Sigma(\varphi)} \models \mathsf{Tr}_{\Sigma(\psi)}(\psi)$. Por definición de $\mathcal{M}_\Sigma$ (que no depende del orden de Σ), y $\bar{p} \notin \Sigma(\psi)$ (porque en cada ocurrencia de $\mathsf{cp}_{\bar{p}}$, $\bar{p}$ es nueva), tenemos $\mathcal{M}_{\Sigma(\psi)\cup\{\bar{p}\}}, w_{\Sigma(\varphi)} \models \mathsf{Tr}_{\Sigma(\psi)\setminus\{\bar{p}\}}(\psi)$. Luego, por definición $\mathcal{M}_{\Sigma(\psi)\cup\{\bar{p}\}}, w_{\Sigma(\varphi)} \models \mathsf{Tr}_{\Sigma(\psi)\cup\{\bar{p}\}}(\mathsf{cp}_{\bar{p}}\psi)$, que es lo mismo que $\mathcal{M}_{\Sigma(\mathsf{cp}_{\bar{p}}\psi)}, w_{\Sigma(\varphi)} \models \mathsf{Tr}_{\Sigma(\mathsf{cp}_{\bar{p}}\psi)}(\mathsf{cp}_{\bar{p}}\psi)$.

□

Vamos a mostrar la cota superior para la clase PSPACE, por medio de un algoritmo basado en tableau que usa espacio polinomial.

Teorema 6.3.2. *El siguiente problema está en* PSPACE*:*

- *entrada: una fórmula $\varphi \in \mathcal{ML}$; Σ un conjunto de secuencias de símbolos proposicionales;*
- *salida: si sii existe un modelo $\mathcal{M}$ tal que $\mathcal{M}_\Sigma, w_\Sigma \models \varphi$.*

Demostración. Vamos a adaptar en método de tableau estándar para $\mathcal{ML}$ (ver [Goré, 1999]) para obtener un procedimiento PSPACE para nuestro problema, mostrado en el Algoritmo 2. ν es llamada *valuación modal* sobre un conjunto de fórmulas Γ si y sólo si $\nu \subseteq \mathsf{PROP} \cup \{\Diamond_a \psi \mid \Diamond_a \psi$ or $\neg\Diamond_a \psi$ en modal depth 1 en $\Gamma\}$. Definimos la relación $\models$ para decir que una valuación satisface una fórmula, como:

$$\begin{array}{lll} \nu \models p & \text{sii} & p \in \nu \\ \nu \models \Diamond_a \varphi & \text{sii} & \Diamond_a \varphi \in \nu \\ \nu \models \neg\varphi & \text{sii} & \nu \not\models \varphi \\ \nu \models \varphi \wedge \varphi' & \text{sii} & \nu \models \varphi \text{ y } \nu \models \varphi'. \end{array}$$

Una valuación c es llamada *valuación de copia* para un conjunto Σ de secuencias de símbolos proposicionales, si y sólo si c contiene exactamente un símbolo proposicional en cada secuencia de Σ. Notar que, dada una valuación modal ν y una valuación de copia c, $\nu \cup c$ es una valuación modal.

Algorithm 2 Satisfactibilidad para el fragmento $\mathcal{ML}(\mathsf{cp})$

```
procedure SAT(φ, Γ, Σ))
    elegir alguna valuación modal ν sobre Γ ∪ {φ}
    for all valuación de copia c sobre Σ do
        if ν ∪ c ⊭ ⋀_{γ_i∈Γ} γ_i then
            return UNSAT
        end if
    end for
    if para ninguna valuación de copia c tenemos ν ∪ c ⊨ φ ∧ ⋀_{γ_i∈Γ} γ_i then
        return UNSAT
    end if
    for all ◇_a ψ ∈ ν do
        if SAT(ψ, {χ | ¬◇_a χ ∈ ν}, Σ) =UNSAT then
            return UNSAT
        end if
    end for
    return SAT
end procedure
```

□

Del resultado anterior, podemos establecer la complejidad del problema de satisfactibilidad para el fragmento $\mathcal{ML}(\mathsf{cp})$:

Teorema 6.3.3. *Decidir si una fórmula de $\mathcal{ML}(cp)$ is satisfactible es* PSPACE*-completo.*

Demostración. PSPACE-hardness se sigue de PSPACE-completitud de satisfactibilidad parar $\mathcal{ML}$. Para demostrar completitud, podemos usar el Teorema 6.3.2, verificando

si existe un modelo $\mathcal{M}$ tal que $\mathcal{M}_{\Sigma(\varphi)}, w_{\Sigma(\varphi)} \models \mathsf{Tr}_{\Sigma(\varphi)}(\varphi)$. Esto puede realizarse (por Teorema 6.3.1) invocando

$$\mathsf{SAT}(\mathsf{Tr}_{\Sigma(\varphi)}(\varphi) \wedge \bigwedge_{p_i, \text{ donde } p_i \text{ es el primer símbolo de prop. de una secuencia en } \Sigma(\varphi)} p_i, \emptyset, \Sigma(\varphi)).$$

□

Vamos a continuar investigando la complejidad del problema de satisfactibilidad para $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$ en la siguiente sección.

6.3.2 *Complejidad del fragmento* $\mathcal{ML}([\![\ ]\!])$

Vamos a ver que el fragmento sin el operador de copia ($\mathcal{ML}([\![\ ]\!])$) puede ser traducido en la lógica modal básica con el operador de pasado $\Diamond^{-1}$ (denotada $\mathcal{ML}(\Diamond^{-1})$).

Sin pérdida de generalidad asumimos que el operador de borrado está parametrizado por expresiones de la forma

$$[\![\varphi_1?; a_1; \varphi_2?; a_2; \ldots; a_{n-1}; \varphi_n?]\!]\psi,$$

donde $\varphi_i?$ son fórmulas Booleanas arbitrarias, y $a_i \in \mathsf{AGT}$. Vamos a introducir axiomas de reducción para obtener una fórmula de $\mathcal{ML}(\Diamond^{-1})$. Las macros introducidas a continuación (del_i^π's) introducen operadores $\Diamond^{-1}$. Como vamos a usar del_i^π's en los axiomas de reducción, los pasos intermedios introducen $\Diamond^{-1}$. Por esta razón los axiomas de reducción son de fórmulas de $\mathcal{ML}([\![\ ]\!], \Diamond^{-1})$ a fórmulas de $\mathcal{ML}(\Diamond^{-1})$, con $\mathcal{ML}([\![\ ]\!], \Diamond^{-1})$ el lenguaje $\mathcal{ML}([\![\ ]\!])$ extendido con $\Diamond^{-1}$. Entonces concluiremos que a partir de cualquier fórmula de $\mathcal{ML}([\![\ ]\!])$,podemos obtener una equivalente en $\mathcal{ML}(\Diamond^{-1})$.

Primero, definamos las macros $\Diamond_{i,j}$ y $\Diamond_{i,j}^{-1}$, para un $\pi = \varphi_1?; a_1; \ldots; a_{n-1}; \varphi_n$ fijo.

$$\Diamond_{i,j} \quad = \quad \begin{cases} \top & j < i \\ \Diamond_{a_i}\varphi_{i+1} & i = j \\ \Diamond_{a_i}(\varphi_{i+1} \wedge \Diamond_{i+1,j}) & i < j \end{cases} \qquad \Diamond_{i,j}^{-1} \quad = \quad \begin{cases} \top & j < i \\ \Diamond_{a_i}^{-1}\varphi_i & i = j \\ \Diamond_{a_j}^{-1}(\Diamond_{i,j-1}^{-1} \wedge \varphi_j) & i < j \end{cases}$$

Definimos la fórmula del_i^π como sigue:

$$del_i^\pi = \Diamond_{1,i-1}^{-1} \wedge \varphi_i \wedge \Diamond_{i,n-1}.$$

Informalmente del_i^π significa "el estado actual está en la posición i en un camino que se corresponde con $\pi = \varphi_1?; a_1; \varphi_2?; a_2; \ldots; a_{n-1}; \varphi_n?$ y que va a ser eliminado".

Por ejemplo, las fórmulas $del_1^\pi, \ldots, del_n^\pi$ se definen como:

$$\begin{array}{lcl} del_1^\pi & = & \varphi_1 \wedge (\Diamond_{a_1}\varphi_2 \wedge (\Diamond_{a_2}\varphi_3 \ldots \wedge \Diamond_{a_{n-2}}(\varphi_{n-1} \wedge \Diamond_{a_{n-1}}\varphi_n)\ldots)) \\ del_2^\pi & = & \Diamond_{a_1}^{-1}\varphi_1 \wedge \varphi_2 \wedge (\Diamond_{a_2}\varphi_3 \ldots \wedge \Diamond_{a_{n-2}}(\varphi_{n-1} \wedge \Diamond_{a_{n-1}}\varphi_n)\ldots) \\ & \ldots & \\ del_{n-1}^\pi & = & \Diamond_{a_{n-2}}^{-1}(\Diamond_{a_{n-3}}^{-1}(\ldots(\Diamond_{a_1}^{-1}\varphi_1 \wedge \varphi_2) \wedge \varphi_3)\ldots) \wedge \varphi_{n-1} \wedge \Diamond_{a_{n-1}}\varphi_n \\ del_n^\pi & = & \Diamond_{a_{n-1}}^{-1}(\Diamond_{a_{n-2}}^{-1}(\ldots(\Diamond_{a_1}^{-1}\varphi_1 \wedge \varphi_2) \wedge \varphi_3\ldots) \wedge \varphi_{n-1}) \wedge \varphi_n \end{array}$$

El siguiente lema clarifica el significado de del_i:

Lema 6.3.4. *Sea $\mathcal{M} = \langle W, R, V\rangle$ un modelo, $w \in W$ y $\pi = \varphi_1?; a_1; \varphi_2?; \ldots; \varphi_n?$ una expresión de camino. Sea i tal que $0 \leq i \leq n$, entonces*

$$\mathcal{M}, w \models del_i^\pi \text{ sii existe } P \in \mathcal{P}_\pi^{\mathcal{M}} \text{ t.q. } P = w_1 a_1 w_2 \ldots w_n,\ w_i = w$$

y para todo $w_j \in P$ tenemos $\mathcal{M}, w_j \models \varphi_j$.

Demostración. La demostración es por inducción en la longitud de π:

- $\pi = \varphi_1?$: $\mathcal{M}, w \models del_1^\pi$ si y sólo si $\mathcal{M}, w \models \varphi_1$ (por definición de del_i^π). Pero $\mathcal{P}_{\varphi_1?}^{\mathcal{M}} = \{v \mid \mathcal{M}, v \models \varphi_1\}$ (todos los caminos son singletons que satisfacen φ_1), entonces $w \in \mathcal{P}_{\varphi_1?}^{\mathcal{M}}$.

- $\pi = \varphi_1?; a_1; \varphi_2?; \ldots; \varphi_n?$: Supongamos $\mathcal{M}, w \models del_i^\pi$. Por definición de del^π, tenemos $\mathcal{M}, w \models \Diamond_{1,i-1}^{-1} \wedge \varphi_i \wedge \Diamond_{i,n-1}$. Ahora sabemos:
  1. $\mathcal{M}, w \models \varphi_i$.
  2. $\mathcal{M}, w \models \Diamond_{1,i-1}^{-1}$, entonces por definición de $\Diamond_{i,j}^{-1}$ tenemos que $\mathcal{M}, w \models \Diamond_{a_{i-1}}^{-1}(\Diamond_{1,i-2}^{-1} \wedge \varphi_{i-1})$. Por definición de $\models$, existe $v \in W$ tal que $(v, w) \in R_{a_{i-1}}$ y $\mathcal{M}, v \models \Diamond_{1,i-2}^{-1} \wedge \varphi_{i-1}$. Definamos $\pi_1 = \varphi_1?; a_1; \varphi_2?; \ldots; \varphi_{i-1}?$. Entonces, por definición de del_i^π, tenemos $\mathcal{M}, v \models del_{i-1}^{\pi_1}$, y por H.I., existe un camino $P_1 \in \mathcal{P}_{\pi_1}^{\mathcal{M}}$ tal que $P_1 = w_1 a_1 \ldots w_{i-1}$, con $w_{i-1} = v$ y para todo $w_j \in P_1$, $\mathcal{M}, w_j \models \varphi_j$ $(0 \leq j \leq i-1)$.
  3. $\mathcal{M}, w \models \Diamond_{i,n-1}$, entonces por definición de $\Diamond_{i,j}$ tenemos $\mathcal{M}, w \models \Diamond_{a_i}(\varphi_{i+1} \wedge \Diamond_{i+1,n-1})$. Por definición de $\models$, existe $t \in W$ tal que $(w, t) \in R_{a_i}$ y $\mathcal{M}, t \models \varphi_{i+1} \wedge \Diamond_{i+1,n-1}$. Definamos $\pi_2 = \varphi_{i+1}?; a_{i+1}; \ldots; \varphi_n?$. Entonces, por definición de del_i^π, tenemos $\mathcal{M}, t \models del_{i+1}^{\pi_2}$, y por H.I., existe un camino $P_2 \in \mathcal{P}_{\pi_2}^{\mathcal{M}}$ tal que $P_2 = w_{i+1} a_{i+1} \ldots w_n$, con $w_{i+1} = t$ y para todo $w_j \in P_2$, $\mathcal{M}, w_j \models \varphi_j$ $(i+1 \leq j \leq n)$.

  Notar que $\pi = \pi_1; a_{i-1}; \varphi_i?; a_i; \pi_2$. Resta elegir $P = P_1 a_{i-1} w_i a_i P_2$ y tenemos lo que buscamos.

□

A continuación introducimos los axiomas de reducción que transforman fórmulas de $\mathcal{ML}([\![\ ]\!], \Diamond^{-1})$ en fórmulas de $\mathcal{ML}(\Diamond^{-1})$.

Definición 6.3.5. *Sea $\varphi = [\![\pi]\!]\theta$ una fórmula de $\mathcal{ML}([\![\ ]\!], \Diamond^{-1})$, con $\pi = \varphi_1?; a_1; \varphi_2?; \ldots; \varphi_n?$. Definimos la fórmula $\mathsf{Tr}(\varphi)$ como la fórmula de $\mathcal{ML}(\Diamond^{-1})$ que resulta de aplicar repetidamente los siguientes axiomas de reducción a la fórmula φ (asumimos que $\Diamond_a\psi$ está escrito como $\neg\Box_a\neg\psi$, y similarmente para $\Diamond^{-1}$).*

(1)	$[\![\pi]\!]p$	$\leftrightarrow$	$p,\ \ p \in \mathsf{PROP}$
(2)	$[\![\pi]\!]\neg\psi$	$\leftrightarrow$	$\neg[\![\pi]\!]\psi$
(3)	$[\![\pi]\!](\psi \wedge \psi')$	$\leftrightarrow$	$([\![\pi]\!]\psi \wedge [\![\pi]\!]\psi')$
(4)	$[\![\pi]\!]\Box_a\psi$	$\leftrightarrow$	$\Box_a[\![\pi]\!]\psi,\ \ \textit{if } a \notin \pi$
(5)	$[\![\pi]\!]\Box_a^{-1}\psi$	$\leftrightarrow$	$\Box_a^{-1}[\![\pi]\!]\psi,\ \ \textit{if } a \notin \pi$
(6)	$[\![\pi]\!]\Box_a\psi$	$\leftrightarrow$	$(\bigwedge_{i\in\{1,\ldots,n-1 \mid a_i=a\}} \neg del_i^\pi \rightarrow \Box_{a_i}[\![\pi]\!]\psi) \wedge$
			$(\bigwedge_{i\in\{1,\ldots,n-1 \mid a_i=a\}} (del_i^\pi \rightarrow \Box_{a_i}(del_{i+1}^\pi \vee [\![\pi]\!]\psi)))$
(7)	$[\![\pi]\!]\Box_a^{-1}\varphi$	$\leftrightarrow$	$(\bigwedge_{i\in\{1,\ldots,n-1 \mid a_i=a\}} \neg del_i^\pi \rightarrow \Box_{a_i}^{-1}[\![\pi]\!]\psi) \wedge$
			$(\bigwedge_{i\in\{1,\ldots,n-1 \mid a_i=a\}} (del_i^\pi \rightarrow \Box_{a_i}^{-1}(del_{i-1}^\pi \vee [\![\pi]\!]\psi))).$

Notar que la fórmula resultante solo contiene $\Box_a$ y $\Box_a^{-1}$, y no contiene $[\![\ ]\!]$. Vamos a demostrar que la reducción preserva equivalencia, discutiendo cada axioma. Primero mostramos que el axioma (1) preserva equivalencia.

Lema 6.3.6. *Sea $\mathcal{M} = \langle W, R, V\rangle$ un modelo, $w \in W$. Sea $p \in \mathsf{PROP}$ un símbolo proposicional y $\pi \in \mathsf{PATH}$ arbitraria, entonces*

$$\mathcal{M}, w \models [\![\pi]\!]p \text{ sii } \mathcal{M}, w \models p.$$

Demostración. Supongamos $\mathcal{M}, w \models [\![\pi]\!]p$. Por definición de $\models$, tenemos $\mathcal{M}_{[\![\pi]\!]}, w \models p$. Como $[\![\pi]\!]$ mantiene la misma valuación en el model variant, $w \in V(p)$. Entonces (por $\models$), $\mathcal{M}, w \models p$. □

Ahora veamos la distributividad de $[\![\ ]\!]$ con repecto a $\wedge$.

Lema 6.3.7. *Sea $\mathcal{M} = \langle W, R, V\rangle$ un modelo, ψ, ψ' dos fórmulas de $\mathcal{ML}([\![\ ]\!], \Diamond^{-1})$ y sea $\pi \in \mathsf{PATH}$ tal que $\pi = \varphi_1?; a_i; \varphi_2?; \ldots; a_{n-1}; \varphi_n?$. Entonces*

$$\mathcal{M}, w \models [\![\pi]\!](\psi \wedge \psi') \text{ sii } \mathcal{M}, w \models [\![\pi]\!]\psi \wedge [\![\pi]\!]\psi'.$$

Demostración. Supongamos $\mathcal{M}, w \models [\![\pi]\!](\psi \wedge \psi')$. Entonces, por definición de $\models$, $\mathcal{M}_{[\![\pi]\!]}, w \models (\psi \wedge \psi')$, lo cual implica que $\mathcal{M}_{[\![\pi]\!]}, w \models \psi$ y $\mathcal{M}_{[\![\pi]\!]}, w \models \psi'$. Aplicando otra vez la definición de $\models$, tenemos $\mathcal{M}, w \models [\![\pi]\!]\psi$ y $\mathcal{M}, w \models [\![\pi]\!]\psi'$, sii $\mathcal{M}, w \models [\![\pi]\!]\psi \wedge [\![\pi]\!]\psi'$. □

Vamos a demostrar que en ciertos casos, $[\![\ ]\!]$ y $\Box_a$ conmutan.

Lema 6.3.8. *Sea $\mathcal{M} = \langle W, R, V\rangle$ un modelo, ψ una fórmula de $\mathcal{ML}([\![\ ]\!], \Diamond^{-1})$ y sea $\pi \in \mathsf{PATH}$ tal que $\pi = \varphi_1?; a_i; \varphi_2?; \ldots; a_{n-1}; \varphi_n?$. Si $a_i \notin \pi$, entonces*

$$\mathcal{M}, w \models [\![\pi]\!]\Box_{a_i}\psi \text{ sii } \mathcal{M}, w \models \Box_{a_i}[\![\pi]\!]\psi.$$

Demostración. Supongamos $\mathcal{M}, w \models [\![\pi]\!]\Box_{a_i}\psi$. Por definición de $\models$ dos veces, tenemos que para todo v tal que $(w, v) \in (R_{[\![\pi]\!]})_{a_i}$, $\mathcal{M}_{[\![\pi]\!]}, v \models \psi$. Asumimos que $a_i \notin \pi$, entonces $(w, v) \in (R_{[\![\pi]\!]})_{a_i}$ sii $(w, v) \in R_{a_i}$, entonces tenemos que para todo v tal que $(w, v) \in R_{a_i}$, $\mathcal{M}_{[\![\pi]\!]}, v \models \psi$, sii para todo v tal que $(w, v) \in R_{a_i}$, $\mathcal{M}, v \models [\![\pi]\!]\psi$. Luego por $\models$, $\mathcal{M}, w \models \Box_{a_i}[\![\pi]\!]\psi$. □

Finalmente, vamos a demostrar la preservación de equivalencia del axioma (6). Una propiedad similar al Lema 6.3.8 vale también para $\Box_{a_i}^{-1}$.

Teorema 6.3.9. *Sea $\mathcal{M} = \langle W, R, V\rangle$ un modelo, $w \in W$, y sea $[\![\pi]\!]\Box_{a_i}\psi$ una fórmula de $\mathcal{ML}([\![\ ]\!], \Diamond^{-1})$ con $\pi = \varphi_1?; a_1; \varphi_2?; \ldots; \varphi_n?$, tal que $a_i \in \pi$. Entonces*

$$\mathcal{M}, w \models [\![\pi]\!]\Box_{a_i}\psi \text{ sii } \mathcal{M}, w \models \delta \wedge \delta'$$

donde

$$\begin{aligned} \delta &= \textstyle\bigwedge_{k \in \{1,\ldots,n-1 \mid a_k = a_i\}} \neg del_k^\pi \rightarrow \Box_{a_k}[\![\pi]\!]\psi \\ \delta' &= \textstyle\bigwedge_{k \in \{1,\ldots,n-1 \mid a_k = a_i\}} (del_k^\pi \rightarrow \Box_{a_k}(del_{k+1}^\pi \vee [\![\pi]\!]\psi)). \end{aligned}$$

Demostración. Supongamos que $\mathcal{M}, w \models [\![\pi]\!]\Box_{a_i}\psi$. Entonces, por definición de $\models$, tenemos que para todo $v \in W$ tal que $(w, v) \in (R_{[\![\pi]\!]})_{a_i}$, $\mathcal{M}_{[\![\pi]\!]}, v \models \psi$. Vamos a verificar δ y δ' de manera separada (para la otra dirección del sii, podemos asumir los dos conjuntos y usar los mismos pasos):
1. Supongamos $\mathcal{M}, w \models \bigwedge_{k\in\{1,\ldots,n-1 \mid a_k=a_i\}} \neg del_k^\pi$. Por definición de $\models$, tenemos $\mathcal{M}, w \not\models \bigvee_{k\in\{1,\ldots,n-1 \mid a_k=a_i\}} del_k^\pi$. Entonces no existe $P \in \mathcal{P}_\pi^{\mathcal{M}}$ que satisfaga el Lema 6.3.4, tal que $w \in P$, luego no se han realizado eliminaciones atravesando w. Entonces para todo $v \in W$, $(w, v) \in R_{a_i}$ sii $(w, v) \in (R_{[\![\pi]\!]})_{a_i}$. Como tenemos que para todo $v \in W$ tal que $(w, v) \in (R_{[\![\pi]\!]})_{a_i}$, $\mathcal{M}_{[\![\pi]\!]}, v \models \psi$, entonces para todo $v \in W$ tal que $(w, v) \in R_{a_i}$, $\mathcal{M}_{[\![\pi]\!]}, v \models \psi$. Luego, tenemos que para todo $v \in W$ tal que $(w, v) \in R_{a_i}$, $\mathcal{M}, v \models [\![\pi]\!]\psi$, entonces (por $\models$) $\mathcal{M}, w \models \Box_{a_i}[\![\pi]\!]\psi$.
2. Supongamos para algún k arbitrario, $\mathcal{M}, w \models del_k^\pi$, donde $k \in \{1, \ldots, n-1 \mid a_k = a_i\}$. Por Lema 6.3.4 existe un camino atravesando w que ha sido eliminado. También sabemos que $\mathcal{M}_{[\![\pi]\!]}, w \models \Box_{a_k}\psi$ por asunción y $k = i$, entonces para todo $v \in W$ tal que $(w, v) \in (R_{[\![\pi]\!]})_{a_k}$, $\mathcal{M}_{[\![\pi]\!]}, v \models \psi$. Entonces, para todo $u \in W$ tal que $(w, u) \in R_{a_k}$, sucede que $\mathcal{M}_{[\![\pi]\!]}, u \models \psi$ or $u \in P$, con $P \in \mathcal{P}_\pi^{\mathcal{M}}$, y u está en la posición $k+1$ (porque w está en la posición $k = i$), es decir, $\mathcal{M}, u \models del_{k+1}^\pi$ (por Lema 6.3.4). Luego, $\mathcal{M}, w \models \Box_{a_k}(del_{k+1}^\pi \vee [\![\pi]\!]\psi)$. □

El siguiente teorema establece que podemos reducir fórmulas de $\mathcal{ML}([\![\,]\!], \Diamond^{-1})$ de acuerdo a los axiomas de la Definición 6.3.5 obteniendo una fórmula de $\mathcal{ML}(\Diamond^{-1})$ equivalente.

Teorema 6.3.10. *Sea $\mathcal{M} = \langle W, R, V\rangle$ un modelo, $w \in W$ y φ una fórmula de $\mathcal{ML}([\![\,]\!], \Diamond^{-1})$. Entonces*

$$\mathcal{M}, w \models \varphi \textit{ sii } \mathcal{M}, w \models \mathsf{Tr}(\varphi).$$

Demostración. La demostración es unc corolario de los lemas 6.3.6 a 6.3.8, el Teorema 6.3.9, y las propiedades similares para $\Box_{a_i}^{-1}$. □

Hemos demostrado que los axiomas de reducción de la Definición 6.3.5 preservan equivalencia de fórmulas. Esto significa que podemos transformar fórmulas de $\mathcal{ML}([\![\,]\!])$ en fórmulas de $\mathcal{ML}(\Diamond^{-1})$. Como consecuencia, tenemos el siguiente teorema:

Teorema 6.3.11. *El problema de satisfactibilidad para $\mathcal{ML}([\![\,]\!])$ es decidible.*

Resumiendo, en este capítulo discutimos un lenguaje con operadores que cambian la accesibilidad que codifica una versión restringida de $\mathcal{AML}$. Dicho lenguaje incluye un operador que elimina aristas ($[\![\,]\!]$) y un operador que replica modelos (cp). La principal diferencia entre $[\![\,]\!]$ y los operadores sabotage introducidos anteriormente es que el nuevo operador depende de un parámetro que caracteriza caminos a borrar. Debido a que no permitimos caracterizaciones modales en las expresiones de camino, la codificación de $\mathcal{AML}$ en $\mathcal{ML}(\mathsf{cp}, [\![\,]\!])$ solo permite precondiciones Booleanas en los action models.

Investigamos varias propiedades del lenguaje $\mathcal{ML}(\mathsf{cp}, [\![\,]\!])$. Primero, mostramos que necesitamos las mismas condiciones que para $\Diamond^{-1}$ a la hora de dar una noción apropiada de bisimulación para el nuevo lenguaje. La razón es que necesitamos diferenciar estados de acuerdo a qué caminos los atraviesan, tanto hacia adelante

como hacia atrás. Demostramos el Teorema 6.1.9, que indica que modelos bisimilares satisfacen las mismas fórmulas en el lenguaje $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$. También investigamos propiedades computacionales, trabajando sobre dos fragmentos del lenguaje. Primero, introducimos un algoritmo PSPACE que verifica satisfactibilidad de fórmulas del fragmento $\mathcal{ML}(\mathsf{cp})$ (el lenguaje sin $[\![\ ]\!]$), concluyendo que su problema de satisfactibilidad es PSPACE-completo. Luego definimos una traducción que preserva equivalencias para el fragmento $\mathcal{ML}([\![\ ]\!])$ al lenguaje $\mathcal{ML}(\Diamond^{-1})$ por medio de axiomas de reducción, mostrando que su problema de satisfactibilidad es decidible. Dicha reducción puede generar posiblemente fórmulas de tamaño exponencial (con respecto a la fórmula de entrada), lo que no nos permitió dar cotas superiores aún.

Existen diferentes preguntas aún abiertas acerca de este lenguaje. Por ejemplo, obtuvimos resultados para diferentes fragmentos de $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$, pero sería interesante investigar el comportamiento computacional de los fragmentos combinados. Dado que pudimos codificar una versión restringida de $\mathcal{AML}$ en $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$, podemos concluirwe:

Teorema 6.3.12. *El problema de satisfactibilidad para $\mathcal{ML}(cp, [\![\ ]\!])$ es* NEXPTIME*-hard.*

Demostración. Codificamos una versión restringida de $\mathcal{AML}$ en $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$, y el problema de satisfactibilidad de $\mathcal{AML}$ es NEXPTIME-hard [Aucher and Schwarzentruber, 2013]. Esta demostración incluye el lenguaje completo, pero la codificación por medio de un problema de tiling solo utiliza precondiciones Booleanas. Luego, el problema de satisfactibilidad para $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$ es también NEXPTIME. □

El teorema anterior nos da una cota inferior para el problema de satisfactibilidad para $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$. La cota superior está abierta, pero conjeturamos que es decidible.

Otra dirección interesante de investigación sería definir procedimientos concretos para verificar satisfactibilidad (por ejemplo, tableaux), e investigar otras tareas de razonamiento como model checking, complejidad de fórmula y complejidad de programas. Además, no hemos discutido la relación entre $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$ y otras lógicas con operadores de cambio, tales como las introducidas anteriormente en esta tesis (sabotage, swap and bridge) y otros (arrow updates, graph modifiers, etc.). Finalmente, resultaría interesante extender la definición de expresiones de caminos que permitan eliminaciones anidadas y fórmulas modales, para codificar el lenguaje $\mathcal{AML}$ completo, con precondiciones modales, y estudiar su comportamiento computacional.

7

Conclusiones

El porvenir es tan irrevocable como el rígido ayer.
No hay una cosa que no sea una letra silenciosa
de la eterna escritura indescifrable cuyo libro es el tiempo.
Quien se aleja de su casa ya ha vuelto.
Nuestra vida es la senda futura y recorrida.
Nada nos dice adiós. Nada nos deja. No te rindas.
La ergástula es oscura, la firme trama es de incesante hierro,
pero en algún recodo de tu encierro puede haber un descuido, una hendidura.
El camino es fatal como la flecha. Pero en las grietas está Dios, que acecha.

from "Para una versión del I King", Jorge Luis Borges.

7.1 QUÉ HICIMOS?

La primer parte de esta tesis está dedicada a analizar las lógicas modales con operadores de cambio de accesibilidad desde un punto de vista abstracto. Investigamos diferentes aspectos lógicos de los lenguajes, sobre todo desde una perspectiva computacional. Elegimos un conjunto de primitivas como representantes de la familia de todos los posibles operadores de cambio de accesibilidad y estudiamos en detalle cada uno de ellos por separado. También estudiamos las relaciones entre ellos. Las principales contribuciones de nuestro trabajo han sido, por un lado, los resultados particulares de las seis lógicas que hemos introducido y, por otra parte, los resultados generales que se pueden aplicar a una familia más grande de lógicas. Hemos aprendido mucho sobre el comportamiento de estas seis lógicas con operadores de cambio de accesibilidad, y podemos utilizar esta experiencia para establecer resultados más generales.

En el capítulo 1 introdujimos la sintaxis y semántica formal de seis operadores de cambio de accesibilidad: sabotage, swap y bridge, cada uno de ellos en una versión local y global. Cada lenguaje es una extensión sintáctica de la lógica modal básica $\mathcal{ML}$ con un operador dinámico. La semántica es basada en modelos de Kripke y *model variants*, que son operaciones que modifican el modelo capturando el comportamiento de los nuevos operadores sintácticos. Introducimos model variants "ad hoc" para cada operador, pero está claro que sería fácil definir model variants basados en funciones de transformación genéricas . Observamos que, a pesar de que las condiciones semánticas parecen inocentes, los operadores capturan comportamiento complejo y esto resulta en un incremento del poder expresivo. De hecho se demostró que dos propiedades de teoría de modelos clásica para las lógicas modales se pierden: la tree y la finite model property. Para la tree model property, en la mayoría de los casos hemos definido fórmulas que fuerzan ciclos, y los operadores de bridge

fuerzan componentes desconectadas. Las fórmulas para forzar modelos infinitos son más complejas. Para lograr esto, utilizamos una herramienta clásica para demostrar resultados de expresividad en lógica: *la técnica de spy point*. La idea es caracterizar un estado que tiene una visión global del resto del modelo (el "spy point"), y lo utilizamos para describir las propiedades que queremos imponer en el modelo.

Hemos demostrado que las lógicas modales con operadores de cambio son más expresivas que $\mathcal{ML}$, demostrando la falta de la tree y la finite model property. En el capítulo 2 obtuvimos más resultados expresividad . La principal herramienta que se usa en lógica modal para investigar el poder expresivo es *bisimulación*. Las bisimulaciones relacionan estados de los modelos en función de su comportamiento. Si dos estados en dos modelos están unidos por un bisimulación, entonces ellos deben satisfacer las mismas fórmulas. Esto se llama *invarianza por bisimulación*. En [Areces *et al.*, 2012; Areces *et al.*, 2013b] introdujimos la definición de bisimulación adecuada para cada operador que captura exactamente su significado. Hemos demostrado en [Areces *et al.*, 2013b] que $\mathcal{ML}(\langle \mathsf{sw} \rangle)$ es un fragmento propio de $\mathcal{FOL}$ y conjeturamos de que los mismos argumentos se pueden aplicar para las otras cinco lógicas. Sin embargo, todos los fragmentos se capturan de diferente manera: todos ellos son incomparables con respecto a su poder expresivo, excepto en el caso de los dos operadores de swap. Sabemos que $\mathcal{ML}(\langle \mathsf{gsw} \rangle) \not\leq \mathcal{ML}(\langle \mathsf{sw} \rangle)$, pero la otra dirección está todavía abierta. Sin embargo, conjeturamos que también son incomparables.

Otro reto ha sido el de establecer límites computacionales. Exploramos en el Capítulo 3 el problema de satisfactibilidad de estas lógicas. Atacamos el problema de dos formas diferentes: mediante la codificación del problema de tiling de $\mathbb{N}\times\mathbb{N}$ (el cual es indecidible) y codificando el problema de satisfactibilidad indecidible para la lógica con memoria $\mathcal{ML}(ⓡ, ⓚ)$. El segundo enfoque dio mejores resultados, y fue aplicado para la versión local de las lógicas. Con el fin de codificar $\mathcal{ML}(ⓡ, ⓚ)$, utilizamos una vez más una técnica de spy point para simular la capacidad de memorizar elementos, sin una memoria. Este es uno de los resultados más difíciles en esta tesis: es necesario forzar algunas restricciones en la forma de los modelos, y una maquinaria diferente para simular los operadores de memorización con cada uno de los operadores de cambio de accesibilidad. Conjeturamos que los mismos argumentos pueden ser aplicados para las versiones globales de los operadores de cambio de accesibilidad.

El problema satisfactibilidad no es la única tarea de razonamiento que hemos investigado. En el capítulo 4 demostramos resultados de complejidad para el problema de model checking. Redujimos el problema de satisfactibilidad para Quantified Boolean Fórmulas (QBF), el cual es PSPACE-completo, al problema de model checking para las seis lógicas investigadas. Esto demuestra una cota inferior PSPACE para los seis problemas de model checking. La codificación es, en general, bastante uniforme, y requiere pequeñas modificaciones para adaptarse a cada operador. Simulamos asignaciones de verdad a variables de WBF con la posición de las aristas de un modelo de Kripke , y cambios en sus valores de verdad son simuladas por los cambios en la relación de accesibilidad. También consideramos la tarea de model checking con un modelo fijo, midiendo su complejidad como una función en el tamaño de una fórmula de entrada (complejidad de fórmula), y fijando una fórmula y midiendo la complejidad en función del tamaño del modelo de entrada (complejidad de programa o de datos). Hemos demostrado que la complejidad de fórmula de las seis lógicas

es lineal con respecto al tamaño de la fórmula, y la complejidad del programa es polinomial con respecto al tamaño del modelo finito. Curiosamente, hemos sido capaces de probar estos resultados para una gran familia de lógicas con operadores de cambio. Definimos operadores de cambio en términos de funciones de transformación. Para cualquier lógica, si tenemos que las funciones de transformación asociadas con sus operadores producen sólo una cantidad polinomial de modelos en cada paso, los resultados de complejidad de fórmula y de programa aplican.

Claramente, incluyendo operadores de cambio de accesibilidad (por lo menos los seis que investigados en esta tesis) algunas tareas de razonamiento se vuelven intratables. Sin embargo, es posible definir procedimientos concretos cuya terminación no está garantizada. En el capítulo 5 definimos métodos de tableau para comprobar satisfactibilidad de lógicas modales con operadores de cambio de accesibilidad. Este trabajo fue publicado por primera vez en [Areces *et al.*, 2013c]. Estos métodos son completos y correctos, pero pueden no terminar. Las fórmulas de tableaux en los cálculos que hemos definido, contienen prefijos, que son muy diferentes a los prefijos en otros métodos de tableau (por ejemplo, para $\mathcal{ML}$, $\mathcal{HL}(@,\downarrow)$, etc.) En nuestro tableaux, los prefijos proporcionan la información sobre el punto de evaluación, así como el model variant en la que se evalúa la fórmula. Por esta razón hemos tenido que definir diferentes reglas para cada lógica (excepto para la versión local y global del mismo modificador). Por ejemplo, prefijos de sabotage mantienen un registro de las aristas eliminadas. Como ha sido el caso de varios resultados, los operadores de swap han sido los más difíciles de investigar. Debemos tener en cuenta algunos casos especiales, como cuando las arista cambiada es un bucle (no provoca cambio) o cuando la arista cambiada se ha intercambiado antes. Las reglas para cada lógica son muy diferentes, y parece difícil encontrar un marco uniforme que permita la combinación de reglas para definir un cálculo de tableau para una lógica con más de un operador de cambio de accesibilidad (sólo podemos combinar la versión local y global del mismo operador).

Nuestros resultados nos dieron experiencia investigando operadores que pueden cambiar la relación de accesibilidad de un modelo. Otros operadores que caen en esta clase pero que han sido diseñado con ciertos propósitos han sido investigados en trabajos previos. Un ejemplo puede ser el de operadores de cambio de información en *Lógicas Dinámicas Epistémicas* ($\mathcal{DEL}$). Resulta natural estudiar conexiones entre estos operadors y aquellos que hemos definido. La primer pregunta que surgió fue: *"Es posible codificar lógica dinámica epistémica con alguna variante de las seis lógicas que introdujimos?"*. En $\mathcal{DEL}$ los modelos representan información y conocimento de agentes, y los operadores dinámicos son usados para modelo comunicación, es decir, cambios de información. En el Capítulo 6 definimos un lenguaje que es más expresivo que $\mathcal{ML}$ pero con un mejor comportamiento computacional que las seis lógicas que investigamos en los primeros capítulos.

7.2 MIRANDO HACIA EL FUTURO

Como ya señalamos en los distintos capítulos, varias preguntas siguen abiertas. Esta tesis aborda muchas preguntas sobre lógicas con operadores de cambio. Pero, por

supuesto, el tiempo es limitado y hay algunas direcciones interesantes a seguir en el futuro.

Conjeturamos, por ejemplo, que $\mathcal{ML}(\langle \mathsf{sw} \rangle)$ y $\mathcal{ML}(\langle \mathsf{gsw} \rangle)$ son incomparables con respecto a su poder expresivo, y que la lógica $\mathcal{ML}(\mathsf{cp}, [\![\ ]\!])$ es decidible. Además, se mencionó que la prueba de la indecidibilidad del problema satisfactibilidad para los operadores locales se pueden adaptar para las globales. Sería interesante finalizar estos resultados para tener una imagen completa de los resultados presentados en esta tesis.

En cuanto a poder expresivo, nos gustaría definir una noción más general de bisimulación que en lugar de considerar los casos particulares de las actualizaciones (borrar, intercambiar o agregar aristas) podría considerar las funciones de actualización en general (similar a los que utilizamos para nuestros resultados de model checking). Las mismas ideas también pueden ser aplicadas, por ejemplo, a la Lógicas Dinámicas Epistémicas . Para $\mathcal{PAL}$ y $\mathcal{AML}$, basta con la noción de bisimulación de la lógica modal básica $\mathcal{ML}$. Si extendemos el lenguaje con operadores más expresivos, sería posible introducir condiciones de actualización en de la misma manera que hicimos para las lógicas modales con oepradors de cambio. También mencionamos en el capítulo 2 que sería interesante investigar *Caracterización van Benthem* para las lógicas modales con operadores de cambio de accesibilidad. Podemos comenzar por comprobar si los resultados introducidos en [Areces *et al.*, 2013a] pueden ser usados para las lógicas modales que introdujimos en esta tesis. Otra propiedad interesante que empezamos a investigar es el Lema de Interpolación de Craig. Se estudió el método constructivo provisto en [Fitting, 1996; Fitting, 2002; Blackburn and Marx, 2003] siguiendo las mismas ideas en las lógicas modales operadores de cambio, pero esto queda como trabajo como futuro.

Bibliography

[Areces and Gorín, 2010] C. Areces and D. Gorín. Coinductive models and normal forms for modal logics (or how we learned to stop worrying and love coinduction). *Journal of Applied Logic*, 8(4):305–318, 2010. Citado en página 36.

[Areces *et al.*, 2009] C. Areces, D. Figueira, D. Gorín, and S. Mera. Tableaux and model checking for memory logics. In *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 5607 of *LNAI*, pages 47–61, Oslo, Norway, 2009. Springer Berling / Heidelberg. Proceedings of Tableaux09. Citado en página 51.

[Areces *et al.*, 2012] C. Areces, R. Fervari, and G. Hoffmann. Moving arrows and four model checking results. In L. Ong and R. Queiroz, editors, *Logic, Language, Information and Computation*, volume 7456 of *Lecture Notes in Computer Science*, pages 142–153. Springer Berlin Heidelberg, 2012. Citado en páginas 5, 12, 15, 31, 36 y 68.

[Areces *et al.*, 2013a] C. Areces, F. Carreiro, and S. Figueira. Characterization, definability and separation via saturated models. *Theoretical Computer Science*, 2013. Citado en páginas 17 y 70.

[Areces *et al.*, 2013b] C. Areces, R. Fervari, and G. Hoffmann. Swap logic. *Logic Journal of IGPL*, 2013. Citado en páginas 5, 12, 19, 31, 36 y 68.

[Areces *et al.*, 2013c] C. Areces, R. Fervari, and G. Hoffmann. Tableaux for relation-changing modal logics. In P. Fontaine, C. Ringeissen, and R. Schmidt, editors, *Frontiers of Combining Systems*, volume 8152 of *Lecture Notes in Computer Science*, pages 263–278. Springer, 2013. Citado en páginas 36, 41 y 69.

[Aucher and Schwarzentruber, 2013] G. Aucher and F. Schwarzentruber. On the complexity of dynamic epistemic logic. In *Proceedings of TARK 2013*, Chennai, India, January 2013. Citado en páginas 53 y 66.

[Aucher *et al.*, 2009] G. Aucher, P. Balbiani, L. Fariñas Del Cerro, and A. Herzig. Global and local graph modifiers. *Electronic Notes in Theoretical Computer Science (ENTCS), Special issue Proceedings of the 5th Workshop on Methods for Modalities (M4M5 2007)*, 231:293–307, 2009. Citado en página 53.

[Beth, 1955] E. W. Beth. Semantic entailment and formal derivability. *Mededellingen van de Koninklijke Nederlandse Akademie van Wetenschappen Afdeling Letterkunde N. R.*, 18(13):309–342, 1955. Citado en página 41.

[Blackburn and Marx, 2003] P. Blackburn and M. Marx. Constructive interpolation in hybrid logic. *Journal of Symbolic Logic*, 68(2):463–480, 2003. Citado en páginas 52 y 70.

[Blackburn and van Benthem, 2006] P. Blackburn and J. van Benthem. Modal logic: A semantic perspective. In *Handbook of Modal Logic*. Elsevier North-Holland, 2006. Citado en página 5.

[Blackburn *et al.*, 2001] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 2001. Citado en páginas 6, 56 y 57.

[Craig, 1957] W. Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22:269–285, 1957. Citado en página 52.

[D'Agostino *et al.*, 1999] M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga. *Handbook of tableau methods*. Kluwer Academic Publishers, 1999. Citado en página 41.

[Dummet and Lemmon, 1959] M. Dummet and E. Lemmon. Modal logics between s4 and s5. *Zeitschrift für mathemathische Logik und Grundla-gen der Mathematik*, 5:250–264, 1959. Citado en página 5.

[Fitting, 1972] M. Fitting. Tableau methods of proof for modal logics. *Notre Dame Journal of Formal Logic*, 13(2):237–247, 1972. Citado en página 41.

[Fitting, 1996] M. Fitting. *First-order logic and automated reasoning (2. ed.)*. Graduate texts in computer science. Springer, 1996. Citado en páginas 52 y 70.

[Fitting, 2002] M. Fitting. Interpolation for first order s5. *Journal of Symbolic Logic*, 67(2):621–634, 2002. Citado en páginas 52 y 70.

[French *et al.*, 2013] T. French, W. van der Hoek, P. Iliev, and B. Kooi. On the succinctness of some modal logics. *Artificial Intelligence*, 197:56–85, 2013. Citado en página 53.

[Goré, 1999] R. Goré. Tableau methods for modal and temporal logics. *Handbook of tableau methods*, pages 297–396, 1999. Citado en página 61.

[Hintikka, 1962] J. Hintikka. *Knowledge and Belief. An Introduction to the Logic of the Two Notions*. Cornell University Press, Ithaca, NY, 1962. Citado en página 53.

[Kooi and Renne, 2011a] B. Kooi and B. Renne. Arrow update logic. *Review of Symbolic Logic*, 4(4):536–559, 2011. Citado en página 53.

[Kooi and Renne, 2011b] B. Kooi and Bryan Renne. Generalized arrow update logic. In K. Apt, editor, *TARK*, pages 205–211. ACM, 2011. Citado en página 53.

[Kripke, 1963] S. Kripke. Semantical analysis of modal logic I. Normal propositional calculi. *Zeitschrift fur mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963. Citado en página 53.

[Kupferman *et al.*, 2000] O. Kupferman, M. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000. Citado en página 35.

[Löding and Rohde, 2003a] C. Löding and P. Rohde. Model checking and satisfiability for sabotage modal logic. In P. Pandya and J. Radhakrishnan, editors, *Proceedings of Foundations of Software Technology and Theoretical Computer Science, 23rd Conference*, volume 2914 of *Lecture Notes in Computer Science*, pages 302–313. Springer, 2003. Citado en páginas 5, 31 y 35.

[Löding and Rohde, 2003b] C. Löding and P. Rohde. Solving the sabotage game is PSPACE-hard. In *Mathematical Foundations of Computer Science 2003*, volume 2747 of *Lecture Notes in Computer Science*, pages 531–540. Springer, Berlin, 2003. Citado en página 31.

[Lutz, 2006] C. Lutz. Complexity and succinctness of public announcement logic. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *AAMAS*, pages 137–143, Hakodate, Japan, 2006. ACM. Citado en página 53.

[Papadimitriou, 1994] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994. Citado en página 31.

[Plaza, 2007] J. Plaza. Logics of public communications. *Synthese*, 158(2):165–179, 2007. Citado en página 53.

[Rohde, 2006] P. Rohde. *On games and logics over dynamically changing structures*. PhD thesis, RWTH Aachen, 2006. Citado en páginas 35 y 36.

[Sahlqvist, 1973] H. Sahlqvist. Completeness and correspondence in the first and second order semantics for modal logic. In S. Kanger, editor, *Third Scandinavian Logic Symposium*, pages 110–143, Uppsala, 1973. North-Holland Publishing Company 1975. Citado en página 5.

[Schmidt and Tishkovsky, 2007] R. A. Schmidt and D. Tishkovsky. Using tableau to decide expressive description logics with role negation. In *ISWC/ASWC*, volume 4825 of *Lecture Notes in Computer Science*, pages 438–451. Springer, 2007. Citado en página 43.

[Schnoebelen, 2002] P. Schnoebelen. The complexity of temporal logic model checking. In P. Balbiani, N. Suzuki, F. Wolter, and M. Zakharyaschev, editors, *Advances in Modal Logic 4*, pages 393–436. King's College Publications, 2002. Citado en página 35.

[Smullyan, 1968] R. Smullyan. *First-Order Logic*. Springer-Verlag, 1968. Citado en página 41.

[van Benthem, 1977] J. van Benthem. *Modal Correspondence Theory*. PhD thesis, University of Amsterdam, Amsterdam, The Netherlands, 1977. Citado en página 17.

[van Benthem, 1984] J. van Benthem. Modal correspondence theory. *Handbook of Philosophical Logic*, 2:167–247, 1984. Citado en página 11.

[van Benthem, 1985] J. van Benthem. *Modal logic and classical logic*. Bibliopolis, 1985. Citado en página 11.

[van Benthem, 2005] J. van Benthem. An essay on sabotage and obstruction. In D. Hutter and W. Stephan, editors, *Mechanizing Mathematical Reasoning*, volume 2605 of *Lecture Notes in Computer Science*, pages 268–276. Springer, 2005. Citado en página 2.

[van Ditmarsch *et al.*, 2007] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Kluwer, 2007. Citado en páginas 35 y 53.

[Vardi and Wolper, 1986] M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *LICS*, pages 332–344. IEEE Computer Society, 1986. Citado en página 35.

[Vardi, 1982] M. Vardi. The complexity of relational query languages. In H. Lewis, B. Simons, W. Burkhard, and L. Landweber, editors, *Symposium on Theory of Computing*, pages 137–146. ACM, 1982. Citado en página 35.

[von Wright, 1951] G. H. von Wright. *An Essay in Modal Logic*. Amsterdam, North-Holland Pub. Co., 1951. Citado en página 53.