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

Fragmentos de CPDL$^+$ mediante propiedades de pathwidth

Tesis de Licenciatura en Ciencias de la Computación

Mauro Luis Schiavinato

Director: Santiago Figueira

Codirector: Edwin Pin Baque

Buenos Aires, 2024

Fragmentos de CPDL$^+$ mediante propiedades de pathwidth

En este trabajo nos centramos en CPDL$^+$, un lenguaje recientemente definido que extiende a PDL, una lógica modal ya conocida y ampliamente estudiada. Sobre esta nueva lógica estudiamos una jerarquización del lenguaje basado en propiedades de pathwidth asociados a los grafos subyacentes de ciertas expresiones que este lenguaje posee. Esta jerarquización es comparable y análoga a la jerarquización de CPDL$^+$ basada en propiedades de treewidth con la que se definió originalmente.

En un principio, nuestro problema se basaba en hacer una comparación de expresividad de varios fragmentos de CPDL$^+$, pero varias propiedades relacionadas con teoría de modelos y juegos de simulación fueron surgiendo a medida que este estudio se fue desarrollando. Finalmente, demostramos que el criterio de simulación estudiado para la jerarquía de treewidth también sirve para pathwidth, con algunas modificaciones semánticas.

Palabras claves: Lógica Modal, PDL, Grafos, Pathwidth, Simulación.

Índice general

Introducción II

1. Preliminares 1
1.1. Aspectos básicos de teoría de grafos . . . 1
1.2. Lógicas modales clásicas . . . 4
1.3. Propositional Dynamic Logic . . . 6
1.3.1. Extensiones de PDL . . . 8
1.4. PDL con esteroides . . . 8
1.4.1. Simulación . . . 11
1.4.2. Expresividad . . . 12

2. Estudio jerárquico de pathwidth 16
2.1. Resultados en los primeros niveles vía reducciones . . . 16
2.2. Algunas propiedades de modelos y separabilidad . . . 19
2.2.1. ¿Vale la Pathwidth Model property? . . . 21
2.3. Equivalencia de modelos y simulación . . . 22
2.3.1. Simulación restringida a modelos determinísticos . . . 22
2.3.2. Criterio de simulación para la jerarquía pathwidth . . . 26

Conclusiones 37

Introducción

La computación teórica es una disciplina fascinante que se centra en la comprensión de los fundamentos matemáticos de la computación. Se trata de un campo con una rica historia que se remonta a la década de 1930, mucho antes de la existencia de las computadoras modernas. Entre los primeros trabajos de esta índole se encuentran las tesis de Alonzo Church y Alan Turing, que propusieron de manera independiente definiciones formales (y equivalentes) de *funciones computables*, iniciando así toda un área de nuevas ideas por explorar que se entrelazaría con otras disciplinas ya existentes.

Se puede decir que en el corazón de la computación teórica se encuentra la *lógica*, que es el estudio de la inferencia sintáctica o semántica, a partir de sistemas formales. Esta rama tiene estrechas relaciones con muchos temas de la computación, ya sea analizando de manera algorítmica ciertos problemas inherentes a una lógica (por ejemplo, estudiando la complejidad de problemas de decisión como satisfacibilidad, model checking, equivalencia de fórmulas, equivalencia de modelos, inferencia, etc), o desde una perspectiva más profunda como sucede con la Computabilidad Descriptiva donde se intenta caracterizar clases de complejidad mediante lenguajes lógicos formales.

En este trabajo nos vamos a enfocar en unos tipos de lógicas muy particulares denominadas *lógicas modales*, las cuales se definen mediante operadores de modalidad que en la práctica se utilizan para entender distintas nociones como necesidad, posibilidad, obligación, permiso, creencia, conocimiento, y otras similares. A estas lógicas se les puede encontrar aplicaciones en una amplia gama de áreas, incluyendo la filosofía, la lingüística, la matemática, la inteligencia artificial y las ciencias de la computación, aunque en este trabajo nos limitamos a estudiar estas lógicas de manera abstracta y general. Comúnmente se toma a la lógica proposicional como punto de partida para definir sobre esta operadores de modalidad. En este sentido, entenderemos una modalidad como un operador $\Diamond$, denominado *diamante*, que actúa sobre fórmulas φ, dando pie (de manera iterativa) a nuevas expresiones de la forma $\Diamond\varphi$ y cuyo significado general se entiende como *desde el mundo observado podemos acceder a otro con la propiedad* φ. Como veremos más adelante, esta noción es fácilmente adaptable y compatible con otros criterios de accesibilidad, dando lugar a lógicas bastante expresivas. A lo largo del trabajo, nos basaremos en el enfoque relacional de la lógica modal y trabajaremos exclusivamente con *Estructuras de Kripke* para la semántica, que son un tipo de estructuras de primer orden que restringidas a relaciones unarias y binarias, se pueden entender como grafos dirigidos con etiquetas en flechas y nodos, también llamados *mundos* bajo este contexto. La visualización de las estructuras de Kripke como grafos hace que la noción de accesibilidad para lidiar con los operadores modales resulte relevante.

En 1979, Fischer y Ladner introdujeron un tipo de lógica modal llamada *Propositional Dynamic Logic* [Fischer and Ladner, 1979], PDL en adelante, la cual surgió para describir correctitud, terminación y equivalencia de programas, a través de operadores de modalidad descritos por medio de operaciones de lenguajes regulares. Más recientemente se ha aplicado con otros propósitos, por ejemplo, como lenguaje de consulta sobre estructuras con forma de grafos con valores de datos en ejes y nodos [Lange, 2006].

Nuestro punto de partida son unas extensiones de PDL llamadas CPDL (converse + PDL), e ICPDL (intersection + CPDL) estudiadas en [Streett, 1982, Göller et al., 2009], donde demuestran que estos lenguajes poseen varias propiedades computacionales buenas, en particular, prueban que ambos lenguajes son decidibles. Más relevante para esta tesis, tenemos que el problema de decidir si un par de estructuras de Kripke finitas son equivalente bajo lenguajes como PDL y CPDL es también decidible mediante un algoritmo polinomial, como se demuestra en [Balcázar et al., 1992] a partir de un criterio de *bisimulación*. Un juego de bisimulación se puede entender como una instancia particular de los *juegos de Ehrenfeucht–Fraïssé* en el que un par de jugadores, Spoiler y Duplicador, intentan demostrar que dos modelos son o no equivalentes. Si bien ya existían varios trabajos que abordaban el estudio de PDL a través de propiedades de bisimulación como [Benevides, 2014, Carreiro, 2015], no existían trabajos similares para ICPDL, hasta muy recientemente.

Motivados por la problemática anterior, en [Figueira et al., 2023] se define una nueva extensión de CPDL mediante un operador de programas denominado *programa conjuntivo* y que depende subyacentemente de propiedades de grafos finitos. De esta manera, queda definido CPDL^+ como una jerarquización de lenguajes dependientes de ciertas clases de grafos. Más específicamente, CPDL^+ se puede ver como una unión infinita de lenguajes $\mathrm{CPDL}^+(\mathcal{G})$ donde $\mathcal{G}$ denota una clase de grafos particulares. En específico, en [Figueira et al., 2023] se usaron familias $\mathcal{G}$ que constan de grafos con *treewidth acotado*. Entre varios de los resultados obtenidos allí, destacamos los siguientes:

- El fragmento $\mathrm{CPDL}^+(\mathrm{TW}_k)$, donde TW_k consta de los grafos finitos y conexos que tienen treewidth menor o igual a k, está caracterizado por un juego de bisimulación, para todo $k \geq 2$.
- ICPDL es lógicamente equivalente al fragmento de $\mathrm{CPDL}^+(\mathrm{TW}_2)$.

En esta tesis se propone estudiar una jerarquización distinta de CPDL^+ que depende de otro parámetro intrínsecamente relacionado al treewidth de un grafo, denominado *pathwidth*. El treewidth y el pathwidth son variables numéricas definidas sobre grafos, comúnmente usadas en *análisis de complejidad parametrizada*. Muchos algoritmos que son NP-hard para grafos en general, se vuelven fáciles de ejecutar (o al menos tratables) cuando el treewidth o el pathwidth de los grafos input está acotado por una constante fija. Intuitivamente, el treewidth indica qué tan distante está un grafo de ser un árbol mientras que el pathwidth nos dice qué tan distante está de ser un camino. La estrategia que seguiremos para estudiar los lenguajes $\mathrm{CPDL}^+(\mathrm{PW}_k)$, con PW_k la clase de los grafos finitos y conexos con pathwidth menor o igual a k, será esencialmente la misma que la del artículo [Figueira et al., 2023], pero haremos especial énfasis en los puntos en los que ambos análisis se diferencian.

¿Por qué se estudia pathwidth en la literatura? La *Complejidad Parametrizada* es una subrama de la Teoría de Complejidad Computacional que se enfoca en la clasificación de

problemas computacionales de acuerdo a su dificultad inherente con respecto a múltiples parámetros. Por ejemplo, algunos problemas pueden resolverse con algoritmos que son exponenciales con respecto a un parámetro fijo pero polinomiales en el tamaño de la entrada. Tales algoritmos se denominan *fixed-parameter tractable algorithms*, o simplemente fpt-algoritmos [Downey and Fellows, 2002]. El treewidth establece un criterio clásico para el análisis de fpt-algoritmos que permite en muchos casos simplificar la complejidad de un problema considerando el treewidth de la entrada. En esta tesis no nos vamos a concentrar en temas de complejidad, pero conocer el trasfondo que tienen ambas propiedades puede ser beneficioso para desarrollar futuras investigaciones.

En el capítulo inicial daremos algunos conceptos preliminares necesarios para entender con mayor profundidad este tipos de lógicas. Partiremos con algunos aspectos básicos de teoría de grafos que serán relevantes a lo largo del trabajo y que conectaremos a su debido momento con la presentación de los lenguajes lógicos de interés. Daremos también algo de desarrollo teórico de las lógicas PDL, CPDL, ICPDL y finalmente CPDL^+ y algunos de sus fragmentos, como fueron planteados en [Figueira et al., 2023].

En el siguiente capítulo expondremos formalmente los problemas particulares a tratar y demostraremos varias propiedades sobre el lenguaje $\mathrm{CPDL}^+(\mathrm{PW}_k)$, definido previamente en el primer capítulo.

Concluimos el trabajo con una serie de posibles directrices para abordar los problemas que quedaron abiertos y algunos otros que podrían extender esta investigación.

Capítulo 1

Preliminares

Para hablar de CPDL$^+$ es necesario contar primero con algunos conceptos de teoría de grafos y de lógicas modales que daremos en este capítulo. Además, aprovechamos las próximas secciones para establecer aspectos de nomenclatura y notación. Los conceptos que veremos a continuación, si bien se consiguen en la bibliografía citada y muchos son de común conocimiento, se colocarán para hacer el trabajo lo más autocontenido posible.

1.1. Aspectos básicos de teoría de grafos

Un grafo (simple, no dirigido) es una tupla $G = (V, E)$ donde V es un conjunto no vacío de vértices y E es un conjunto de ejes (también llamadas aristas), donde cada eje es un conjunto no vacío de vértices en V de tamaño a lo sumo dos. Nos referimos a V y E como $V(G)$ y $E(G)$, respectivamente. Decimos que un grafo G es finito si $V(G)$ es un conjunto finito. En general, asumimos que los grafos son finitos a menos que se indique lo contrario.

Dados un par de grafos G, H, decimos que H es subgrafo de G si $V(H) \subseteq V(G)$ y $E(H) \subseteq E(G)$. Dado $X \subseteq V(G)$ no vacío, el subgrafo inducido por X es el grafo H tal que $V(H) = X$ y $E(H)$ consta de todas las aristas $\{u, v\} \in E(G)$ con $u, v \in X$, o en otras palabras, H es el subgrafo maximal de G con conjunto de vértices X. A veces nos referiremos al grafo inducido o generado por una tupla $\bar{u}$ de vértices, entendiéndose que nos referimos al subgrafo inducido por las entradas de la tupla.

Un grafo H se denomina menor del grafo G si se puede formar H a partir de G aplicando sucesivamente operaciones de eliminación de aristas, eliminación de vértices y contracción de aristas. Una familia $\mathcal{F}$ de grafos se dice que es cerrada por menores si los menores de cualquier $G \in \mathcal{F}$ también están en $\mathcal{F}$. Un grafo G es un menor prohibido minimal de una familia de grafos $\mathcal{F}$ cerrada por menores si $G \notin \mathcal{F}$ pero cualquier menor propio de G está en $\mathcal{F}$.

Un camino en un grafo G es una secuencia $v_1, v_2, \ldots, v_n$ de vértices de G tal que para todo $i = 1, \ldots, n-1$, se tiene que $\{v_i, v_{i+1}\} \in E(G)$. Tal camino se denomina ciclo si además se cumple que $v_1 = v_n$. Notemos que todo camino se puede ver como un subgrafo.

Definimos la longitud del camino $v_1, v_2, \ldots, v_n$ como n. La distancia entre dos vértices es la menor longitud de los caminos entre esos vértices, o indefinido si no existe tal camino.

Un grafo G es conexo si para todo par $u, v \in V(G)$ con $u \neq v$, existe un camino $v_1, \ldots, v_n$ en G tal que $v_1 = u$ y $v_n = v$. Una componente conexa de G es un subgrafo conexo maximal.

Los siguientes son algunas tipos de grafos de interés:

- Un árbol es un grafo conexo sin ciclos (en particular, sin ejes de la forma $\{v\}$).
- Un árbol con raíz es un árbol con un vértice distinguido r denominado raíz.
- Un árbol se dice caterpillar si todos los vértices están a distancia a lo sumo 1 de un camino central.
- Un k-clique es un grafo completo de k vértices.

Un digrafo es un grafo G donde los ejes ahora tienen dirección, es decir, los elementos de $E(G)$ son conjuntos no vacíos de tamaño a lo sumo 2 y *totalmente ordenados*. Como es usual, en un digrafo denotamos a los ejes como pares ordenados (u, v).

Todas las definiciones vistas sobre grafos, aplican también sobre digrafos, salvo la de caminos que requieren un poco más de especificación.

En un digrafo G, un camino dirigido $v_1, v_2, \ldots, v_n$ es una secuencia de vértices de G tal que $(v_i, v_{i+1}) \in E(G)$. Por otra parte, decimos que $v_1, v_2, \ldots, v_n$ es un camino no dirigido, o simplemente camino, si para todo $i = 1, \ldots, n-1$, $(v_i, v_{i+1}) \in E(G)$ o $(v_{i+1}, v_i) \in E(G)$, es decir, en un camino no dirigido podemos recorrer ejes en el sentido contrario a su dirección. Durante este trabajo cuando decimos que un digrafo es conexo, es respecto a caminos no dirigidos.

Todo digrafo G tiene un grafo subyacente asociado, el cual se obtiene quitando la dirección de los ejes en $E(G)$. De esta correlación, es claro que un digrafo es conexo si y solo si su grafo subyacente asociado es conexo.

Una descomposición de árbol de un grafo G es un par $(T, \mathbf{v})$ donde T es un árbol y $\mathbf{v} : V(T) \to \wp(V(G))$ es una función que asocia cada vértice de T, llamado bolsa, con un conjunto de vértices de G. Cuando $x \in \mathbf{v}(b)$ diremos que la bolsa $b \in V(T)$ contiene al vértice x. Además, se deben cumplir las siguientes propiedades:

Td1 cada vértice x de G está contenido en al menos una bolsa de T;

Td2 para cada eje de G entre x e y, hay al menos una bolsa de T que contiene tanto a x como a y; y

Td3 para cada vértice x de G, el conjunto de bolsas de T que contienen a x es un subgrafo conexo de $V(T)$.

La descomposición de árbol de un grafo infinito es un árbol infinito y una función que satisface las condiciones **Td1**, **Td2** y **Td3** de antes. El ancho de $(T, \mathbf{v})$ es el máximo (o supremo si T es infinito) de $|\mathbf{v}(b)| - 1$ cuando b varía sobre $V(T)$.

El treewidth de un grafo G, denotado $\mathrm{tw}(G)$, es el mínimo ancho sobre todas las descomposiciones de árbol de G. Denotamos por TW_k al conjunto de todos los grafos (finitos) de treewidth a lo sumo k.

Informalmente, el treewidth de un grafo es un parámetro que nos indica qué tan distinto es el grafo de un árbol; entre más bajo el valor, más parecido a un árbol es. El menor treewidth posible es 1.

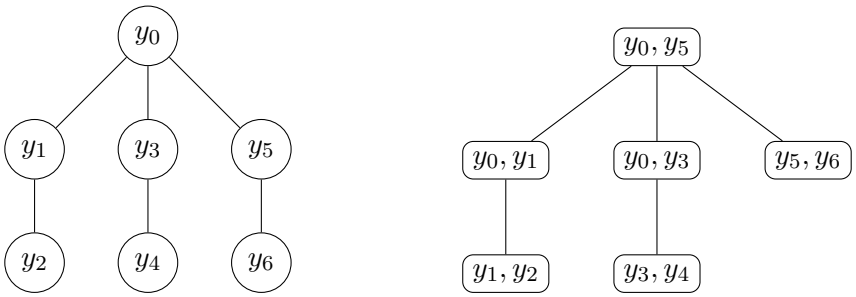


Figura 1.1: A la izquierda un árbol G y a la derecha una posible descomposición de árbol de G. El ancho de la descomposición es 1. Como no existe una menor, entonces para este grafo implica $\mathrm{tw}(G) = 1$.

Ejemplo 1. Para todo G que sea un árbol, se tiene que $\mathrm{tw}(G) = 1$. Tomemos la descomposición de árbol tal que por cada arista de G tenemos un vértice en la descomposición y cuya bolsa tiene adentro las variables que conecta. Los vértices de la descomposición se conectan si las aristas asociadas compartían un nodo en el árbol G. Notemos que esta descomposición descrita cumple con las propiedades **Td1**, **Td2**, **Td3** y tiene ancho 1. Ver Figura 1.1 para un caso particular.

Una descomposición de camino de un grafo G es un par $(P, \mathbf{v})$ que debe ser una descomposición de árbol con P restringido a ser un camino. El pathwidth de un grafo G, denotado $\mathrm{pw}(G)$, es el mínimo ancho sobre todas las descomposiciones de caminos de G. Denotamos por PW_k al conjunto de todos los grafos (finitos) de pathwidth a lo sumo k. Ver Figura 1.2 para un ejemplo.

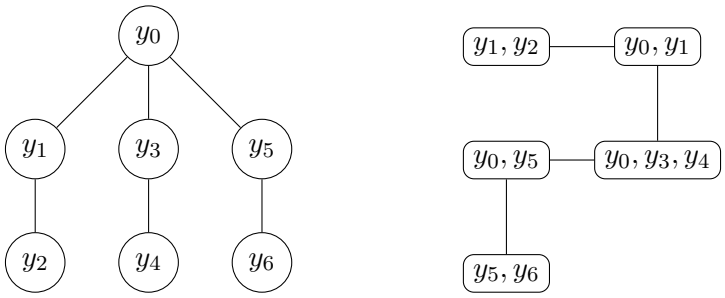


Figura 1.2: A la izquierda un árbol G y a la derecha una posible descomposición de camino de G. Note que $\mathrm{pw}(G) = 2$, pues cualquier intento de crear una descomposición de camino con a lo sumo 2 nodos en cada bolsa rompe la regla **Td3**. Si eliminamos alguno de los nodos y_2, y_4, y_6 queda un árbol de $\mathrm{pw} = 1$ (por ser caterpillar), en otras palabras, este árbol resulta ser un menor prohibido minimal de la clase PW_1.

Ejemplo 2. Todo camino tiene $\mathrm{pw} = 1$, lo cual sale por un razonamiento similar al usado para el treewidth de árboles del Ejemplo 1. Sin embargo, los caminos no son los únicos grafos con pathwidth 1, como queda establecido en la siguiente proposición de propiedades de treewidth y pathwidth.

Proposición 1. *Son verdaderas las siguientes propiedades:*

(1) Para todo G, $\mathrm{tw}(G) \leq \mathrm{pw}(G)$, y por lo tanto $\mathrm{PW}_k \subseteq \mathrm{TW}_k$ para todo k.

(2) Un grafo conexo G es un árbol si y solo si $\mathrm{tw}(G) = 1$.

(3) Un árbol T es caterpillar si y solo si $\mathrm{pw}(T) = 1$.

(4) Si G es un k-clique con $k \geq 2$, entonces $\mathrm{tw}(G) = \mathrm{pw}(G) = k - 1$.

(5) Si H es subgrafo o un menor de G, entonces $\mathrm{tw}(H) \leq \mathrm{tw}(G)$ y $\mathrm{pw}(H) \leq \mathrm{pw}(G)$.

(6) Es NP-completo *determinar si un grafo G tiene treewidth (o pathwidth) a lo sumo un valor dado k.*

(7) Cuando k es cualquier valor constante fijo, se pueden reconocer los grafos de treewidth k, y dar una correspondiente descomposición de árbol de ancho k, en tiempo lineal.

Las propiedades (1)-(5) salen por definición, mientras que los resultados de complejidad (6) y (7) son herramientas bien conocidas en el área y cuyas pruebas pueden conseguirse en [Courcelle, 1990, Bodlaender, 1993].

Por otra parte, no es posible establecer un criterio similar al (1) que nos permita sumergir a TW_k como una subclase de PW_m para algún $m > k$ y esto lo podemos verificar incluso para $k = 1$, pues existen árboles que pueden tener pathwidth arbitrariamente grande. Esto se puede demostrar por el resultado de [Scheffler, 1989] donde se muestra que los árboles binarios completos tienen un pathwidth que crece con la profundidad del árbol (Figura 1.3). Este resultado también aparece citado en [Bodlaender, 1998, Theorem 67].

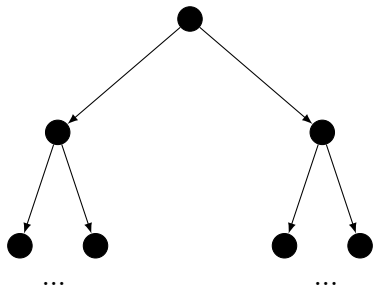


Figura 1.3: El pathwidth de los árboles binarios completos depende de su profundidad.

1.2. Lógicas modales clásicas

Los lenguajes de la lógica modal son lenguajes proposicionales a los que se les han agregado operadores sentenciales, usualmente llamados *modalidades*. A pesar de su simplicidad sintáctica, estos lenguajes resultan ser herramientas útiles para describir y razonar sobre estructuras relacionales. Las estructuras relacionales, que definiremos más adelante, son comunes en matemáticas, ciencias de la computación, inteligencia artificial y lingüística, y también se utilizan para interpretar lenguajes de primer orden.

Ahora bien, cuando trabajamos con estructuras relacionales, a menudo nos interesan las estructuras que poseen ciertas propiedades. Tal vez una determinada relación binaria transitiva sea particularmente importante. O quizás nos interesen aplicaciones donde los "dead ends", los "loops" y las "bifurcaciones" sean cruciales, o donde cada relación sea una función parcial. Sean cuales sean nuestros intereses, los operadores modales son esencialmente una forma sencilla de acceder a la información contenida en las estructuras relacionales. El método de acceso local e interno que ofrecen las modalidades es lo suficientemente potente como para describir, restringir y razonar sobre muchos aspectos interesantes e importantes de las estructuras relacionales. Ahora definamos formalmente a las lógicas modales.

Una modalidad es un símbolo en nuestra lógica que tiene una aridad y una semántica definida. Un lenguaje modal se forma por un conjunto de variables proposicionales $\mathbb{P}$ y un conjunto $\mathbb{O}$ de modalidades, ambos disjuntos. El conjunto de fórmulas de un lenguaje modal lo definimos como:

$$\varphi ::= p \mid \bot \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \triangle(\varphi_1, \ldots, \varphi_n)$$

donde $p \in \mathbb{P}$ y $\triangle \in \mathbb{O}$ tiene aridad n fija. Esta definición nos dice que una fórmula puede ser una variable proposicional, la constante proposicional para denotar falsedad (también llamada 'bottom'), una fórmula negada, una disyunción de fórmulas, o un operador modal aplicado a fórmulas (la cantidad de fórmulas dependiendo de cada operador).

Las lógicas modales pueden tener operadores modales de aridad arbitraria pero no los vamos a estudiar en esta tesis, simplemente nos limitaremos al caso de operadores modales básicos de aridad 1.

Definición 1. El lenguaje modal básico es definido por un conjunto de variables proposicionales $p \in \mathbb{P}$ y operadores modales unarios $\langle a \rangle$ (diamante) con $a \in \mathbb{A}$, con $\mathbb{P}$ y $\mathbb{A}$ infinito numerables y disjuntos. El conjunto de fórmulas lo definimos como:

$$\varphi ::= p \mid \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle a \rangle \varphi$$

Luego con esta definición podemos definir el operador modal dual $[a]$ (box) de nuestro diamante como $[a]\varphi := \neg\langle a \rangle \neg\varphi$.

Si solo se usa una etiqueta fija a, denotamos $\langle a \rangle$ como $\Diamond$ y $[a]$ como $\Box$. Bajo esta restricción, la interpretación de $\Diamond\varphi$ suele ser 'es posible que φ' y la de $\Box\varphi$ suele ser 'es necesario que φ'. Aunque como la lógica modal tiene muchas aplicaciones donde se usan múltiples operadores modales, la interpretación concreta de estos puede variar, como vemos en el siguiente ejemplo tomado de [Blackburn et al., 2001].

Ejemplo 3. El *lenguaje básico temporal* se forma con las modalidades $\mathbb{O} = \{\langle F \rangle, \langle P \rangle\}$. La interpretación de la fórmula $\langle F \rangle \varphi$ es 'φ va a ser cierto en algún momento en el futuro' mientras que $\langle P \rangle \varphi$ se interpreta como 'φ fue cierto en algún momento en el pasado'. El lenguaje básico temporal es el lenguaje fundacional de la rama de las lógicas temporales.

Para terminar de definir el lenguaje modal, nos falta explicar sobre qué tipo de modelos aplican las fórmulas, es decir, la semántica. En adelante, $\mathbb{P}$ y $\mathbb{A}$ denotan conjuntos infinito numerables, disjuntos, de proposiciones atómicas y programas atómicos, respectivamente.

Definición 2. Una estructura de Kripke es una tupla

$$K = (X, \{\rightarrow_a \mid a \in \mathbb{A}\}, \{X_p \mid p \in \mathbb{P}\}) \tag{1.1}$$

donde X es un conjunto de mundos, $\rightarrow_a \subseteq X \times X$ es una relación de transición para cada $a \in \mathbb{A}$ y $X_p \subseteq X$ es una relación unaria para cada $p \in \mathbb{P}$. Denotamos al conjunto de mundos de K como $W(K)$.

Toda estructura de Kripke se puede visualizar como un digrafo con múltiples etiquetas en los vértices y en las flechas como se muestra en la Figura 1.4, por lo que muchas propiedades usualmente estudiadas sobre grafos pueden plantearse como propiedades de estructuras de Kripke. Por ejemplo, la distancia entre dos mundos de K es simplemente la longitud del camino no dirigido más corto que los une, o indefinido si tal camino no existe.

Decimos que K es de grado finito si cada mundo de K tiene un número finito de "vecinos" a distancia 1, o en otras palabras, la cantidad de flechas que entran y salen de cada mundo es una cantidad finita. El treewidth de una estructura de Kripke es el treewidth de su (posiblemente infinito) grafo subyacente.

Dada una estructura de Kripke como la de la Definición 2, los símbolos de la lógica modal se interpretan así:

$$\begin{aligned}
[\![p]\!]_K &:= X_p \text{ para } p \in \mathbb{P}, \\
[\![\neg\varphi]\!]_K &:= X \setminus [\![\varphi]\!]_K, \\
[\![\varphi_1 \wedge \varphi_2]\!]_K &:= [\![\varphi_1]\!]_K \cap [\![\varphi_2]\!]_K, \\
[\![\langle a \rangle \varphi]\!]_K &:= \{u \in X \mid \exists v \in X.\, u \rightarrow_a v \text{ y } v \in [\![\varphi]\!]_K\}.
\end{aligned}$$

Para el resto de lenguajes a presentar, siempre nos valdremos de las Estructuras de Kripke para definir la semántica de los nuevos símbolos agregados.

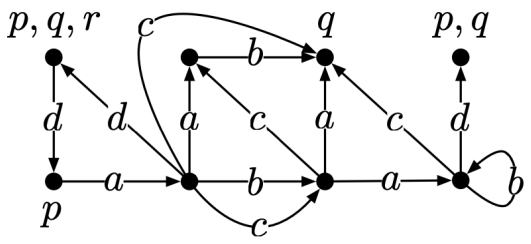


Figura 1.4: Representación visual de una estructura de Kripke, cuyos mundos están representados por los puntos negros. Las etiquetas a, b, c, d son programas atómicos y p, q, r son proposiciones atómicas. Note que entre un par de mundos pueden existir múltiples flechas, cada una con una etiqueta distinta y pueden haber mundos sin proposiciones atómicas.

1.3. Propositional Dynamic Logic

Una lógica importante para este trabajo es *Propositional Dynamic Logic*, o simplemente PDL, que fue introducida en [Fischer and Ladner, 1979].

En PDL se involucran solo operadores unarios como en la lógica modal básica, pero la distinción es que se pueden generar nuevos operadores diamantes a partir de ciertas operaciones. Cada uno de estos diamantes tiene la forma $\langle\pi\rangle$, donde π denota un programa. La interpretación de $\langle\pi\rangle\varphi$ es de 'alguna terminación de la ejecución de π del mundo presente lleva al mundo con la información φ'. La sentencia dual $[\pi]\varphi$ se interpreta como 'desde el mundo presente (donde se evalúa), cada ejecución de π lleva a algún mundo con la información φ'.

A pesar de la aparente sencillez de este lenguaje, la construcción inductiva de los programas es lo que lo hace flexible y fuertemente expresivo. Supongamos que tenemos un conjunto $\mathbb{A}$ de programas básicos a, cada uno de los cuales define una modalidad $\langle a\rangle$. Entonces podemos armar nuevos programas complejos π mediante las reglas:

- Si π es un programa básico, entonces es un programa complejo.
- (decisión) Si π_1 y π_2 son programas complejos, entonces $\pi_1 \cup \pi_2$ también, con la interpretación de que se ejecuta π_1 ó π_2.
- (composición) Si π_1 y π_2 son programas complejos, entonces $\pi_1 \circ \pi_2$ también, con la interpretación de que se ejecuta primero π_1 y luego π_2.
- (iteración) Si π es un programa complejo, entonces π^* también, con la interpretación de que se ejecuta π una finita cantidad de veces.

Notemos que todas estas son operaciones básicas de lenguajes regulares. Entendiendo este concepto, vamos a definir formalmente a PDL. Las expresiones de este lenguaje pueden ser fórmulas φ o programas π, definidos por la siguiente gramática, donde p varía en $\mathbb{P}$ y a en $\mathbb{A}$:

$$\begin{aligned}\varphi &::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\pi\rangle\\ \pi &::= \epsilon \mid a \mid \pi \cup \pi \mid \pi \circ \pi \mid \pi^* \mid \varphi?\end{aligned}$$

Note que varias de las expresiones coinciden con las ya presentadas para la lógica modal, así que en adelante solo mostraremos la semántica de las expresiones nuevas. Dada una estructura de Kripke K como (1.1), definimos la semántica de programas $[\![\pi]\!]_K \subseteq X \times X$ y de fórmulas $[\![\varphi]\!]_K \subseteq X$, como sigue:

$$\begin{aligned}[\![\langle\pi\rangle]\!]_K &:= \{u \in X \mid \exists v \in X.(u,v) \in [\![\pi]\!]_K\},\\ [\![\epsilon]\!]_K &:= \{(u,u) \mid u \in X\},\\ [\![a]\!]_K &:= \to_a\ ,\\ [\![\pi_1 \star \pi_2]\!]_K &:= [\![\pi_1]\!]_K \star [\![\pi_2]\!]_K \text{ para } \star \in \{\cup, \circ\},\\ [\![\pi^*]\!]_K &:= \text{la clausura reflexiva y transitiva de } [\![\pi]\!]_K,\\ [\![\varphi?]\!]_K &:= \{(u,u) \mid u \in X, u \in [\![\varphi]\!]_K\}.\end{aligned}$$

Escribimos $K, u \models \varphi$ para $u \in [\![\varphi]\!]_K$ y $K, u, v \models \pi$ para $(u,v) \in [\![\pi]\!]_K$ y escribimos $\varphi_1 \equiv \varphi_2$ (resp. $\pi_1 \equiv \pi_2$) si $[\![\varphi_1]\!]_K = [\![\varphi_2]\!]_K$ (resp. $[\![\pi_1]\!]_K = [\![\pi_2]\!]_K$) para cada estructura de Kripke K, en cuyo caso decimos que φ_1, φ_2 (resp. π_1, π_2) son equivalentes.

También utilizaremos las siguientes notaciones:

$$\begin{aligned}\langle\pi\rangle\varphi &\equiv \langle\pi \circ \varphi?\rangle\\ [\pi]\varphi &\equiv \neg\langle\pi\rangle\neg\varphi,\end{aligned}$$

donde la primera de estas involucra la noción original de la lógica modal de que los operadores modales actúan sobre proposiciones y la segunda es la relación usual entre box y diamante.

Definición 3. Decimos que una fórmula φ (resp. programa π) es satisfacible si existe una estructura de Kripke K y mundo $u \in W(K)$ (resp. mundos $u, v \in W(K)$) tal que $K, u \models \varphi$ (resp. $K, u, v \models \pi$).

1.3.1. Extensiones de PDL

Desde su introducción, PDL se adaptó a un creciente número de aplicaciones lo cual lo llevó a muchas modificaciones y extensiones. La adhesión de nuevos operadores a esta lógica no solo incrementa el poder expresivo de PDL sino que además se preservan varias de las propiedades computacionales que este posee.

Daremos algunas extensiones de bastante relevancia para esta tesis.

Definición 4. Converse-PDL, en adelante CPDL, se define como PDL más inverso sobre programas atómicos $\pi ::= \bar{a}$ con $a \in \mathbb{A}$, cuya semántica está dada por

$$[\![\bar{a}]\!]_K := \{(v, u) \in X^2 \mid (u \to_a v)\}.$$

Definición 5. ICPDL se define como CPDL más intersección de programas $\pi ::= \pi \cap \pi$, cuya semántica está dada por $[\![\pi_1 \cap \pi_2]\!]_K := [\![\pi_1]\!]_K \cap [\![\pi_2]\!]_K$.

Definición 6. loop-CPDL se define como CPDL más el operador loop $\varphi ::= \mathrm{loop}(\pi)$ cuya semántica está dada por $[\![\mathrm{loop}(\pi)]\!]_K := \{x \mid (x, x) \in [\![\pi]\!]_K\}$.

Esta última no será de gran relevancia para esta tesis, pero la nombramos en este punto porque podemos entenderla también como un fragmento de ICPDL, pues tenemos la equivalencia de fórmulas

$$\mathrm{loop}(\pi) \equiv \langle \pi \cap \epsilon \rangle.$$

Observación 1. Todo programa (resp. fórmula) en cualquiera de los lenguajes vistos puede pensarse como una expresión relacional aplicada sobre dos variables (resp. una variable). En efecto, para cada programa π podemos considerar el símbolo binario R_π tal que para cualquier estructura de Kripke K y mundos $u, v \in W(K)$ se tiene que $K, u, v \models R_\pi(x, y)$ si y solo si $K, u, v \models \pi$. Es decir, x e y fungen como las variables a ser interpretadas como los mundos u y v, respectivamente. Análogo para fórmulas con relaciones unarias, o alternativamente, podemos asociar una fórmula φ con la relación binaria $R_{\varphi?}(x, x)$. Para simplificar notación, siempre que haya variables involucradas, escribiremos $\pi(x, y)$ en lugar de $R_\pi(x, y)$.

Esta forma de entender a las expresiones de estos lenguajes será bastante explotada en la siguiente sección.

1.4. PDL con esteroides

Recientemente, en [Figueira et al., 2023] se introdujo el lenguaje CPDL$^+$ como una extensión de PDL con inverso (converse) e intersección. A pesar de esto, decidimos presentarlo en una sección aparte porque varias ideas requerirán algo de desarrollo.

Ya estamos listos para definir CPDL$^+$ formalmente. Consideremos $\mathbb{V}$ un conjunto infinito numerable de variables, disjunto de $\mathbb{P}$ y $\mathbb{A}$.

Definición 7. En adelante, un programa atomizado es una expresión de la forma $\pi(x, x')$, donde π es un programa en CPDL$^+$ y $x, x' \in \mathbb{V}$. Para cada programa atomizado $\pi(x, x')$ definimos $\mathit{vars}(\pi(x, x')) := \{x, x'\}$, y para un conjunto de programas atomizados C definimos $\mathit{vars}(C) := \bigcup_{A \in C} \mathit{vars}(A)$.

Definimos CPDL$^+$ como una extensión de CPDL que admite programas de la forma

$$\pi ::= C[x_s, x_t]$$

donde:

(1) C es un conjunto finito de programas atomizados;

(2) $x_s, x_t \in \mathit{vars}(C)$ (x_s y x_t podrían ser iguales);

(3) el grafo subyacente G_C de C es conexo, donde G_C está definido como $V(\mathrm{G}_C) = \mathit{vars}(C)$, y $E(\mathrm{G}_C) = \{\mathit{vars}(A) \mid A \in C\}$.

Llamamos a este tipo de expresiones programas conjuntivos. Observe que $\{x_s, x_t\} \subseteq \mathit{vars}(C)$, por lo que también definimos $\mathit{vars}(C[x_s, x_t]) := \mathit{vars}(C)$. Ver Figura 1.5 para un ejemplo visual de programa conjuntivo.

Decimos que una fórmula o programa es positivo si no contiene el símbolo $\neg$ (negación) en ninguna parte.

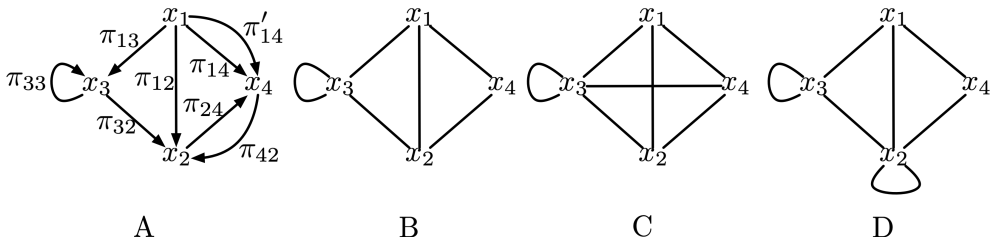


Figura 1.5: A. Representación gráfica del conjunto C que consta de los programas atomizados $\pi_{13}(x_1, x_3), \pi_{12}(x_1, x_2), \pi_{14}(x_1, x_4), \pi'_{14}(x_1, x_4), \pi_{33}(x_3, x_3), \pi_{32}(x_3, x_2), \pi_{24}(x_2, x_4), \pi_{42}(x_4, x_2)$; B. $\mathrm{G}_{\pi_\mathrm{B}}$ con $\pi_\mathrm{B} = C[x_1, x_2]$; C. $\mathrm{G}_{\pi_\mathrm{C}}$ con $\pi_\mathrm{C} = C[x_3, x_4]$; D. $\mathrm{G}_{\pi_\mathrm{D}}$ con $\pi_\mathrm{D} = C[x_2, x_2]$. Observe que $\mathrm{G}_{\pi_\mathrm{B}}, \mathrm{G}_{\pi_\mathrm{D}} \in \mathrm{TW}_2$, pero $\mathrm{G}_{\pi_\mathrm{C}} \in \mathrm{TW}_3 \setminus \mathrm{TW}_2$. Se obtienen las mismas pertenencias al sustituir la clase TW_i con el respectivo PW_i.

Dada una estructura de Kripke K, una función de asignación sobre un conjunto de programas C es una función $f : \mathit{vars}(C) \to W(K)$. Una función de asignación f sobre C, es de aceptación si $(f(x), f(x')) \in [\![\pi']\!]_K$ para cada programa atomizado $\pi'(x, x') \in C$.

La semántica $[\![C[x_s, x_t]]\!]_K$ de un programa conjuntivo sobre una estructura de Kripke K es el conjunto de todos los pares $(w_s, w_t) \in W(K) \times W(K)$ tales que $f(x_s) = w_s$ y $f(x_t) = w_t$ para alguna asignación de aceptación f sobre C.

Notemos que la definición de programa conjuntivo es recursiva, por lo que un programa conjuntivo $C[x_s, x_t]$ podría contener a su vez programas conjuntivos en C. En estos casos

cuando existe una asignación de aceptación para $C[x_s, x_t]$, esta induce una asignación de aceptación para los programas conjuntivos que están en C.

La siguiente proposición no forma parte del trabajo original [Figueira et al., 2023], pero se deduce de las definiciones ya vistas de ese trabajo.

Proposición 2. *Para toda estructura de Kripke K, mundos $u, v \in W(K)$ y programa π de* CPDL^+*, si $K, u, v \models \pi$ entonces u, v pertenecen a una misma componente conexa de K.*

Demostración. Para los casos básicos ϵ y $a, \bar{a}$ con $a \in \mathbb{A}$ esto es obvio. El resto de los casos se deducen directamente de la hipótesis inductiva. Haremos solo el análisis para programas conjuntivos: sea $\pi = C[x_s, x_t]$ y supongamos que $K, u, v \models \pi$. Entonces existe una asignación de aceptación f de C sobre K tal que $f(x_s) = u$ y $f(x_t) = v$. Por definición, tenemos que $K, f(x), f(x') \models \pi'$ para todo $\pi'(x, x') \in C$, y por hipótesis inductiva, los mundos $f(x), f(x')$ están en una misma componente conexa de K. Como G_C es conexo y la propiedad de conexidad es transitiva y simétrica, entonces todos los elementos en $\{f(x) \mid x \in \mathit{vars}(C)\}$ pertenecen a una misma componente conexa de K, en particular, esto vale para u y v. □

Observación 2. De la demostración anterior también podemos deducir que si $K, u, v \models \pi$, entonces u y v están conectados por un camino en K que solo usa programas atómicos que aparecen en π.

Para una clase $\mathcal{G}$ de grafos conexos definimos $\mathrm{CPDL}^+(\mathcal{G})$ como el fragmento de CPDL^+ cuyos programas tienen una de las formas permitidas en $\mathcal{G}$. Formalmente, para cualquier programa conjuntivo $\pi = C[x_s, x_t]$ consideramos el grafo subyacente $\mathrm{G}_{C[x_s,x_t]}$ de $C[x_s, x_t]$ como aquel que tiene $V(\mathrm{G}_{C[x_s,x_t]}) = V(\mathrm{G}_C)$ y $E(\mathrm{G}_{C[x_s,x_t]}) = \{\{x_s, x_t\}\} \cup E(\mathrm{G}_C)$. Observe que x_s y x_t siempre están conectados mediante un eje en $\mathrm{G}_{C[x_s,x_t]}$ pero no necesariamente en G_C. $\mathrm{CPDL}^+(\mathcal{G})$ es el fragmento de CPDL^+ cuyos programas conjuntivos permitidos son de la forma $C[x_s, x_t]$ con $\mathrm{G}_{C[x_s,x_t]} \in \mathcal{G}$

Las clases de grafos que nos interesa estudiar son la de TW_k y la de PW_k. La Figura 1.5 ilustra ejemplos de programas conjuntivos respecto a estas clases de grafos. Todo grafo tiene asociado un valor fijo de treewidth y pathwidth, por lo que es claro que podemos escribir

$$\mathrm{CPDL}^+ = \bigcup_{k \in \mathbb{N}} \mathrm{CPDL}^+(\mathrm{TW}_k) = \bigcup_{k \in \mathbb{N}} \mathrm{CPDL}^+(\mathrm{PW}_k).$$

En adelante cuando hablemos de la jerarquización treewidth (resp. jerarquización pathwidth) estamos haciendo referencia a la primera (resp. segunda) igualdad. A cada lenguaje $\mathrm{CPDL}^+(\mathrm{TW}_k)$ y $\mathrm{CPDL}^+(\mathrm{PW}_k)$ lo llamamos nivel (o k-ésimo nivel, si se requiere mayor precisión) de su respectiva jerarquización.

Utilizamos la notación estándar de $\bar{v}$ para denotar una tupla de elementos de algún conjunto X y $\bar{v}[i]$ al elemento i-ésimo de la tupla. Si $\bar{q} = (q_1, \ldots, q_k)$ es una tupla de mundos, r un mundo y $1 \leq i \leq k$, denotamos $\bar{q}[i \mapsto r]$ como la tupla $(q_1, \ldots, q_{i-1}, r, q_{i+1}, \ldots, q_k)$.

Un homomorfismo entre las estructuras de Kripke $K = (X, \{\to_a \mid a \in \mathbb{A}\}, \{X_p \mid p \in \mathbb{P}\})$ y $K' = (X', \{\to'_a \mid a \in \mathbb{A}\}, \{X'_p \mid p \in \mathbb{P}\})$ es una función $f : X \to X'$ tal que para cada par $w, w' \in X$ tenemos que $w \in X_p$ implica $f(w) \in X'_p$ y $(w, w') \in \to_a$ implica $(f(w), f(w')) \in \to'_a$, para cada $p \in \mathbb{P}$ y $a \in \mathbb{A}$. Una función $f : \hat{X} \to X'$ con $\hat{X} \subseteq X$ es un homomorfismo parcial

si es un homomorfismo de $\hat{K}$ en K', donde $\hat{K}$ es la subestructura de K inducida por $\hat{X}$. Denotamos por $Hom_k(K, K')$ al conjunto de todos las tuplas $(\bar{u}, \bar{v}) \in W(K)^k \times W(K')^k$ tales que $\{\bar{u}[i] \mapsto \bar{v}[i] \mid 1 \leq i \leq k\}$ es un homomorfismo parcial de K en K'. A veces abusaremos de la notación y escribiremos $(\bar{u}', \bar{v}') \in Hom_k(K, K')$ con $\bar{u}', \bar{v}'$ de dimensión $k' < k$ para denotar $(\bar{u}', \bar{v}') \in Hom_{k'}(K, K')$.

1.4.1. Simulación

En [Figueira et al., 2023] se demostró que cada nivel de la jerarquización treewidth de CPDL$^+$ está caracterizado por un juego de bisimulación. En este trabajo, nos vamos a centrar en describir la propiedad de simulación simplemente, que caracteriza el fragmento de expresiones positivas de CPDL$^+$ (es decir, aquellas fórmulas y programas que no contienen el símbolo $\neg$).

Definimos la noción de TW$_k$-simulación entre pares de estructuras de Kripke (K, K') dado un juego de suma cero $G[\rightharpoonup_k^{TW}]$, descrito de la siguiente manera: El área del juego tiene como conjunto de posiciones $S \cup D$ con:

$$S = \{s\} \times Hom_k(K, K')$$
$$D = \{d_1, ..., d_k\} \times (W(K)^k \times W(K')^k),$$

donde a Spoiler le pertenecen todas las posiciones de S y a Duplicador todas las de D. El conjunto de movimientos de $G[\rightharpoonup_k^{TW}]$ es el conjunto más pequeño que satisface que:

1. Hay un movimiento de $(s, \bar{u}, \bar{v})$ a $(d_i, \bar{u}', \bar{v})$ si $\bar{u}' = \bar{u}[i \mapsto w]$, donde w es un mundo de K a distancia ≤ 1 de $\bar{u}[j]$ para algún $1 \leq j \leq k$ con $i \neq j$.
2. Hay un movimiento de $(d_i, \bar{u}', \bar{v})$ a $(s, \bar{u}', \bar{v}')$ si $\bar{v}' = \bar{v}[i \mapsto w]$ donde w es un mundo de K' a distancia ≤ 1 de $\bar{v}[j]$ para algún $1 \leq j \leq k$ con $i \neq j$.

Dada una posición de tipo $(s, \bar{u}, \bar{v})$ o $(d_i, \bar{u}, \bar{v})$, decimos que $\bar{u}$ (resp. $\bar{v}$) es una configuración en K de Spoiler (resp. en K' de Duplicador). La condición ganadora de Duplicador es cualquier jugada infinita, lo cual explicamos en la siguiente definición.

Definición 8. Dado dos estructuras de Kripke K, K', y tuplas $\bar{v} \in W(K)^k$ y $\bar{v}' \in W(K')^k$, decimos que $K', \bar{v}'$ TW$_k$-simula $K, \bar{v}$, notado como $K, \bar{v} \rightharpoonup_k^{TW} K', \bar{v}'$ si

(R1) $(s, \bar{v}, \bar{v}')$ es una posición válida de $G[\rightharpoonup_k^{TW}]$ en (K, K').

(R2) Duplicador tiene una estrategia ganadora empezando en $(s, \bar{v}, \bar{v}')$.

siempre que todos los mundos en $\bar{v}$ estén en una misma componente conexa de K.

Observar que si hay elementos de $\bar{v}$ en distintas componentes conexas de K, entonces $K, \bar{v} \rightharpoonup_k^{TW} K', \bar{v}'$ para todo $\bar{v}' \in W(K')^k$. Aclaramos que la Definición 8 difiera de la presentada en [Figueira et al., 2023], pues corrige un error conceptual que había en el artículo.

De las reglas del juego, se infiere que si todos los elementos de $\bar{v}$ están en una misma componente conexa de K y $K, \bar{v} \rightharpoonup_k^{TW} K', \bar{v}'$, entonces los mundos de $\bar{v}'$ también deben pertenecer a una misma componente conexa de K'.

Además, por la condición (R2), ocurre que si Spoiler pasa de la configuración $\bar{u}$ a $\bar{u}'$ aplicando la regla 1, entonces Duplicador necesariamente debe poder realizar un movimiento 2 donde $\bar{v}'$ es tal que $K, \bar{u}' \rightharpoonup_k^{TW} K', \bar{v}'$. Ver Figura 1.6 para un ejemplo de estrategia ganadora para Spoiler. Por otra parte en la Figura 1.7 utilizamos una piedra menos y observamos que Duplicador gana.

El juego enunciado es importante por el siguiente teorema que relaciona el concepto de simulación con satisfacibilidad:

Teorema 1 (Extraído de [Figueira et al., 2023])**.** *Dado $k \geq 2$. Dado K, K' estructuras de Kripke de grado finito y mundos $u, v \in W(K)$ y $u', v' \in W(K')$, son equivalentes:*

1. *Para toda* $\mathrm{CPDL}^+(\mathrm{TW}_k)$*-fórmula positiva φ, tenemos que $K, v \models \varphi$ implica $K', v' \models \varphi$*
2. $K, v \rightharpoonup_{k+1}^{TW} K', v'$

y también lo siguiente es equivalente:

1. *Para todo* $\mathrm{CPDL}^+(\mathrm{TW}_k)$*-programa positivo π, tenemos que $K, u, v \models \pi$ implica $K', u', v' \models \pi$*
2. $K, u, v \rightharpoonup_{k+1}^{TW} K', u', v'$

Observación 3. Cuando hablamos de simulación durante este trabajo nos permitimos el siguiente abuso de notación: Si $\bar{u}_1$ y $\bar{u}_2$ son n-tuplas con $n < k$ (o simplemente un elemento), escribir $K, \bar{u}_1 \rightharpoonup_k^{TW} K, \bar{u}_2$ quiere decir realmente que $K, \bar{u}_1' \rightharpoonup_k^{TW} K, \bar{u}_2'$, donde $\bar{u}_i'$ es la k-tupla $\bar{u}_i[j] = \bar{u}_i[j]$ para todo $1 \leq j \leq n$ y $\bar{u}_i'[j] = \bar{u}_i[n]$ para todo $n+1 \leq j \leq k$. Es decir que extendemos la tupla con el último valor para llegar a la longitud k. En términos de juegos, la tupla $\bar{u}_1$ indica que Spoiler tiene sus primeras n piedritas dispuestas en los mundos de $\bar{u}_1$ y las $k-n$ piedritas restantes están apiladas con la n-ésima piedrita (análogo para Duplicador). Así por ejemplo, si escribimos $K, v \rightharpoonup_k^{TW} K', v'$ tenemos que Spoiler (resp. Duplicador) tiene todas sus k piedritas apiladas sobre el mundo v de K (resp. sobre el mundo v' de K'.), o si tenemos $K, u, v \rightharpoonup_k^{TW} K', u', v'$ entonces Spoiler tiene la primera piedrita sobre u y las $k-1$ piedritas restantes sobre v (análogo para Duplicador).

1.4.2. Expresividad

Para lógicas $\mathcal{L}_1$ y $\mathcal{L}_2$ con expresiones de fórmulas y programas, decimos que $\mathcal{L}_2$ es al menos tan expresivo como $\mathcal{L}_1$, notado $\mathcal{L}_1 \leqq \mathcal{L}_2$, si existe una traducción de expresiones de $\mathcal{L}_1$ a $\mathcal{L}_2$ que preserva la semántica.

Escribimos $\mathcal{L}_1 \lneqq \mathcal{L}_2$ para denotar que $\mathcal{L}_1 \leqq \mathcal{L}_2$ y que $\mathcal{L}_2 \nleqq \mathcal{L}_1$. En este caso, decimos que las lógicas $\mathcal{L}_2$ y $\mathcal{L}_1$ (en ese orden) son separables. Decimos que $\mathcal{L}_1$ es equi-expresivo a $\mathcal{L}_2$, notado $\mathcal{L}_1 \equiv \mathcal{L}_2$, si $\mathcal{L}_1 \leqq \mathcal{L}_2$ y $\mathcal{L}_2 \leqq \mathcal{L}_1$.

La lógica de primer orden es al menos tan expresiva como las lógicas modales clásicas (aquellas que no usan operadores complejos como la estrella de Kleene), y esto se demuestra por medio de *la traducción estándar* [Blackburn and van Benthem, 2007].

Más adelante utilizaremos el siguiente resultado, que relaciona varios fragmentos lógicos ya mencionados mediante la propiedad de expresabilidad.

Teorema 2 (Extraído de [Figueira et al., 2023])**.** ICPDL, $\mathrm{CPDL}^+(\mathrm{TW}_1)$ *y* $\mathrm{CPDL}^+(\mathrm{TW}_2)$ *son equiexpresivos.*

Notemos que aunque el Teorema 1 no caracteriza al fragmento $\mathrm{CPDL}^+(\mathrm{TW}_1)$ directamente, gracias a la equiexpresividad de los lenguajes $\mathrm{CPDL}^+(\mathrm{TW}_1)$ y $\mathrm{CPDL}^+(\mathrm{TW}_2)$, el juego de simulación con 3 piedras caracteriza también a $\mathrm{CPDL}^+(\mathrm{TW}_1)$.

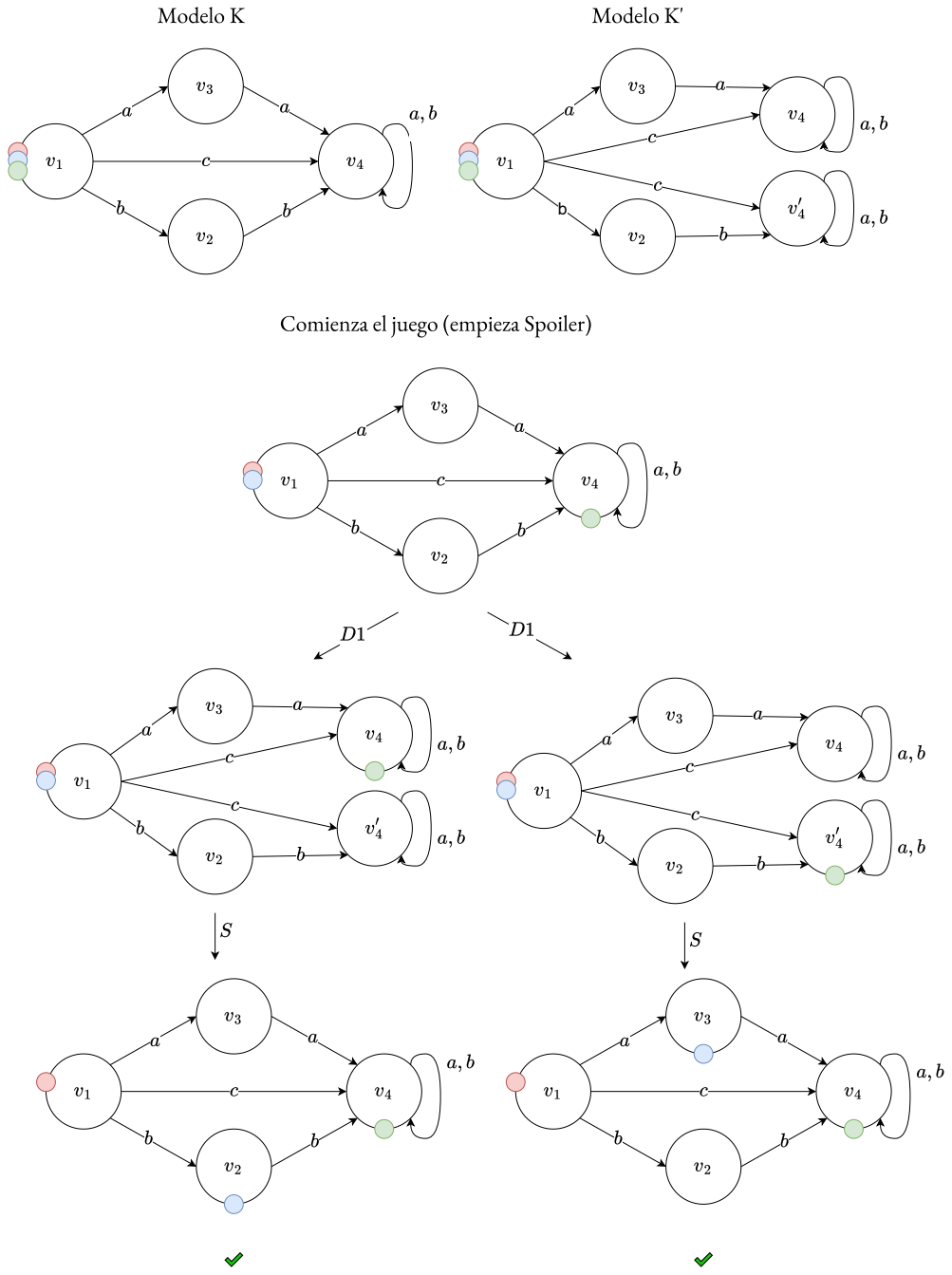


Figura 1.6: Ejemplo de juego de simulación con 3 piedras. Notemos que como Duplicador no tiene opciones de juego en las hojas de este árbol, entonces Spoiler es el ganador y además los modelos son distinguibles por una fórmula en $\mathrm{CPDL}^{+}(\mathrm{TW}_2)$. Especificamente, por $\langle C[x_1, x_4]\rangle$, donde $C = \{a(x_1, x_3), a(x_3, x_4), c(x_1, x_4), b(x_1, x_2), b(x_2, x_4)\}$.

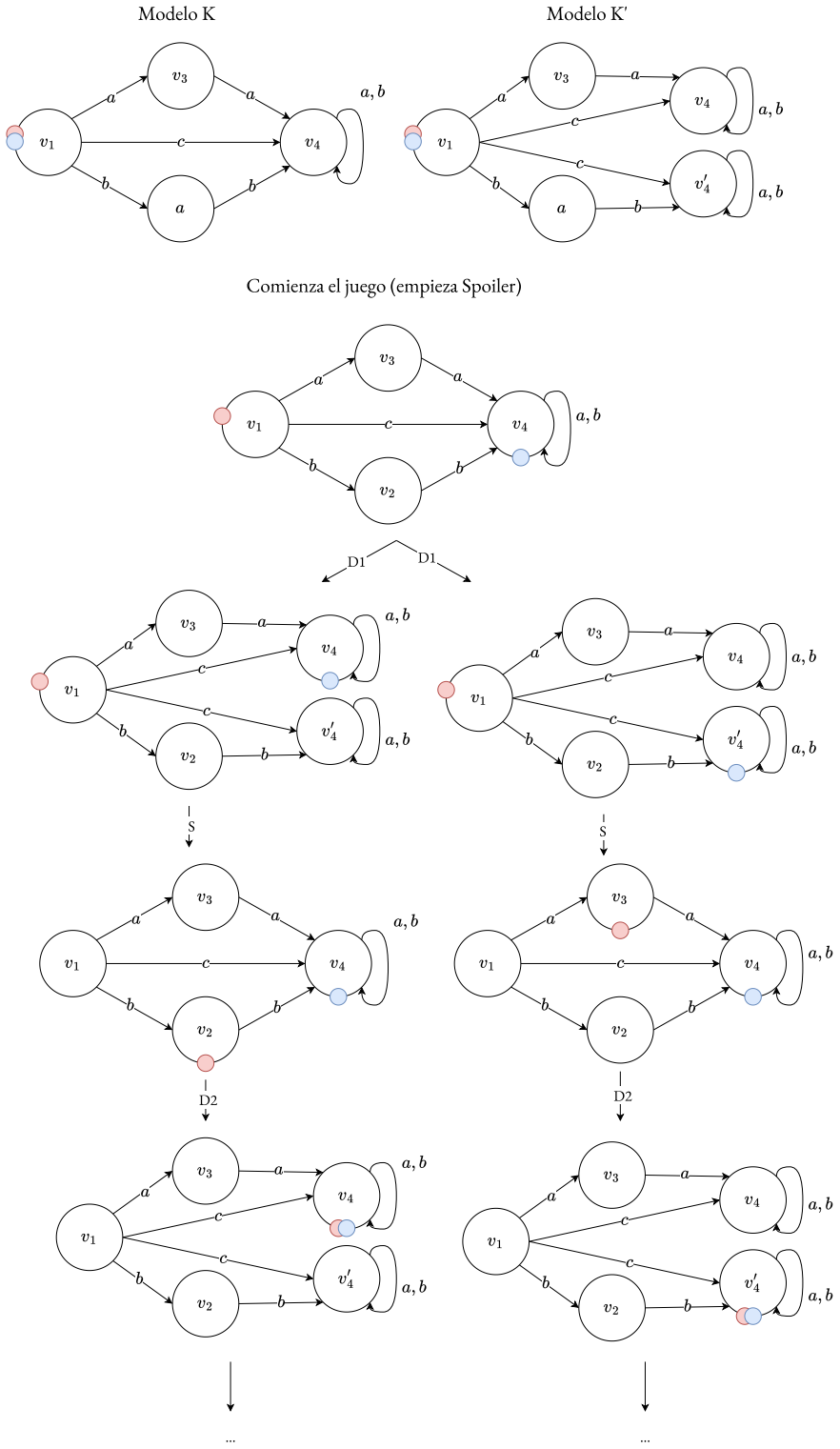


Figura 1.7: Ejemplo de juego de simulación con 2 piedras que Duplicador gana. Notemos que esto no implica que los modelos K, u y K', u' son indistinguibles en $\mathrm{CPDL}^{+}(\mathrm{TW}_1)$, pues el Teorema 1 no aplica para este lenguaje, por lo que no se contradice lo obtenido en la Figura 1.6.

Capítulo 2

Estudio jerárquico de pathwidth

En este capítulo vamos a estudiar algunas propiedades del lenguaje $\mathrm{CPDL}^+(\mathrm{PW}_k)$ que fue introducido formalmente en el capitulo anterior, e intentaremos resolver los siguientes problemas:

P1 ¿Son $\mathrm{CPDL}^+(\mathrm{TW}_k)$ y $\mathrm{CPDL}^+(\mathrm{PW}_k)$ equiexpresivos para todo k? ¿O son equiexpresivos hasta cierto nivel?

P2 ¿Para cuáles k es $\mathrm{CPDL}^+(\mathrm{PW}_{k+1})$ más expresivo que $\mathrm{CPDL}^+(\mathrm{PW}_k)$? ¿Es análoga la jerarquización pathwidth a la jerarquización treewidth?

P3 ¿Hay criterios de simulación para $\mathrm{CPDL}^+(\mathrm{PW}_k)$ con $k \geq 2$?

Algunas de estas preguntas se responderán en su totalidad y otras se resolverán parcialmente. Presentamos estos problemas en el orden en el que este proyecto se fue desarrollando. Originalmente, intentamos responder las preguntas de **P1** con algunas de las técnicas planteadas en [Figueira et al., 2023], adaptando en la medida de lo posible las demostraciones a condiciones de pathwidth. Las respuestas obtenidas para los problemas **P2** y **P3** surgieron como consecuencia de los obstáculos que surgieron al intentar responder **P1**, por lo que hemos optado por presentar los resultados en el mismo orden en el que este trabajo progresó.

2.1. Resultados en los primeros niveles vía reducciones

Nos gustaría establecer un criterio de comparación entre los lenguajes $\mathrm{CPDL}^+(\mathrm{TW}_k)$ y $\mathrm{CPDL}^+(\mathrm{PW}_k)$ para todo k, ya sea de equiexpresabilidad o de separabilidad. En principio, dado que $\mathrm{PW}_k \subset \mathrm{TW}_k$, obtenemos directamente que $\mathrm{CPDL}^+(\mathrm{PW}_k) \leqq \mathrm{CPDL}^+(\mathrm{TW}_k)$. La otra dirección requiere un poco más de esfuerzo.

En esta sección nos concentraremos en los niveles bajos de la jerarquización pathwidth, específicamente, obtendremos resultados de comparación entre los lenguajes para $k = 1$ y $k = 2$. Aunque mucho de lo que veremos puede reformularse como corolarios de algunos teoremas de [Figueira et al., 2023], hemos preferido hacer una construcción independiente y un poco más detallada en base a las definiciones presentadas en el capítulo anterior. Por ejemplo, los siguientes resultados, aunque simples, pueden servir para entender la semántica asociada a los lenguajes $\mathrm{CPDL}^+(\mathrm{TW}_k)$ y $\mathrm{CPDL}^+(\mathrm{PW}_k)$.

Proposición 3. *Sea* $C = \{\pi_1(x, y_1), \ldots, \pi_n(x, y_n)\}$ *con* π_i *programas en* CPDL^+, $x, y_1, \ldots, y_n \in \mathbb{V}$ *y sea* $C_i = \{\pi(x, y_i)\}$ *donde* $\pi = C[x, y_i]$. *Existe una asignación de aceptación sobre* C *si y solo si existe una asignación de aceptación sobre* C_i *para cualquier* i. *Más aún,* $[\![C[x, y_i]]\!]_K = [\![C_i[x, y_i]]\!]_K$ *para toda estructura de Kripke* K.

Demostración. Sea K un estructura de Kripke y supongamos que f es una asignación de aceptación sobre C. Entonces f es una extensión de $f_i : \{x, y_i\} \to W(K)$ dada por $f_i(x) = f(x)$ y $f_i(y_i) = f(y_i)$, que resulta ser de aceptación sobre C_i.

Por otro lado, si $g : \{x, y_i\} \to W(K)$ es una asignación de aceptación sobre C_i, entonces $(g(x), g(y_i)) \in [\![\pi]\!]_K = [\![C[x, y_i]]\!]_K$, y por definición existe una asignación de aceptación f sobre C tal que $f(x) = g(x)$ y $f(y_i) = g(y_i)$.

La igualdad $[\![C[x, y_i]]\!]_K = [\![C_i[x, y_i]]\!]_K$ es consecuencia del análisis anterior. □

Con la interpretación gráfica, notar que G_C de la proposición anterior es un caterpillar con vértebra $\{x, y_i\}$ y que G_{C_i} es el menor de G_C que se obtiene al contraer las aristas $\{x, y_j\}$ con $j \neq i$.

Proposición 4. *Sea* $C = \{\pi_1(x, y), \pi_2(y, z)\}$ *con* π_1 *y* π_2 *programas en* CPDL^+, $x, y, z \in \mathbb{V}$ *y sea* $C' = \{\pi(x, y)\}$ *donde* $\pi = C[x, y]$. *Existe una asignación de aceptación sobre* C *si y solo si existe una asignación de aceptación sobre* C'. *Más aún,* $[\![C[x, y]]\!]_K = [\![C'[x, y]]\!]_K$ *para toda estructura de Kripke* K.

Demostración. Análoga a la de la Proposición 3. □

Notar que G_C de la proposición anterior es un caterpillar (un camino de hecho) y que $\mathrm{G}_{C'}$ es el menor de G_C que se obtiene al contraer la arista $\{y, z\}$.

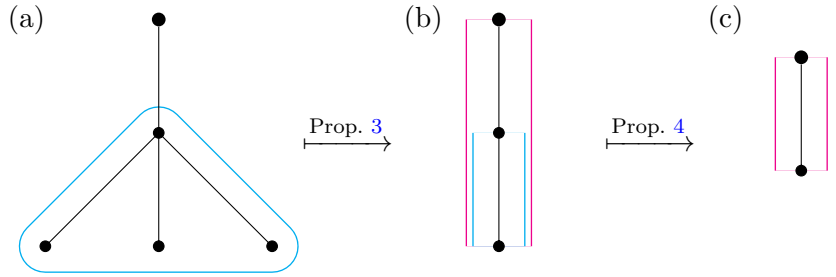


Figura 2.1: Cualquier programa conjuntivo con grafo subyacente igual al grafo (a) es equivalente a un programa cuyo grafo subyacente es una arista, por aplicabilidad de la Proposición 3 cuando compactamos los nodos dentro del triángulo azul y de la Proposición 4 cuando compactamos los nodos dentro del entorno rosa.

La siguiente proposición se demuestra aplicando un proceso similar al de la Figura 2.1.

Teorema 3. $\mathrm{CPDL}^+(\mathrm{PW}_1)$ *y* $\mathrm{CPDL}^+(\mathrm{TW}_1)$ *son equiexpresables.*

Demostración. Basta verificar que todo programa conjuntivo $C[x, y]$ de $\mathrm{CPDL}^+(\mathrm{TW}_1)$ se puede traducir a un programa equivalente en $\mathrm{CPDL}^+(\mathrm{PW}_1)$. La idea general es contraer las

hojas de una descomposición de árbol de $\mathrm{G}_{C[x,y]}$ con sus respectivos padres valiéndonos de las Proposiciones 3 y 4. Notemos además que en C hay un programa atomizado $\pi(x, y)$, de lo contrario, $\mathrm{G}_{C[x,y]}$ no sería un árbol ya que el eje $\{x, y\}$ provocaría la aparición de un ciclo en $\mathrm{G}_{C[x,y]}$.[1]

Para el siguiente algoritmo, asumimos que tenemos una codificación efectiva de programas conjuntivos en $\mathrm{CPDL}^+(\mathrm{TW}_1)$, los cuales serán transformados en un programa conjuntivo equivalente en $\mathrm{CPDL}^+(\mathrm{PW}_1)$:

Código 2.1: Programa para pasar de TW_1 a PW_1

```
1  Input: programa conjuntivo C[x,y] perteneciente a TW = 1
2  Output: programa conjuntivo C[x,y] perteneciente a PW = 1
3  // Cuando hacemos operaciones sobre el árbol subyacente de C[x,y]
4  // usamos a 'x' como raíz.
5  mientras pw(C) > 1:
6      v <- alguna hoja de mayor nivel de C que no sea hijo de 'x'
7      si #hijos(padre(v)) = 1:
8          // Aplicamos la Proposición 4
9          m, n <- abuelo(v), padre(v)
10         C' <- programas que involucran a 'm', 'n' y 'v'
11         C <- C - C' + { C'[m,n] }
12     si no:
13         // Aplicamos la Proposición 3
14         m, n <- padre(v), v
15         C' <- programas que involucran a todos los hijos de 'm'
16         C <- C - C' + { C'[m,n] }
17     fin si
18 fin mientras
```

Analicemos el Código 2.1. Primero observemos que tomamos a x como raíz, esto nos va a servir para no terminar colapsando a x ni y, pues deben ser utilizadas en el programa final. En la Figura 2.2 podemos ver como queda el árbol subyacente de la entrada C.

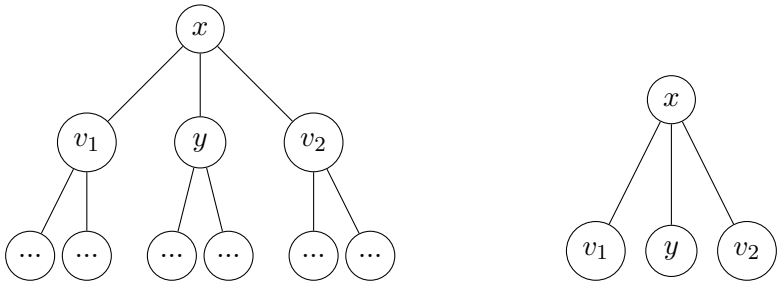


Figura 2.2: A la izquierda, el árbol G_C para un conjunto C tal que $C[x, y]$ está en $\mathrm{CPDL}^+(\mathrm{TW}_1)$. A la derecha, el resultado después de aplicar el algoritmo del Teorema 3, $G_{C[x,y]}$ es un caterpillar, por lo cual el resultante $C[x, y] \in \mathrm{CPDL}^+(\mathrm{PW}_1)$.

[1] En otras palabras, los programas conjuntivos $C[x, y]$ de $\mathrm{CPDL}^+(\mathrm{TW}_1)$ cumplen que $\mathrm{G}_{C[x,y]}$ y G_C son iguales.

En cada iteración del ciclo buscamos alguna hoja de mayor nivel posible (es decir, a distancia mayor de la raíz x) de G_C. De ahí analizamos si esa hoja tiene hermanos, si no tiene entonces removemos los programas que generan esa rama de C y agregamos un programa conjuntivo nuevo a C que equivale a esa rama (Proposición 4). Si por el otro lado la hoja tiene hermanos entonces removemos los programas que generan la bifurcación de C y agregamos un programa conjuntivo nuevo a C que equivale a esa rama (Proposición 3), finalizando entonces con una expresión que tiene pathwidth 1, por lo que el algoritmo es correcto. □

Corolario 1. *Los lenguajes* $\mathrm{CPDL}^+(\mathrm{PW}_1)$, $\mathrm{CPDL}^+(\mathrm{PW}_2)$, $\mathrm{CPDL}^+(\mathrm{TW}_1)$, $\mathrm{CPDL}^+(\mathrm{TW}_2)$ *e* ICPDL *son equiexpresables entre sí.*

Demostración. Como $\mathrm{PW}_1 \subset \mathrm{PW}_2 \subset \mathrm{TW}_2$, entonces

$$\mathrm{CPDL}^+(\mathrm{PW}_1) \leqq \mathrm{CPDL}^+(\mathrm{PW}_2) \leqq \mathrm{CPDL}^+(\mathrm{TW}_2).$$

Por los Teoremas 3 y 2, $\mathrm{CPDL}^+(\mathrm{PW}_1) \equiv \mathrm{CPDL}^+(\mathrm{TW}_2) \equiv \mathrm{CPDL}^+(\mathrm{PW}_2) \equiv \mathrm{ICPDL}$. □

Notemos que la estrategia utilizada en el Teorema 3 no es aplicable o fácilmente adaptable para niveles superiores de la jerarquía. Más específicamente, dado un conjunto C de programas conjuntivos tal que $\mathrm{G}_C \in \mathrm{TW}_k$ con $k \geq 3$, de las demostraciones hechas no se desprende una estrategia que nos permita compactar C en otro conjunto equivalente C' con $\mathrm{G}_{C'} \in \mathrm{PW}_k$, incluso considerando una descomposición de árbol apropiada para G_C. Veamos el ejemplo de la Figura 2.3, donde si intentáramos compactar una expresión con esa forma usando un método similar a los ya vistos y considerando de por medio alguna descomposición de árbol, tendríamos que poder redefinir la expresión original mediante operaciones binarias nuevas que preservan la semántica, pero cada variable está conectada con otras que pertenecen a distintos cliques que definen sus propias bolsas en el árbol de descomposición. Note que este problema surge de hecho en cualquier descomposición de árbol que se considere de este grafo, porque los cliques de un grafo siempre deben estar contenidos en una bolsa de una descomposición de árbol. Esto hace pensar que no es posible realizar una traducción como en el primer nivel de las jerarquías (Teorema 3).

Por lo que necesitamos alguna otra estrategia que nos permita concluir si $\mathrm{CPDL}^+(\mathrm{TW}_k)$ y $\mathrm{CPDL}^+(\mathrm{PW}_k)$ son equiexpresables o no. Por ejemplo, siguiendo el mismo esquema de [Figueira et al., 2023] pasamos a estudiar propiedades de modelos.

2.2. Algunas propiedades de modelos y separabilidad

En esta sección mostramos que a pesar de que $\mathrm{CPDL}^+(\mathrm{PW}_1)$ y $\mathrm{CPDL}^+(\mathrm{PW}_2)$ son equiexpresivos, tenemos una jerarquía estricta $\mathrm{CPDL}^+(\mathrm{PW}_k) \lneqq \mathrm{CPDL}^+(\mathrm{PW}_{k+1})$ a partir de $k \geq 2$. Esto será consecuencia de un resultado de [Figueira et al., 2023].

Proposición 5 (Extraído de [Figueira et al., 2023])**.** *Para todo* $k \geq 2$, $\mathrm{CPDL}^+(\mathrm{TW}_k)$ *tiene la* TW_k-model property*: Si una fórmula* $\varphi \in \mathrm{CPDL}^+(\mathrm{TW}_k)$ *es satisfacible, entonces es satisfacible en una estructura de Kripke (posiblemente infinita) de treewidth a lo sumo* k*.*

Como $\mathrm{CPDL}^+(\mathrm{PW}_k) \subset \mathrm{CPDL}^+(\mathrm{TW}_k)$ obtenemos trivialmente el siguiente corolario:

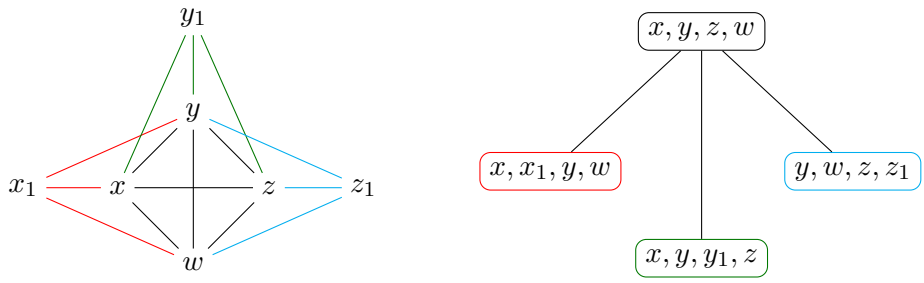


Figura 2.3: Supongamos que a la izquierda tenemos el grafo subyacente de un conjunto C de programas conjuntivos sobre las variables $x, y, z, w, x_1, y_1, z_1$. A la derecha una correspondiente descomposición de árbol que demuestra que el treewidth de este grafo es 3. Los colores se han colocado para identificar con mayor facilidad las secciones del grafo identificadas con las bolsas de la descomposición. Notemos que este grafo tiene pathwidth mayor que 3, pues cualquier intento de arreglar las cuatro bolsas en un camino, rompe las reglas de descomposición de camino.

Corolario 2. $\mathrm{CPDL}^+(\mathrm{PW}_k)$ *tiene la* TW_k*-model property.*

Con esta propiedad podremos separar a $\mathrm{CPDL}^+(\mathrm{PW}_{k+1})$ de $\mathrm{CPDL}^+(\mathrm{PW}_k)$. Aclaramos que la siguiente demostración fue tomada de [Figueira et al., 2023] para separar a $\mathrm{CPDL}^+(\mathrm{TW}_{k+1})$ de $\mathrm{CPDL}^+(\mathrm{TW}_k)$, que se adapta fácilmente a nuestro caso por propiedades de cliques (Proposición 1(4)). Reproducimos la prueba por lo ilustrativa que es.

Teorema 4. *Para todo* $k \geq 2$, $\mathrm{CPDL}^+(\mathrm{PW}_k) \lneq \mathrm{CPDL}^+(\mathrm{PW}_{k+1})$.

Demostración. Para esto demostramos que la propiedad de contener un $(k+1)$-clique puede ser expresada en $\mathrm{CPDL}^+(\mathrm{PW}_k)$ pero no en $\mathrm{CPDL}^+(\mathrm{PW}_{k-1})$ para $k \geq 3$. Consideramos la fórmula:

$$\xi_{k+1} := \langle C[x_1, x_{k+1}] \rangle \wedge \neg \langle C'[x_1, y] \rangle$$

donde:

$$C = \{a(x_i, x_j) | 1 \leq i < j \leq k+1\}$$
$$C' = \{a(x_1, y), a(y, y)\}, \text{ para algún } a \in \mathbb{A} \text{ fijo.}$$

Notemos que por definición ξ_{k+1} es una fórmula de $\mathrm{CPDL}^+(\mathrm{PW}_k)$, ya que $C[x_1, x_{k+1}]$ tiene de grafo subyacente a un $(k+1)$-clique, y el grafo subyacente de $C'[x_1, y]$ es simplemente una arista que une a x_1 con y y un loop en y. También observemos que la fórmula fuerza sobre un modelo K, u que la satisface a que contenga un $(k+1)$-clique de a's empezando desde el mundo u.

La fórmula es trivialmente satisfacible en una estructura con un $(k+1)$-clique. Y sabemos que no puede haber un $(k+1)$-clique en una estructura con *treewidth* $\leq k-1$ (consecuencia de algunas propiedades vistas en la Proposición 1). Entonces, por el Corolario 2, si ξ_k fuese expresable por alguna fórmula en $\mathrm{CPDL}^+(\mathrm{PW}_{k-1})$, tendría que ser satisfacible en una estructura con treewidth a lo sumo $k-1$, lo cual es absurdo. $\square$

Esto prueba que la jerarquización pathwidth de CPDL^+ es estrictamente creciente, como era de esperar y en analogía a lo que ocurre con la jerarquización treewidth. En este punto nos preguntamos si es posible establecer un criterio similar, para separar el lenguaje $\mathrm{CPDL}^+(\mathrm{TW}_k)$ de $\mathrm{CPDL}^+(\mathrm{PW}_k)$ para todo $k \geq 3$, también a través de una propiedad de modelos apropiada que involucre pathwidth acotado.

2.2.1. ¿Vale la Pathwidth Model property?

En [Blackburn et al., 2001] se recopilan algunas demostraciones de que la lógica modal clásica tiene la *finite model property* la cual establece, muy a groso modo, lo siguiente: si una fórmula es satisfacible, entonces también es satisfacible en un modelo finito. Esta propiedad luego abre la puerta a resultados de decibilidad.

En [Fischer and Ladner, 1979] también demuestran que en PDL vale la finite model property, e incluso dan una versión más fuerte, denominada *small model property* que establece que si una fórmula de PDL es satisfacible entonces se debe satisfacer en un modelo cuyo tamaño es exponencial en el tamaño de la fórmula. ICPDL por otra parte ya no tiene la finite model property, lo cual se verifica fácilmente con la siguiente fórmula:

$$[a^*](\langle a\rangle \wedge \neg\langle a^+ \cap \varepsilon\rangle),$$

que se satisface en cualquier modelo que tenga un camino simple infinito de a's.

En consecuencia, $\mathrm{CPDL}^+(\mathrm{TW}_k)$ tampoco tiene la finite model property (por ser extensión de ICPDL) pero sí tiene la TW_k-model property, que es justamente la Proposición 5 usada en la sección anterior y formalmente demostrada en [Figueira et al., 2023]. Vale preguntarse si es cierta una propiedad análoga para PW_k.

PW_k-*model property* : si una fórmula de $\mathrm{CPDL}^+(\mathrm{PW}_k)$ es satisfacible, ¿lo es también en una estructura de Kripke K de pathwidth a lo sumo k? Lamentablemente, demostraremos con un contraejemplo que no vale tal propiedad.

De hecho, veremos que no es necesario usar siquiera el recurso sintáctico de los programas conjuntivos.

Utilizaremos primero el siguiente teorema de [Scheffler, 1989]:

Teorema 5. *Sea T un árbol binario completo de profundidad k, entonces $pw(T) = \lceil k/2 \rceil$.*

Y ahora veamos la fórmula:

$$[(a \cup b)^*](\langle a\rangle \wedge \langle b\rangle \wedge \neg\langle (a \cup b)^+ \cap \epsilon\rangle \wedge \neg\langle a(a \cup b)^* \cap b(a \cup b)^*\rangle)$$

Analicemos por partes:

- $[(a \cup b)^*](...)$ establece que después de un camino de a's o b's pasa lo que está descripto en los puntos suspensivos.
- $\langle a\rangle \wedge \langle b\rangle$ establece que en todo mundo hay una arista saliente a y b. Esto por ser recursivo fuerza que haya un camino infinitos de a's y b's.
- $\neg\langle (a \cup b)^+ \cap \epsilon\rangle$ establece que no puede haber un camino que vuelva, y así evitando que no haya un camino infinito en el modelo.

- $\neg\langle a(a \cup b)^* \cap b(a \cup b)^*\rangle$ establece que para los hijos de cada mundo, los caminos de a's o b's no pueden juntarse entre sí, evitando así formar el árbol binario completo.

La fórmula mostrada fuerza un árbol binario completo infinito (Figura 2.4), por lo tanto, a pesar de que sea un árbol y tenga treewidth 1, su pathwidth es infinito ya que por Teorema 5 en estos árboles su pathwidth crece como su profundidad.

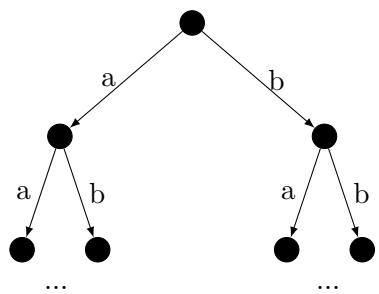


Figura 2.4: Representación del árbol binario completo que fuerza la fórmula de ICPDL.

Notemos que el contraejemplo es una fórmula de ICPDL, con lo cual demostramos que la PW_k-*model property* no vale ni siquiera para lenguajes menos expresivos que CPDL^+ como ICPDL, o dicho de otro modo, no vale ni siquiera en los primeros niveles de la jerarquía pathwidth, al recordar que ICPDL es equivalente a $\mathrm{CPDL}^+(\mathrm{PW}_2)$.

2.3. Equivalencia de modelos y simulación

Otra herramienta útil para demostrar separabilidad de lenguajes es a través de juegos de bisimulación. Todavía con el objetivo de contestar la pregunta **P1**, nos planteamos la posibilidad de hallar un juego de simulación que caracterice a $\mathrm{CPDL}^+(\mathrm{PW}_k)$. La caracterización por bisimulación de una lógica nos dice cuándo dos estructuras son equivalentes bajo esa lógica. Como ya fue mencionado previamente, haremos solo el análisis para el fragmento positivo de $\mathrm{CPDL}^+(\mathrm{PW}_k)$, o sea, solo trataremos con el caso de simulación, pero todos los resultados son adaptables a bisimulación mediante la agregación de la regla de cambio de modelos para Spoiler.

Este problema lo abordamos de dos maneras diferentes. En §2.3.1 veremos una adaptación del Teorema 1 valiéndonos de una nueva hipótesis que restringe fuertemente la semántica sobre la que actúa Duplicador. En §2.3.2 obtenemos un resultado más interesante, pues mediante nuevas condiciones de victoria logramos caracterizar al fragmento positivo de $\mathrm{CPDL}^+(\mathrm{PW}_k)$ sobre modelos generales.

2.3.1. Simulación restringida a modelos determinísticos

Nuestro primer resultado en esta sección lo encontramos restringiendo el modelo K' del Teorema 1 a estructuras determinísticas. La demostración que vamos a ver sigue bastante de cerca el formato de la demostración hecha en [Figueira et al., 2023] del Teorema 1.

Definición 9. *Una estructura de Kripke $K = (X, \{\to_a \mid a \in \mathbb{A}\}, \{X_p \mid p \in \mathbb{P}\})$ es determinística si las relaciones $\to_a$ son la imagen de una función parcial sobre X que es inyectiva, es decir, para todo $a \in \mathbb{A}$ la aplicación $f_a(u) = v$ si y solo si $u \to_a v$ con $u, v \in X$ es una función inyectiva.*

Para simplificar las demostraciones de los siguientes teoremas, que resultan algo intrincadas, suponemos de ahora en adelante que tanto K como K' están definidos sobre finitos símbolos de $\mathbb{P}$ y $\mathbb{A}$.

Teorema 6. *Sea $k \geq 2$. Dadas dos estructuras de Kripke K y K', con K' determinística y mundos $u, v \in W(K)$ y $u', v' \in W(K')$, son equivalentes:*

1. *Para toda* $\mathrm{CPDL}^+(\mathrm{PW}_k)$*-fórmula positiva φ, tenemos que $K, v \models \varphi$ implica $K', v' \models \varphi$*
2. $K, v \rightharpoonup^{TW}_{k+1} K', v'$

y también lo siguiente es equivalente:

1. *Para todo* $\mathrm{CPDL}^+(\mathrm{PW}_k)$*-programa positivo π, tenemos que $K, u, v \models \pi$ implica $K', u', v' \models \pi$*
2. $K, u, v \rightharpoonup^{TW}_{k+1} K', u', v'$

Más aún, la hipótesis de determinismo finito solo es necesaria para las implicaciones 1 a 2.

Demostración $2 \Rightarrow 1$. Es directo por Teorema 1 y porque $\mathrm{CPDL}^+(\mathrm{PW}_k) \subseteq \mathrm{CPDL}^+(\mathrm{TW}_k)$. □

Recomendamos tener fresca la Definición 8 (TW_k-simulación) para la siguiente parte.

Demostración $1 \Rightarrow 2$. Por contrarrecíproco, supongamos que $K, u, v \not\rightharpoonup^{TW}_{k+1} K', u', v'$, es decir, u, v están en una misma componente conexa de K y no ocurre (R1) o (R2). Debemos hallar un programa π en $\mathrm{CPDL}^+(\mathrm{TW}_k)$ tal que $K, u, v \models \pi$ pero $K', u', v' \not\models \pi$. Cuando no ocurre (R1), es decir, cuando $((u, v), (u', v')) \notin Hom_k(K, K')$, el resultado es bastante directo, basta analizar las finitas posibilidades para que esto último suceda, lo cual ocurre cuando:

- existe un átomo $a \in \mathbb{A}$ tal que $(u, v) \in [\![a]\!]_K$ y $(u', v') \notin [\![a]\!]_{K'}$, por lo que basta tomar $\pi = a$; o
- existe $p \in \mathbb{P}$ tal que $u \in [\![p]\!]_K$ y $u' \notin [\![p]\!]_{K'}$ (resp. $v \in [\![p]\!]_K$ y $v' \notin [\![p]\!]_{K'}$), por lo que tomamos $\pi = p? \circ \pi'$ (resp. $\pi = \pi' \circ p?$) donde π' es un camino que conecta a u con v en K.

Asumimos que no sucede (R2). Quiere decir que Spoiler tiene una estrategia para ganar en una finita cantidad de movimientos en $G[\rightharpoonup^{TW}_{k+1}]$ empezando desde $(s, (u, v), (u', v'))$. Describimos esta estrategia como un árbol cuyos vértices tienen etiquetas que reflejan las posiciones que han jugado Spoiler y Duplicador. Notemos las particularidades de este árbol:

- La raíz está etiquetada con $(s, (u, v), (u', v'))$[2]
- Todo vértice con etiqueta $(s, \bar{u}, \bar{e})$ tiene exactamente un hijo con etiqueta $(d_i, \bar{u}', \bar{e})$ consistente con $G[\rightharpoonup^{TW}_{k+1}]$.

[2] Recordemos que estamos utilizando el abuso de notación 3 y (u, v) es en realidad la k-tupla $(u, v, \ldots, v)$.

- Todo vértice con etiqueta $(d_i, \bar{u}, \bar{e})$ tiene a lo sumo un único hijo con etiqueta $(s, \bar{u}, \bar{e}')$ dado por $G[\rightharpoonup_{k+1}^{TW}]$. Si este hijo existe, la unicidad se da porque en el anterior movimiento la piedra i que movió Spoiler la tuvo que haber puesto a distancia ≤ 1 de otra piedra y Duplicador puede copiar esto, pero solo podría hacerlo de una manera por el determinismo de K'.
- Por los items anteriores, el árbol es simplemente un camino.

Consideremos el camino T y la función $\lambda : V(T) \longrightarrow W(K)^{k+1}$ donde:

1. $V(T)$ consiste de todos los vértices que no pertenecen a Spoiler en su árbol de estrategia, excepto la raíz y agregamos una arista entre el padre y el hijo de cada vértice que sacamos.
2. Proyectamos cada vértice a su componente $W(K)^{k+1}$, es decir, para todo $y \in V(T)$, $\lambda(y) = \bar{u}$, si la etiqueta de y en el árbol de estrategia de Spoiler es de la forma $(\ell, \bar{u}, \bar{v})$, con $\ell \in \{s, d_1, \ldots, d_{k+1}\}$.

De este camino se puede construir una fórmula conjuntiva $C[x_s, x_t]$ de pathwidth k que contiene los tipos homomórficos de las tuplas proyectadas (esto es, una codificación de las subestructuras de K generadas por las tuplas $\bar{u}$). Sea Q el conjunto de todos los pares (y, i) donde y es una arista de T y $1 \leq i \leq k+1$, y sea $\approx$ la relación de equivalencia más chica en Q tal que:

- $(y, i) \approx (y, i')$ si $\lambda(y)[i'] = \lambda(y)[i]$,
- $(y, i) \approx (y', i)$ si y' es hijo de y y $\lambda(y)[i] = \lambda(y')[i]$.

El programa conjuntivo C usa una variable por cada clase de equivalencia de $\approx$, que denotamos $[y, i]_\approx$, y contiene:

1. Un átomo $a([y, i]_\approx, [y', j]_\approx)$ si $y = y'$ y $(\lambda(y)[i'], \lambda(y)[j]) \in [a]_K$.
2. Un átomo $p?([y, i]_\approx, [y, i]_\approx)$ si $\lambda(y)[i] \in [p]_K$.

La variable x_s (resp x_t) es la clase $[y, i]_\approx$ donde y es la raíz de T e i es tal que $\lambda(y)[i] = u$ (resp $\lambda(y)[i] = v$).

Si en el conjunto C descrito no aparecen x_s ó x_t como variables de algún programa atomizado, debemos agregar un programa $\pi'(x_s, x_t)$ donde π' es un camino que conecta a u y v en K (esto es análogo a lo que sucede en el caso básico cuando no hay una conexión directa entre u y v). Notemos que como K está definido sobre finitos símbolos de $\mathbb{P}$ y $\mathbb{A}$, entonces C es finito.

Afirmación 1. $C[x_s, x_t] \in \mathrm{CPDL}^+(\mathrm{PW}_k)$.

Demostración. G_C es conexo. Por otro lado $G_{C[x_s,x_t]}$ tiene pathwidth k porque (T, λ) induce una descomposición de camino (T, λ') de ancho k para $G_{C[x_s,x_t]}$, donde $\lambda'(y) = \{[y, i]_\approx : 1 \leq i \leq k+1\}$ para cada $y \in V(T)$. $\square$

Afirmación 2. $K, u, v \models C[x_s, x_t]$ *y* $K', u', v' \not\models C[x_s, x_t]$.

Demostración. Para ver que $K, u, v \models C[x_s, x_t]$ tomamos la asignación $f : vars(C) \to W(K)$ dada por $f([y, i]) = \lambda(y)[i]$, la cual está bien definida porque para todo par $(y, i), (y', i') \in Q$, $(y, i) \approx (y', i')$ implica $\lambda(y)[i] = \lambda(y')[i']$ por definición de $\approx$. Luego, f es una asignación de aceptación de C.

Veamos por contradicción que $K', u', v' \not\models C[x_s, x_t]$. Si $K', u', v' \models C[x_s, x_t]$, consideremos $f' : vars(C) \to W(K')$ la correspondiente asignación de aceptación, la f del párrafo anterior y la descomposición de camino (T, λ') de la afirmación anterior. Para cada $y \in V(T)$ definamos C_y como el conjunto de todos los programas atomizados $\pi(x, x')$ en C tales que $x, x' \in \lambda'(y)$. Tenemos que f (resp. f') es una asignación de aceptación de C si y solo si f (resp. f') es una asignación de aceptación de C_y para todo $y \in V(T)$.

Consideremos además para cada $y \in V(T)$ las tuplas

$$\bar{u}_y = (f([y, 1]), \ldots, f([y, k+1]))$$
$$\bar{v}_y = (f'([y, 1]), \ldots, f'([y, k+1]))$$

de mundos de K y K', respectivamente. Como C_y codifica la subestructura de K generada por la tupla $\bar{u}_y$, cuando sustituimos la componente $\bar{u}_y[i]$ por la correspondiente variable $[y, i]$,[3] obtenemos entonces que $(\bar{u}_y, \bar{v}_y) \in Hom_k(K, K')$ para todo $y \in V(T)$.

Sea $y \to y'$ una arista de T. Vamos a ver que esta arista se corresponde con algunos movimientos válidos en $G[\rightharpoonup_{k+1}^{TW}]$ que se utilizan en el árbol de estrategia de Spoiler. Si y es el nodo raíz de T, $\bar{u}_y$ (resp. $\bar{v}_y$) se corresponde con la configuración u, v de Spoiler (resp. u', v' de Duplicator), y por lo tanto, la arista $y \to y'$ proviene de la primera jugada realizada por Spoiler en su árbol de estrategia. Esto implica que $\bar{u}_y$ y $\bar{u}_{y'}$ son iguales salvo en alguna componente $1 \leq i \leq k+1$. Como $(\bar{u}_{y'}, \bar{v}_{y'}) \in Hom_k(K, K')$, entonces $\bar{v}_y$ y $\bar{v}_{y'}$ son iguales salvo en la componente i-ésima. Por definición de $G[\rightharpoonup_{k+1}^{TW}]$ tenemos que los movimientos

$$\begin{array}{ll} \text{de } (s, \bar{u}_y, \bar{v}_y) \text{ a } (d_i, \bar{u}_{y'}, \bar{v}_y) & \text{y} \\ \text{de } (d_i, \bar{u}_{y'}, \bar{v}_y) \text{ a } (s, \bar{u}_{y'}, \bar{v}_{y'}) & \end{array} \tag{2.1}$$

son válidos, y por lo tanto, deben aparecer en la estrategia de Spoiler. Más general, supongamos que a las configuraciones $\bar{u}_y$ y $\bar{v}_y$ llegamos por medio de una serie de jugadas que aparecen en el árbol de estrategia de Spoiler. Igual que antes, la diferencia entre $\bar{u}_y$ y $\bar{u}_{y'}$, y entre $\bar{v}_y$ y $\bar{v}_{y'}$ se debe a una misma componente i, por lo que en $G[\rightharpoonup_{k+1}^{TW}]$ son válidos algunos movimientos tipo (2.1) que además deben aparecer en la estrategia de Spoiler por definición.

Notemos que el análisis anterior es válido incluso si y' es el último nodo de T, pero esto es una contradicción, porque en el árbol (camino) de estrategia de Spoiler no deberían existir ningún nodo etiquetado con alguna jugada hecha por Duplicator por debajo de la que tiene la etiqueta $(d_i, \bar{u}_{y'}, \bar{v}_y)$. □

El caso para fórmulas, de $K, u \rightharpoonup_k^{TW} K', u'$ es análogo. En este caso $C[x_s, x_t]$ es igual que antes pero ahora $x_s = x_t$ y la fórmula final es entonces $\langle C[x_s, x_t] \rangle$. □

La restricción a modelos determinísticos fue puesta con el propósito exclusivo de que la prueba del Teorema 1 (el caso $1 \Rightarrow 2$) hecha en [Figueira et al., 2023], diera como resultado

[3] Este fue el proceso realizado durante la construcción del programa $C[x_s, x_t]$.

una fórmula de $\mathrm{CPDL}^+(\mathrm{PW}_k)$. Es una imposición semántica que tiene sentido hacer para alcanzar tal propósito, pero que resulta poco satisfactoria pues limita en gran medida el tipo de modelos con los que se puede trabajar.

Corolario 3. *Dadas K, K' estructuras de Kripke con K' determinística y mundos $v \in W(K)$, $v' \in W(K')$, las siguientes son equivalentes:*

1. *existe $\varphi \in \mathrm{CPDL}^+(\mathrm{TW}_k)$ positivo tal que $K, v \models \varphi$ y $K', v \not\models \varphi$*
2. *existe $\varphi' \in \mathrm{CPDL}^+(\mathrm{PW}_k)$ positivo tal que $K, v \models \varphi'$ y $K', v \not\models \varphi'$*

Demostración $2 \Rightarrow 1$. Directo de Teorema 1 y $\mathrm{CPDL}^+(\mathrm{PW}_k) \subseteq \mathrm{CPDL}^+(\mathrm{TW}_k)$. □

Demostración $1 \Rightarrow 2$. Como existe $\varphi \in \mathrm{CPDL}^+(\mathrm{TW}_k)$ tal que $K, v \models \varphi$ y $K', v \not\models \varphi$ entonces por Teorema 1 tenemos que $K, v \not\rightharpoonup_k^{TW} K', v'$, pero como K' es determinístico, por Teorema 6, existe una fórmula φ' de $\mathrm{CPDL}^+(\mathrm{PW}_k)$ tal que $K, v \models \varphi'$ y $K', v \not\models \varphi'$. □

Hemos demostrado que si dos estructuras de Kripke (con la segunda de ellas determinística) son distinguibles mediante una fórmula de $\mathrm{CPDL}^+(\mathrm{TW}_k)$ entonces también lo son con una fórmula de $\mathrm{CPDL}^+(\mathrm{PW}_k)$, con lo cual bajo la hipótesis de que K' es determinístico, para verificar que dos estructuras son equivalentes con respecto a $\mathrm{CPDL}^+(\mathrm{TW}_k)$, bastaría con verificar la equivalencia con respecto a $\mathrm{CPDL}^+(\mathrm{PW}_k)$, que en retrospectiva, es un lenguaje más simple por contener menos expresiones. Notemos que esto no quiere decir necesariamente que ambos lenguajes son equi-expresivos (ni siquiera sobre modelos determinísticos), ya que puede ser que haya una fórmula $\varphi \in \mathrm{CPDL}^+(\mathrm{TW}_k)$ que no sea expresable en $\mathrm{CPDL}^+(\mathrm{PW}_k)$.

Ahora veremos un resultado más interesante, que no involucra restringir la clase de modelos, sino más bien modificar las reglas del juego de simulación.

2.3.2. Criterio de simulación para la jerarquía pathwidth

Definimos la noción de PW_k-simulación entre pares de estructuras de Kripke (K, K') dado un juego de suma cero $G[\rightharpoonup_k^{PW}]$. El área del juego tiene un conjunto de posiciones $S \cup D$ con:

$$S = \{s\} \times Hom_k(K, K')$$
$$D = \{d_1, \ldots, d_k\} \times (W(K)^k \times W(K')^k)$$

donde a Spoiler le pertenecen todas las posiciones de S y a Duplicador todas las de D. El conjunto de movimientos de $G[\rightharpoonup_k^{PW}]$ es el conjunto más pequeño que satisface:

1. Hay un movimiento de $(s, \bar{u}, \bar{v})$ a $(d_i, \bar{u}', \bar{v})$ si $\bar{u}' = \bar{u}[i \mapsto w]$, donde w es un mundo de K a distancia ≤ 1 de $\bar{u}[j]$ para algún $1 \leq j \leq k$ con $i \neq j$.
2. Hay un movimiento de $(d_i, \bar{u}', \bar{v})$ a $(s, \bar{u}', \bar{v}')$ si $\bar{v}' = \bar{v}[i \mapsto w]$ donde w es un mundo de K' a distancia ≤ 1 de $\bar{v}[j]$ para algún $1 \leq j \leq k$ con $i \neq j$.

Definición 10. *Dada una estructura de Kripke K y $\bar{u} \in W(K)^k$, una $\bar{u}$-sucesión es una secuencia infinita $(\bar{u}_0, \bar{u}_1, \bar{u}_2, \ldots)$ de tuplas en $W(K)^k$ tal que $\bar{u}_0 = \bar{u}$, y para todo $n \in \mathbb{N}$, $\bar{u}_{n+1} = \bar{u}_n[i \mapsto w]$, donde $w \in W(K)$ está a distancia ≤ 1 de $\bar{u}_n[j]$, para algún $1 \leq j \leq k$ con $i \neq j$.*

El desarrollo de este juego es el mismo que el definido para treewidth, pero nos enfocaremos en otros criterios de victoria para ambos jugadores: **La condición ganadora de Spoiler es que haya una configuración $\bar{u}$ y una $\bar{u}$-sucesión donde el árbol de jugadas asociado es finito. El árbol asociado de jugadas es el árbol donde Spoiler solo puede hacer los movimientos de la $\bar{u}$-sucesión. También en otras palabras, Duplicador gana si para cualquier $\bar{u}$-sucesión hay una jugada infinita.** En adelante, *estrategia ganadora* hace referencia a esta nueva noción de victoria. En la Figura 2.6 podemos ver un ejemplo de un árbol de jugadas del juego descripto.

Definición 11. *Dado dos estructuras de Kripke K, K', y mundos $\bar{v} \in W(K)^k$ y $\bar{v}' \in W(K')^k$, decimos que $K', \bar{v}'$ PW_k-simula $K, \bar{v}$, notado como $K, \bar{v} \rightharpoonup_k^{PW} K', \bar{v}'$ si*

1. *$(s, \bar{v}, \bar{v}')$ es una posición válida de $G[\rightharpoonup_k^{PW}]$ en (K, K').*
2. *Duplicador tiene una estrategia ganadora empezando en $(s, \bar{v}, \bar{v}')$.*

siempre que todos los mundos en $\bar{v}$ estén en una misma componente conexa de K.

Teorema 7. *Sea $k \geq 2$. Dadas K, K' estructuras de Kripke, con K' de grado finito y mundos $u, v \in W(K)$ y $u', v' \in W(K')$, son equivalentes:*

1. *Para toda $\mathrm{CPDL}^+(\mathrm{PW}_k)$-fórmula positiva φ, tenemos que $K, v \models \varphi$ implica $K', v' \models \varphi$*
2. *$K, v \rightharpoonup_{k+1}^{PW} K', v'$*

y también lo siguiente es equivalente:

1. *Para todo $\mathrm{CPDL}^+(\mathrm{PW}_k)$-programa positivo π, tenemos que $K, u, v \models \pi$ implica $K', u', v' \models \pi$*
2. *$K, u, v \rightharpoonup_{k+1}^{PW} K', u', v'$*

Demostración $1 \Rightarrow 2$. Por contrarrecíproco, supongamos que $K, u, v \not\rightharpoonup_{k+1}^{PW} K', u', v'$, es decir, u, v están en una misma componente conexa de K y no ocurre la condición 1 o 2. Debemos hallar un programa π en $\mathrm{CPDL}^+(\mathrm{PW}_k)$ tal que $K, u, v \models \pi$ pero $K', u', v' \not\models \pi$. Cuando no ocurre la condición 1, se procede igual que en la demostración del Teorema 6.

Asumimos que no sucede la condición 2. Quiere decir que existe una (u, v)-sucesión fija y una estrategia ganadora de Spoiler descrito como un árbol finito de movimientos en $G[\rightharpoonup_{k+1}^{PW}]$, empezando desde $(s, (u, v), (u', v'))$ y tal que los movimientos realizados por Spoiler están determinados por la (u, v)-sucesión. Supongamos que esta (u, v)-sucesión tiene la forma $(\bar{u}_0, \bar{u}_1, \ldots, \bar{u}_m, \ldots)$, donde $\bar{u}_0 = (u, v)$.[4]

Notemos las particularidades de este árbol:

- La raíz tiene etiqueta $(s, (u, v), (u', v')) = (s, \bar{u}_0, (u', v'))$.
- Todo vértice con etiqueta $(s, \bar{u}_m, \bar{e})$ tiene como único hijo un vértice con etiqueta $(d_i, \bar{u}_{m+1}, \bar{e})$, donde i es la componente que se utilizó para definir $\bar{u}_{m+1}$ a partir de $\bar{u}_m$ (Definición 10).

[4] Recordemos que estamos utilizado el abuso de notación 3 y (u, v) es en realidad la k-tupla $(u, v, \ldots, v)$.

- Todo vértice con etiqueta $(d_i, \bar{u}, \bar{e})$ tiene un hijo con etiqueta $(s, \bar{u}, \bar{e}')$ por cada posible movimiento de Duplicador en $G[\rightharpoonup_{k+1}^{PW}]$.
- Como K' tiene grado finito, entonces la cantidad de ramas es finita y por lo tanto el árbol es finito.

Consideremos la rama más larga de este árbol, cuya longitud es un entero positivo m y sea el camino P y la función $\lambda : V(P) \longrightarrow W(K)^{k+1}$ donde:

1. $V(P) = \{y_0, y_1, \ldots, y_m\}$.
2. $\lambda(y_i) = \bar{u}_i$, para todo $1 \leq i \leq m$. Note que con esta función estamos proyectando la componente $W(K)^{k+1}$ de todas las jugadas del árbol de estrategia que hemos mantenido.

Podemos ver un ejemplo de esta transformación en la Figura 2.7. De este camino P, se puede construir una fórmula conjuntiva $C[x_s, x_t]$ de pathwidth k que contiene los tipos homomórficos de las tuplas. Sea Q el conjunto de todos los pares (y, i) donde y es una arista de T y $1 \leq i \leq k+1$, y sea $\approx$ la relación de equivalencia más chica en Q tal que:

- $(y, i) \approx (y, i')$ si $\lambda(y)[i']$
- $(y, i) \approx (y', i)$ si y' es hijo de y y $\lambda(y)[i] = \lambda(y')[i]$.

El programa conjuntivo C usa una variable por cada clase de equivalencia de $\approx$, que denotamos $[y, i]_\approx$, y contiene:

1. Un átomo $a([y, i]_\approx, [y', j]_\approx)$ si $y = y'$ y $(\lambda(y)[i'], \lambda(y)[j]) \in [a]_K$.
2. Un átomo $p?([y, i]_\approx, [y, i]_\approx)$ si $\lambda(y)[i] \in [p]_K$.

La variable x_s (resp x_t) es la clase $[y, i]_\approx$ donde y es la raíz de T y $\lambda(y)[i] = u$ (resp $\lambda(y)[i] = v$). Podemos ver un ejemplo de la construcción por partes del programa conjuntivo C en la Figura 2.8.

Si en el conjunto C descrito no aparecen x_s ó x_t como variables de algún programa atomizado, debemos agregar un programa $\pi'(x_s, x_t)$ donde π' es un camino que conecta a u y v en K (como ya hemos discutido, esto se puede hacer por hipótesis de conexidad).

Afirmación 1. $C[x_s, x_t] \in \mathrm{CPDL}^+(\mathrm{PW}_k)$.

Demostración. G_C es conexo. Por otro lado $G_{C[x_s,x_t]}$ tiene pathwidth k dada por la path-decomposition (P, λ') donde $\lambda'(y) = \{[y, i]_\approx : 1 \leq i \leq k+1\}$.

Afirmación 2. $K, u, v \models C[x_s, x_t]$ *y* $K', u', v' \not\models C[x_s, x_t]$.

Demostración. Nuevamente, para ver que $K, u, v \models C[x_s, x_t]$ basta que tomemos la asignación $f : \mathit{vars}(C) \to W(K)$ dada por $f([y, i]) = \lambda(y)[i]$, la cual es una asignación de aceptación de C (análogo a la demostración del Teorema 6).

Veamos por contradicción que $K', u', v' \not\models C[x_s, x_t]$. Si $K', u', v' \models C[x_s, x_t]$, consideremos $f' : \mathit{vars}(C) \to W(K')$ la correspondiente asignación de aceptación, la f del párrafo anterior y la descomposición de camino (P, λ') de la afirmación anterior. Para cada $y_i \in V(P)$ definamos C_i como el conjunto de todos los programas atomizados $\pi(x, x')$ en C tales que $x, x' \in \lambda'(y_i)$. Tenemos que f (resp. f') es una asignación de aceptación de C si y solo si f (resp. f') es una asignación de aceptación de C_i para todo $0 \leq i \leq m$.

Consideremos además para cada $y_i \in V(P)$ las tuplas

$$\bar{u}_i = (f([y_i, 1]), \ldots, f([y_i, k+1]))$$
$$\bar{v}_i = (f'([y_i, 1]), \ldots, f'([y_i, k+1]))$$

de mundos de K y K', respectivamente. Como C_i codifica la subestructura de K generada por la tupla $\bar{u}_i$, cuando sustituimos la componente $\bar{u}_i[j]$ por la correspondiente variable $[y, j]$,[5] obtenemos entonces que $(\bar{u}_i, \bar{v}_i) \in Hom_k(K, K')$ para todo $y_i \in V(P)$.

Sea $y_i \to y_{i+1}$ una arista de P. Vamos a ver que esta arista se corresponde con algunos movimientos válidos en $G[\rightharpoonup_{k+1}^{PW}]$ que se utilizan en el árbol de estrategia de Spoiler. Si $i = 0$, $\bar{u}_0$ (resp. $\bar{v}_0$) se corresponde con la configuración u, v de Spoiler (resp. u', v' de Duplicator), y por lo tanto, la arista $y_0 \to y_1$ proviene de la primera jugada realizada por Spoiler en su árbol de estrategia. Esto implica que $\bar{u}_0$ y $\bar{u}_1$ son iguales salvo en alguna componente $1 \leq j \leq k+1$. Como $(\bar{u}_1, \bar{v}_1) \in Hom_k(K, K')$, entonces $\bar{v}_0$ y $\bar{v}_1$ son iguales salvo en la componente j-ésima. Por definición de $G[\rightharpoonup_{k+1}^{PW}]$ tenemos que los movimientos

$$\begin{aligned} &\text{de } (s, \bar{u}_i, \bar{v}_i) \text{ a } (d_j, \bar{u}_{i+1}, \bar{v}_i) \qquad \text{y} \\ &\text{de } (d_j, \bar{u}_{i+1}, \bar{v}_i) \text{ a } (s, \bar{u}_{i+1}, \bar{v}_{i+1}) \end{aligned} \tag{2.2}$$

son válidos (para $i = 0$), y por lo tanto, deben aparecer en el árbol de estrategia de Spoiler. Más general, supongamos que a las configuraciones $\bar{u}_i$ y $\bar{v}_i$ llegamos por medio de una serie de jugadas que aparecen en el árbol de estrategia de Spoiler. Igual que antes, la diferencia entre $\bar{u}_i$ y $\bar{u}_{i+1}$, y entre $\bar{v}_i$ y $\bar{v}_{i+1}$ se debe a una misma componente j, por lo que en $G[\rightharpoonup_{k+1}^{PW}]$ son válidos algunos movimientos tipo (2.2) que además deben aparecer en la estrategia ganadora de Spoiler por definición.

Notemos que el análisis anterior es válido incluso para y_m que es el último nodo de P, pero esto es una contradicción, porque en el árbol de estrategia de Spoiler no debería existir ningún nodo etiquetado con alguna jugada hecha por Duplicador por debajo de la que tiene la etiqueta $(d_i, \bar{u}_m, \bar{v}_{m-1})$.

□

Demostración $2 \Rightarrow 1$. Vamos a demostrar algo más fuerte que lo que se pide en el enunciado y vamos a hacer inducción sobre $\mathrm{CPDL}^+(\mathrm{TW}_k)$ en lugar de $\mathrm{CPDL}^+(\mathrm{PW}_k)$. En lo que sigue, haremos una reproducción de la demostración que se hizo de esta implicación del Teorema 1, tal como aparece en [Figueira et al., 2023]. Hacemos esto porque posteriormente habrá una consecuencia importante en nuestro estudio.

Demostraremos por inducción en las expresiones de $\mathrm{CPDL}^+(\mathrm{TW}_k)$ lo siguiente: para cualquier $(\bar{u}, \bar{v}) \in \{(v, v'), ((u, v), (u', v'))\}$, asumiendo que $K, \bar{v} \rightharpoonup_{k+1}^{PW} K', \bar{v}'$ y $K, \bar{v} \models e$, para alguna expresión $e \in \mathrm{CPDL}^+(\mathrm{TW}_k)$, mostraremos que $K', \bar{v}' \models e$. En adelante, llamamos $\mathrm{CPDL}^+(\mathrm{TW}_k)$-$f$ (resp. $\mathrm{CPDL}^+(\mathrm{TW}_k)$-$p$) al conjunto de fórmulas (resp. programas) de $\mathrm{CPDL}^+(\mathrm{TW}_k)$.

- $e = p$ para $p \in \mathbb{P}$: Como $K, v \models e$ y como $K, v \rightharpoonup_{k+1}^{PW} K', v'$ implica que hay un homomorfismo parcial que mapea $v \mapsto v'$, entonces $K', v' \models e$.

[5] Este fue el proceso realizado durante la construcción del programa $C[x_s, x_t]$.

- $e = a$ para $a \in \mathbb{A}$: Como $K, v, u \models e$ y como $K, v, u \rightharpoonup_{k+1}^{PW} K', v', u'$ implica que hay un homomorfismo parcial que mapea $v \mapsto v'$ y $u \mapsto u'$, entonces $K', v', u' \models e$.
- $e = \bar{a}$ para $a \in \mathbb{A}$: Análogo al caso anterior.
- $e = \varphi_1 \wedge \varphi_2$ para $\varphi_1, \varphi_2 \in \text{CPDL}^+(\text{TW}_k)$-$f$: Si $K, v \models \varphi_1 \wedge \varphi_2$ entonces $K, v \models \varphi_1$ y $K, v \models \varphi_2$. Por HI, $K', v' \models \varphi_1$ y $K', v' \models \varphi_2$, por lo tanto, $K', v' \models \varphi_1 \wedge \varphi_2$.
- $e = \pi_1 \cup \pi_2$ para $\pi_1, \pi_2 \in \text{CPDL}^+(\text{TW}_k)$-$p$: Análogo al caso anterior.
- $e = \varphi?$ para $\varphi \in \text{CPDL}^+(\text{TW}_k)$-$f$: Si $K, v, u \models \varphi?$ entonces $u = v$ y $K, u \models \varphi$. Por HI $K', u' \models \varphi$ y por lo tanto $K, u', v' \models \varphi?$.
- $e = \langle \pi \rangle$ para $\pi \in \text{CPDL}^+(\text{TW}_k)$-$p$: Si $K, w \models \langle \pi \rangle$ entonces existe un $\tilde{v}$ tal que $(v, \tilde{v}) \in [\pi]_K$. Como tenemos que v y $\tilde{v}$ son parte de la misma componente en K y $K, v \rightharpoonup_k^{PW} K', v'$, entonces esto implica que existe $\tilde{v}'$ tal que $K, v, \tilde{v} \rightharpoonup_k^{PW} K', v', \tilde{v}'$. Más precisamente, $\tilde{v}'$ se puede obtener usando la estrategia ganadora de Duplicador, pues si Spoiler realiza una serie de movimientos para pasar de la configuración v a la configuración $v, \tilde{v}$, estas jugadas deben ser replicadas por Duplicador en K', pudiendo así pasar de la configuración v' a la configuración $v', \tilde{v}'$ preservando la relación $\rightharpoonup_k^{PW}$. Por HI, $K', v', \tilde{v}' \models \pi$, de donde $K', v' \models \langle \pi \rangle$.
- $e = \pi^*$ para $\pi \in \text{CPDL}^+(\text{TW}_k)$-$p$: Como $K, u, v \models \pi^*$ y $K, u, v \rightharpoonup_k^{PW} K', u', v'$ procedemos por inducción en el número de iteraciones. El caso base, cuando $K, u, v \models \pi^0$, que pasa solamente si $u = v$, lo que implica que $u' = v'$ por definición de $\rightharpoonup_k^{PW}$, y entonces $K', v', u' \models \pi^0$. Para el caso inductivo, notemos que π^n es equivalente a $\tilde{\pi} = C[x_0, x_n]$ para $C[x_0, x_n] = \{\pi(x_i, x_{i+1}) | 0 \leq i < n\}$. Como las subexpresiones de $\tilde{\pi}$ son iguales a las de π^* entonces reemplazamos π^n por $\tilde{\pi}$ y podemos invocar el caso anterior, obteniendo que $K', u', v' \models \pi^n$.
- $e = \pi \circ \pi'$ para $\pi, \pi' \in \text{CPDL}^+(\text{TW}_k)$-$p$: Caso similar al anterior.

Ahora tenemos que ver para el caso donde e es un programa conjuntivo perteneciente a $\text{CPDL}^+(\text{PW}_k)$-$p$. Para esto, vamos a trabajar con la versión general de programas conjuntivos $C[\bar{z}]$, donde $\bar{z}$ puede tener $1 \leq t \leq k+1$ cantidad de variables de $vars(C)$ y el grafo subyacente es G_C más todas las aristas $\{y, y'\}$ para cada par distinto de variables de $\bar{z}$. La semántica $[\![C[\bar{z}]]\!]_K$ es el conjunto de *t-tuplas* $\bar{u}$ de K tal que $f(\bar{z}) = \bar{u}$ para alguna asignación de aceptación f.

Afirmación 3. *Por cada $C \subset \hat{C}$ y cada tupla $\bar{x} = (x_1, \ldots, x_n) \in vars(C)^n$ con $n \leq k+1$ tal que $G_{\hat{C}[\bar{x}]} \in TW_k$, para todo $\bar{v} \in W(K)^n$ y $\bar{v}' \in W(K')^n$: si $K, \bar{v} \rightharpoonup_{k+1}^{PW} K', \bar{v}'$ y $\bar{v} \in [\![C[\bar{x}]]\!]_K$ entonces $\bar{v}' \in [\![C[\bar{x}]]\!]_{K'}$.*

Demostración. Vamos a demostrar lo anterior por inducción en el tamaño del programa conjuntivo. Sea (T, v) una descomposición de árbol de G_π de ancho k para $\pi = C[\bar{x}]$. Sin pérdida de generalidad, supongamos que la bolsa raíz de T es $v(b) = \{x_1, \ldots, x_n\}$.

Supongamos $(K, \bar{v}) \rightharpoonup_{k+1}^{PW} (K', \bar{v}')$ y $\bar{v} \in [\![\pi]\!]_K$. Queremos ver que $\bar{v}' \in [\![\pi]\!]_{K'}$. Sea h una asignación de aceptación en K tal que $\bar{v} = (h(x_1), \ldots, h(x_n))$.

Supongamos que la raíz b de T tiene l hijos $b_1, \ldots, b_l$ y los correspondientes subárboles son $T_1, \ldots, T_l$. Sea $\bar{y}_i$ algún vector que se forma de agarrar todas las variables del conjunto $v(b_i) \cap v(b)$. Sea $\bar{v}_0 = \bar{v}$ y $\bar{v}_i = h(\bar{y}_i)$ para cada $1 \leq i \leq l$.

Sea C_0 el conjunto de todos los átomos de C en la raíz. Sea C_i el conjunto de todos los átomos de $C \setminus C_0$ contenidos en bolsas de T_i (tal que ambas variables estén en alguna bolsa juntas).

Podemos observar que $K, \bar{v} \models C[\bar{x}]$ si y solo si $K, \bar{v}_i \models C_i[\bar{y}_i]$ para todo $0 \leq i \leq l$. Tenemos dos posibilidades, que $C_0 = C$ o $C_0 \subsetneq C$.

- Caso $C_0 = C$: Esto implica que todos los átomos de C usan variables de $\bar{x}$. Consideremos la función $h' : vars(C) \to K'$ tal que $h'(x_i) = \bar{v}'[i]$. Veamos que h' es una función de asignación para K'. Para cada átomo $\pi'(x_i, x_j) \in C = C_0$, tenemos que $(h(x_i), h(x_j)) = (\bar{v}[i], \bar{v}[j]) \in [\![\pi']\!]_K$ por hipótesis de h. También tenemos que $K, \bar{v}[i], \bar{v}[j] \rightharpoonup^{PW}_{k+1} K', \bar{v}'[i], \bar{v}'[j]$. Entonces $(\bar{v}'[i], \bar{v}'[j]) \in [\![\pi']\!]_{K'}$ por hipótesis inductiva de π'. Esto implica que h' es un asignación de aceptación en K'.

- Caso $C_0 \subsetneq C$: Entonces $|C_i| < |C|$ para todo i. Podemos aplicar la hipótesis inductiva sobre cada C_i porque $K, \bar{v}_i \models C_i[\bar{y}_i]$ y $K, \bar{v}_i \rightharpoonup^{PW}_{k+1} K', \bar{v}'_i$, obteniendo $K', \bar{v}'_i \models C_i[\bar{y}_i]$ para alguna asignación de aceptación h'_i en K' que mapea $\bar{y}_i \mapsto \bar{v}'_i$. Esto implica que $h' := h'_0 \cup \ldots \cup h'_l$ es una asignación de aceptación en K' y entonces $K', (h'(x_1), \ldots, h'(x_n)) \models C[\bar{x}]$.

Por esta afirmación, concluimos que si $K, \bar{v} \rightharpoonup^{PW}_{k+1} K', \bar{v}'$ y $K, \bar{v} \models e$ para alguna expresión $e \in \mathrm{CPDL}^+(\mathrm{TW}_k)$, entonces $K', \bar{v}' \models e$.

□

La razón por la que hemos reproducido la demostración inductiva anterior, es porque en el transcurso de nuestro análisis notamos que esta prueba depende exclusivamente de dos factores: uno, la cantidad de piedras en juego, y dos, que el juego sea lo suficientemente permisivo para pasar de una configuración a otra. Para ambos criterios de juego, ya sea $G[\rightharpoonup^{PW}_k]$ o $G[\rightharpoonup^{TW}_k]$, no resulta fundamental que las descomposiciones consideradas de un programa conjuntivo tengan forma de árbol, sino que podamos mediante una estrategia ganadora de Duplicador (que por hipótesis la tiene), podamos realizar con movimientos de Spoiler todos los pasos necesarios para garantizar la satisfacibilidad del programa en K', u', v'. Note que del planteamiento anterior se obtiene entonces que

Corolario 4. *Sea $k \geq 2$. Dadas K, K' estructuras de Kripke, con K' de grado finito y mundos $u, v \in W(K)$ y $u', v' \in W(K')$, son equivalentes:*

1. *Para todo* $\mathrm{CPDL}^+(\mathrm{TW}_k)$*-fórmula positiva φ, tenemos que $K, v \models \varphi$ implica $K', v' \models \varphi$*
2. $K, v \rightharpoonup^{PW}_{k+1} K', v'$

y también lo siguiente es equivalente:

1. *Para todo* $\mathrm{CPDL}^+(\mathrm{TW}_k)$*-programa positivo π, tenemos que $K, u, v \models \pi$ implica $K', u', v' \models \pi$*
2. $K, u, v \rightharpoonup^{PW}_{k+1} K', u', v'$

Demostración $1 \Rightarrow 2$. Directo por $1 \Rightarrow 2$ del Teorema 7 y que $\mathrm{CPDL}^{+}(\mathrm{PW}_k) \subset \mathrm{CPDL}^{+}(\mathrm{TW}_k)$. □

Demostración $2 \Rightarrow 1$. En la demostración $1 \Rightarrow 2$ del Teorema 7 demostramos esto directamente. □

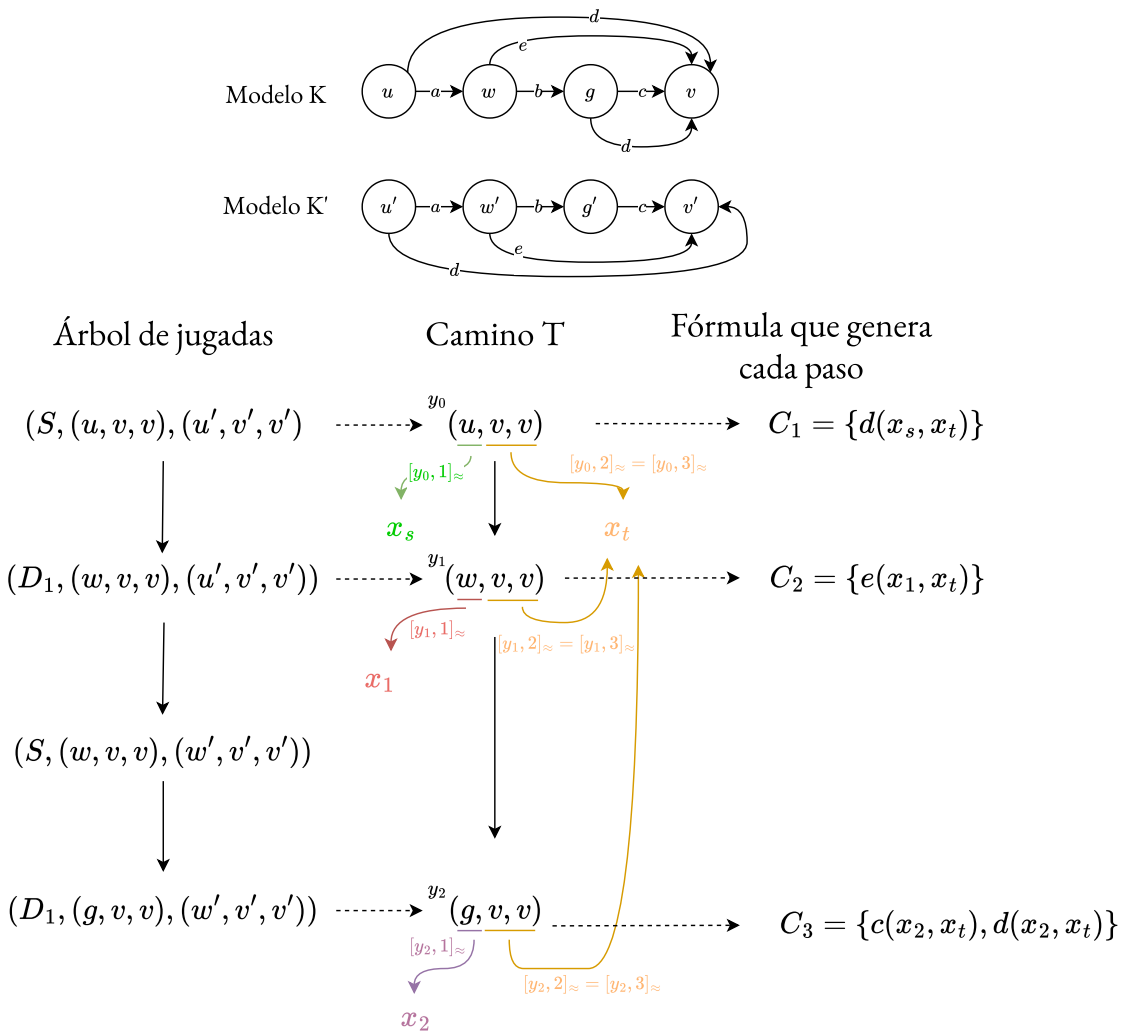
Por otra parte, obtenemos que el concepto de indistinguibilidad con respecto a los lenguajes $\mathrm{CPDL}^{+}(\mathrm{TW}_k)$ y $\mathrm{CPDL}^{+}(\mathrm{PW}_k)$ es de hecho el mismo sobre estructuras en general, a diferencia de lo obtenido en la sección anterior, donde el juego de simulación era el mismo que se definió en [Figueira et al., 2023], pero requería una hipótesis restrictiva fuerte.

Corolario 5. *Dados K, K' estructuras de kripke, $v \in W(K), v' \in W(K')$ entonces son equivalentes:*

1. $\exists \varphi \in \mathrm{CPDL}^{+}(\mathrm{TW}_k)$ *tal que* $K, v \models \varphi$ *y* $K', v' \not\models \varphi$
2. $\exists \varphi \in \mathrm{CPDL}^{+}(\mathrm{PW}_k)$ *tal que* $K, v \models \varphi$ *y* $K', v' \not\models \varphi$

Demostración $2 \Rightarrow 1$. Directo de que $\mathrm{CPDL}^{+}(\mathrm{PW}_k) \subset \mathrm{CPDL}^{+}(\mathrm{TW}_k)$ □

Demostración $1 \Rightarrow 2$. Si existe tal φ, por Teorema 4 entonces $K, v \not\rightarrow^{PW}_{k+1} K', v'$. Luego, por Teorema 7 existe $\varphi' \in \mathrm{CPDL}^{+}(\mathrm{PW}_k)$ tal que $K, v \models \varphi'$ y $K', v' \not\models \varphi'$. □



Fórmula final

$$C = \{d(x_s,x_t), e(x_1,x_t), c(x_2,x_t), d(x_2,x_t)\}$$

Figura 2.5: Ejemplo de árbol de jugadas de juego treewidth con restricción de modelos deterministas, y transformación de la Demostración $1 \Rightarrow 2$.

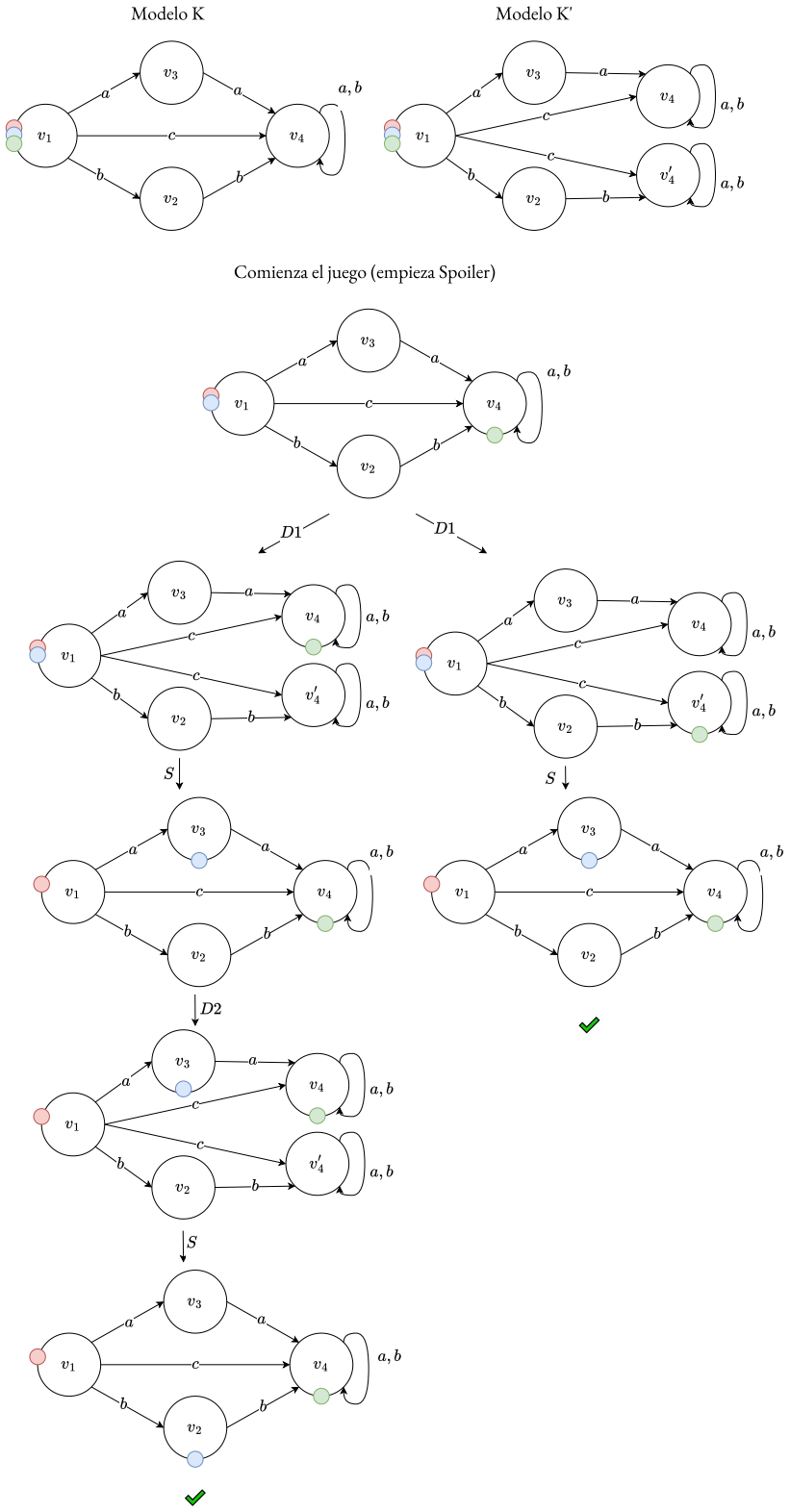


Figura 2.6: Ejemplo de juego pathwidth con 3 piedras. Notemos que la diferencia con el juego treewidth, en el 3er nivel Spoiler está obligado a realizar el mismo movimiento en ambas ramas del árbol.

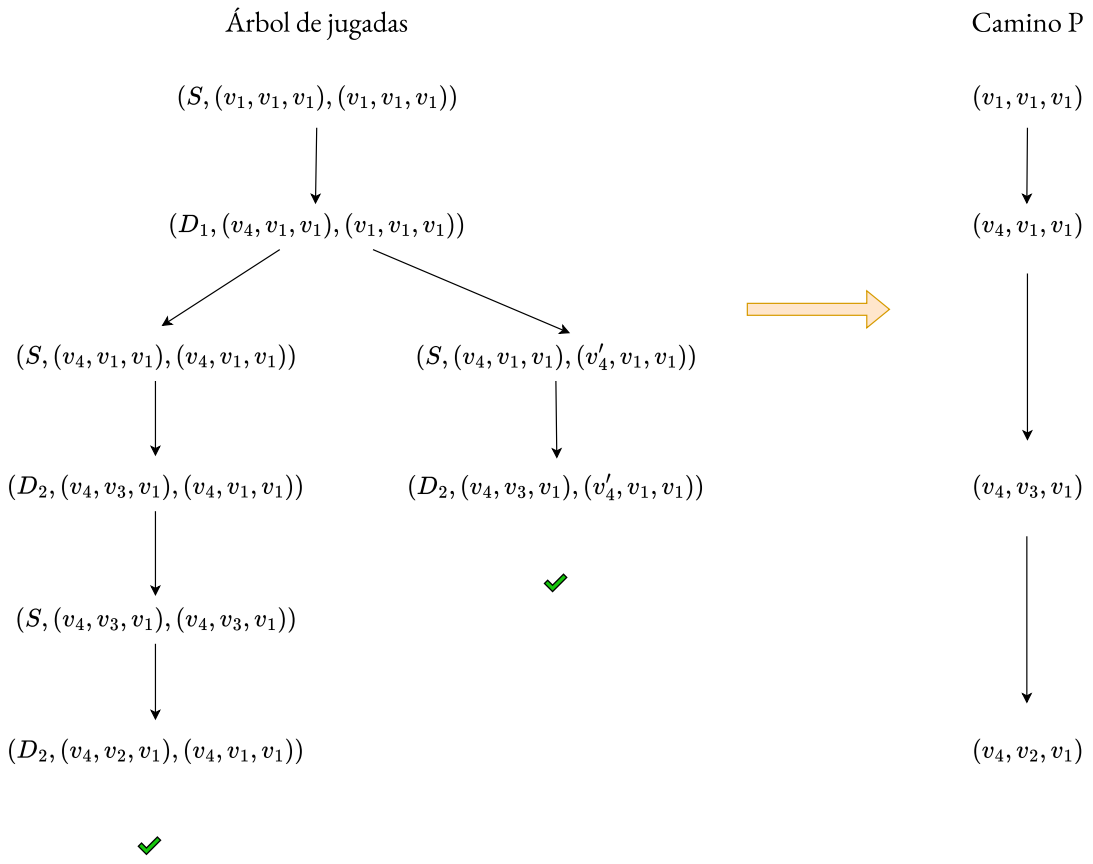


Figura 2.7: Ejemplo de transformación del árbol de jugadas al camino P de la Demostración $1 \Rightarrow 2$. En este caso, usamos como base el juego ganado por Spoiler de la Figura 2.6.

Camino P | Fórmula que genera cada paso

$(v_1, v_1, v_1) \xrightarrow{v_1 \mapsto x_s} \emptyset$

$(v_4, v_1, v_1) \xrightarrow{v_4 \mapsto z_4} c(x_s, z_4), a(z_4, z_4), b(z_4, z_4)$

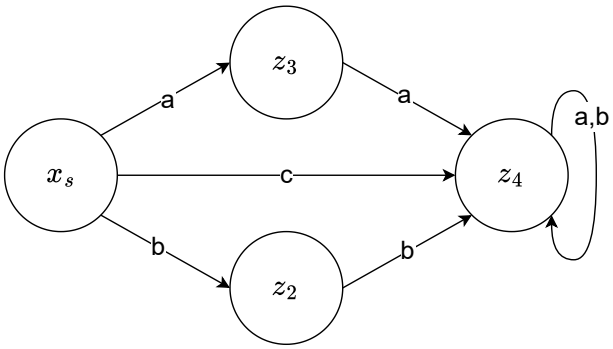
$(v_4, v_3, v_1) \xrightarrow{v_3 \mapsto z_3} c(x_s, z_4), a(z_3, z_4), a(x_s, z_3), a(z_4, z_4), b(z_4, z_4)$

$(v_4, v_2, v_1) \xrightarrow{v_2 \mapsto z_2} b(x_s, z_2), b(z_2, z_4), c(x_s, z_4), a(z_4, z_4), b(z_4, z_4)$

Fórmula final

$C = \{c(x_s, z_4), a(z_4, z_4), b(z_4, z_4), c(x_s, z_4), a(z_3, z_4), a(x_s, z_3), b(x_s, z_2), b(z_2, z_4)\}$

Grafo subyacente



Path Decomposition (equivalente a P)

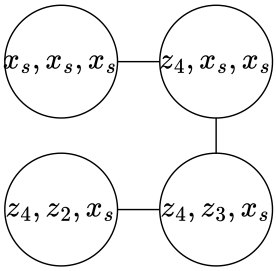


Figura 2.8: Ejemplo de como se genera la fórmula C a partir del camino P de la Demostración $1 \Rightarrow 2$. Usamos como base las Figuras 2.6 y 2.7.

Conclusiones

En esta tesis, hemos demostrado la separabilidad de varios fragmentos de CPDL^+, específicamente, las de los lenguajes $\mathrm{CPDL}^+(\mathrm{PW}_k)$, observando que se forma una jerarquía similar a la que se presenta con $\mathrm{CPDL}^+(\mathrm{TW}_k)$. Este hallazgo subraya la complejidad inherente de estos lenguajes y su estructura jerárquica.

A lo largo de nuestro estudio, hemos constatado que es imposible establecer un criterio de model property para $\mathrm{CPDL}^+(\mathrm{PW}_k)$. Este hecho refleja una de las diferencias fundamentales que existe con respecto a los lenguajes $\mathrm{CPDL}^+(\mathrm{TW}_k)$, lo que subraya la necesidad de estudiar nuevos enfoques para resolver algunas cuestiones semánticas que quedaron abiertas.

Logramos restringir el juego de simulación definido en [Figueira et al., 2023] a modelos determinísticos y nos dimos cuenta que en este caso se puede simplificar el análisis de estructuras equivalentes respecto a $\mathrm{CPDL}^+(\mathrm{TW}_k)$ a una clase más chica de expresiones como $\mathrm{CPDL}^+(\mathrm{PW}_k)$.

También llegamos a obtener un juego de simulación para los distintos niveles de la jerarquía pathwidth, con lo que hemos concluido formalmente que si dos estructuras son equivalentes bajo $\mathrm{CPDL}^+(\mathrm{PW}_k)$ deben serlo también bajo $\mathrm{CPDL}^+(\mathrm{TW}_k)$.

A pesar de estos avances, queda pendiente para futuras investigaciones la cuestión de si $\mathrm{CPDL}^+(\mathrm{TW}_k)$ y $\mathrm{CPDL}^+(\mathrm{PW}_k)$ tienen la misma expresividad. Este es un problema que no pudimos resolver y tal vez requiera de nuevas técnicas y herramientas teóricas para su resolución. Conjeturamos que de hecho ambos no pueden ser equiexpresivos por la limitación de que los lenguajes solo admiten expresiones que se interpretan de manera unaria o binaria. Una situación particular que se consigue en la literatura, es que la lógica modal clásica y PDL están caracterizados por un mismo juego de bisimulación, y sin embargo, PDL es mucho más expresivo que la lógica modal. Creemos que una situación similar puede estar ocurriendo acá, en el que ambos lenguajes $\mathrm{CPDL}^+(\mathrm{PW}_k)$ y $\mathrm{CPDL}^+(\mathrm{TW}_k)$ están caracterizados por el juego de simulación $G[\rightharpoonup_k^{PW}]$ pero $\mathrm{CPDL}^+(\mathrm{TW}_k)$ es más expresivo que $\mathrm{CPDL}^+(\mathrm{PW}_k)$.

Otra línea de investigación futura que se propone es la exploración de la relación entre CPDL^+ y los menores de grafos. En particular, sería interesante investigar las implicaciones de los menores prohibidos y cómo afectan la estructura y propiedades de los modelos de ciertos fragmentos de CPDL^+. En [Robertson and Seymour, 1983] (uno de los primeros papers al respecto) encontramos varios resultados que relacionan pathwidth con menores de grafos.

En resumen, los resultados obtenidos en esta tesis no solo amplían nuestro conocimiento sobre distintos tipos de jerarquías que se pueden definir sobre CPDL^+, sino que también plantean nuevas preguntas que podrían impulsar futuras investigaciones en esta área.

Bibliografía

[Balcázar et al., 1992] Balcázar, J., Gabarró, J., and Sántha, M. (1992). Deciding bisimilarity is p-complete. *Form. Asp. Comput.*, 4(Suppl 1):638–648.

[Benevides, 2014] Benevides, M. R. (2014). Bisimilar and logically equivalent programs in pdl. *Electronic Notes in Theoretical Computer Science*, 305:5–18. Proceedings of the 8th Workshop on Logical and Semantic Frameworks (LSFA).

[Blackburn et al., 2001] Blackburn, P., de Rijke, M., and Venema, Y. (2001). *Modal Logic*. Number 53 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK.

[Blackburn and van Benthem, 2007] Blackburn, P. and van Benthem, J. (2007). *Modal Logic: A semantic perspective*. Studies in Logic and Practical Reasoning. Elsevier.

[Bodlaender, 1993] Bodlaender, H. L. (1993). A linear time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, page 226–234, New York, NY, USA. Association for Computing Machinery.

[Bodlaender, 1998] Bodlaender, H. L. (1998). A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45.

[Carreiro, 2015] Carreiro, F. (2015). Pdl is the bisimulation-invariant fragment of weak chain logic. In *2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 341–352.

[Courcelle, 1990] Courcelle, B. (1990). The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75.

[Downey and Fellows, 2002] Downey, R. G. and Fellows, M. R. (2002). Parameterized complexity. *Bulletin of Symbolic Logic*, 8(4):528–529.

[Figueira et al., 2023] Figueira, D., Figueira, S., and Pin, E. (2023). PDL on Steroids: on Expressive Extensions of PDL with Intersection and Converse. In *Annual Symposium on Logic in Computer Science (LICS)*, Proceedings of Annual Symposium on Logic in Computer Science (LICS), Boston, United States.

[Fischer and Ladner, 1979] Fischer, M. J. and Ladner, R. E. (1979). Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211.

[Göller et al., 2009] Göller, S., Lohrey, M., and Lutz, C. (2009). PDL with intersection and converse: satisfiability and infinite-state model checking. 74(1):279–314.

[Lange, 2006] Lange, M. (2006). Model checking propositional dynamic logic with all extras. *J. Appl. Log.*, 4(1):39–49.

[Robertson and Seymour, 1983] Robertson, N. and Seymour, P. (1983). Graph minors. i. excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39–61.

[Scheffler, 1989] Scheffler, P. (1989). *Die Baumweite von Graphen als ein Maß für die Kompliziertheit algorithmischer Probleme.* PhD thesis.

[Streett, 1982] Streett, R. S. (1982). Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Control*, 54(1):121–141.