UNIVERSIDAD DE BUENOS AIRES
Facultad de Ciencias Exactas y Naturales
Departamento de Matemática

Tesis de Licenciatura

Aspectos lógicos de la teoría de la complejidad descriptiva.

Mariano Merzbacher

Director: Rafael Grimson

24 de febrero de 2014

Índice general

Introducción y preliminares **3**

1. Lógicas **7**
1.1. Lógica de primer orden 8
1.2. Juegos de Ehrenfeucht-Fraïssé 11
1.2.1. Definiciones previas 11
1.2.2. El juego 12
1.2.3. Comparación de lógicas 15
1.3. Lógicas de punto fijo 15
1.3.1. Operadores de punto fijo 15
1.3.2. FO(IFP) y FO(PFP) 16
1.4. Lógicas con fórmulas infinitas 19
1.4.1. Finitas variables 20
1.4.2. Juegos con fichas 22
1.5. Fragmento existencial de la lógica de segundo orden 23
1.5.1. Σ^1_1 24
1.6. Lógicas que saben contar 25
1.6.1. Punto fijo con contadores (FO(IFP)+C) 25
1.6.2. FO+C 28

2. Complejidad Descriptiva **29**
2.1. Modelo de cómputo 29
2.1.1. Clases de complejidad 30
2.1.2. Codificación de estructuras 32
2.2. Lógicas que capturan clases de complejidad 33
2.2.1. La complejidad de la relación de satisfacción para FO . 34
2.2.2. El caso de FO(IFP) y FO(PFP) 35
2.2.3. El teorema de Immerman-Vardi 36
2.2.4. Algunas consecuencias 38
2.2.5. FO(IFP) + C sobre estructuras ordenadas 38
2.3. Una lógica para PTIME 39

3. Órdenes definibles y canonizaciones **41**
3.1. Órdenes definibles 41
3.2. Estructuras rígidas 44
3.2.1. Orden de tipos 44
3.2.2. Estructuras k-rígidas 47
3.3. Transducciones 48
3.3.1. Transducciones y complejidad 51
3.3.2. Todo es un grafo 52
3.3.3. Transducciones y definibilidad 55
3.3.4. Composición de transducciones 57
3.4. Canonizaciones 58
3.4.1. Canonizaciones normales 60
3.4.2. Canonización de árboles 61
3.4.3. Levantamiento de canonizaciones 64

Índice alfabético **69**

Bibliografía **69**

Introducción y preliminares

Introducción

La teoría de la *complejidad descriptiva* se encuentra en la intersección entre la teoría de modelos finitos y la teoría de la *complejidad computacional*. Estudia las caracterizaciones de las clases de complejidad a través del tipo de lógica necesario para expresar (describir) las *propiedades* de modelos finitos en dichas clases. Los estudios sobre la conexión entre clases de complejidad y lógicas comenzaron con Fagin [11], donde prueba la equivalencia entre la clase NPTIME y las clases definibles en Σ_1^1 (más adelante daremos una definición de esta equivalencia). El trabajo de Fagin fue el disparador para otras investigaciones, entre las que se destacan las de Immerman [20], Vardi [27] y Gurevich [16] entre otros. Esta conexión entre clases de complejidad y formalismos lógicos (a la que hacemos referencia diciendo que una lógica *captura* una clase de complejidad) permite traducir problemas de un área a la otra con facilidad, mostrando que muchas clases de complejidad tienen formulaciones naturales y ampliando la técnicas disponibles para la demostración de teoremas.

La lógica tiene la capacidad, a diferencia de los algoritmos, de expresar una propiedad sobre una estructura independientemente de su representación. Un algoritmo requiere una codificación de una estructura, lo cual impone un orden sobre los elementos de la misma. Los algoritmos hacen uso de éste constantemente (a la hora de iterar un procedimiento, de llevar un contador, etc.) por lo cual cuando una lógica no dispone de un orden, el problema de encontrar una conexión con una clase de complejidad es mucho más difícil. Es por esto que el orden juega un rol fundamental en la teoría de la complejidad descriptiva.

Nos concentramos en los aspectos lógicos de la teoría. En el Capítulo 1, definimos las distintas lógicas que utilizamos a lo largo del trabajo. Definimos también distintos juegos, que utilizamos para hacer un análisis del poder expresivo de las lógicas definidas y compararlas.

En el Capítulo 2, introducimos las nociones de teoría de la complejidad

necesarias para luego presentar los resultados clásicos de la teoría, como el teorema de Immerman-Vardi y el de Fagin. Presentamos algunas consecuencias de éstos, que ilustran como pueden traducirse desigualdades entre clases de complejidad a desigualdades entre el poder expresivo de distintas lógicas. Nos concentramos también en el problema de encontrar una lógica que capture la clase PTIME. La pregunta de si existe una lógica que exprese, exactamente, todas las propiedades decidibles en tiempo polinomial, es uno de los principales problemas abiertos de la teoría. Las lógicas propuestas hasta el momento, mostraron no tener el poder expresivo necesario.

Por último, en el Capítulo 3, introducimos herramientas que permiten extender los resultados clásicos, que presuponen un orden en las estructuras, a estructuras que no disponen de uno. Por un lado presentamos la noción de orden definible, estudiando sus alcances y limitaciones. Presentamos algunos resultados de Dawar [7], completando y resumiendo algunas demostraciones. Introducimos también la noción de canonización, que extiende la de orden definible. En particular nos concentramos en canonizaciones definibles mediante una lógica. Nos basamos en los trabajos de Grohe [15, 14, 13], dando algunas definiciones propias, que permiten expresar algunos resultados de forma más simple.

Notación y preliminares

Presuponemos cierta familiaridad con la teoría de modelos, en particular con la de modelos finitos. Una introducción a los conceptos básicos pueden encontrarse en los primeros capítulos de [22] o [23]. Las siguientes definiciones son presentadas con la intención principal de fijar notación.

Todos los lenguajes serán finitos y *relacionales*, es decir, sólo consideramos símbolos de relación, cada uno con su respectiva aridad. Nos interesaremos particularmente en estructuras *finitas*. Dado un lenguaje $\sigma = \{R_1^{k_1}, ..., R_n^{k_n}\}$, una σ-*estructura* finita, $\mathcal{A}$, es una tupla $\langle A, R_1^{\mathcal{A}}, ..., R_n^{\mathcal{A}} \rangle$, donde A es un conjunto finito que llamamos *universo*. Cada $R_i^{\mathcal{A}}$ es una relación en A^{k_i}. Obviaremos subíndices y supraíndices si el contexto permite evitar confusiones. El *tamaño* de $\mathcal{A}$ es $|A|$ (el cardinal de A). A menos que se aclare lo contrario todas las estructuras serán finitas, por lo que las llamaremos simplemente estructuras. Una *subestructura* de una estructura $\mathcal{A}$ es un subconjunto de A, donde cada símbolo de relación $\mathcal{R}$, se interpreta como la correspondiente restricción de $R^{\mathcal{A}}$. Dadas dos estructuras $\mathcal{A}$ y $\mathcal{B}$, un *isomorfismo* entre $\mathcal{A}$ y $\mathcal{B}$ es una función biyectiva $f : A \to B$ tal que para todo $a_1, ..., a_k \in A^k$ y

$R^k \in \sigma$ se cumple

$$R^{\mathcal{A}}(a_1, ..., a_k) \quad \text{sii} \quad R^{\mathcal{B}}(f(a_1), ..., f(a_k)).$$

Naturalmente, $\mathcal{A} \approx \mathcal{B}$ significa que existe un isomorfismo entre $\mathcal{A}$ y $\mathcal{B}$.

Siempre que hablemos de estructuras hay un vocabulario σ implícito, que especificaremos si el contexto es ambiguo. Una *clase* de estructuras es cualquier familia o colección de éstas. Una *propiedad* de estructuras es una clase que a su vez es cerrada por isomorfismos, es decir que si $\mathcal{A} \in \mathcal{P}$ y $\mathcal{A} \approx \mathcal{B}$, entonces $\mathcal{B} \in \mathcal{P}$. Notamos con Struc$[\sigma]$ a la clase de todas las σ-estructuras.

Hablamos de un *alfabeto* Σ, cuando nos referimos a un conjunto de símbolos con el que pretendemos formar *palabras*, tiras finitas de elementos de Σ. Al conjunto de palabras lo notaremos, como es usual, con Σ^*.

A lo largo de la exposición nos referiremos distintas *lógicas*. Todas consisten en un conjunto de fórmulas (que daremos a partir de ciertas reglas de formación, su *sintaxis*) y una regla para asignarle a cada fórmula una propiedad de estructuras (es decir darle significado a la fórmula, la *semántica*). Dar con una definición general de lógica, o sea, qué propiedades queremos que tengan la sintaxis y la semántica no es para nada trivial y depende fuertemente de lo que uno quiera hacer con ella. Si bien en la Sección 2.3 abordaremos (brevemente) la definición de lógica abstracta, siempre que digamos *lógica* tendremos en mente alguna de las definidas. Ebbinghaus [8] hace un análisis profundo de qué propiedades dan pie a qué lógicas.

Para la construcción de fórmulas, es necesario fijar tanto un vocabulario como una lógica. Por lo tanto si $\mathcal{L}$ es la lógica y σ el vocabulario, lo adecuado es hablar de $\mathcal{L}[\sigma]$-fórmulas. Para hacer más agradable la lectura, nos referiremos a σ-fórmulas, $\mathcal{L}$-fórmulas o simplemente fórmulas, dependiendo de en qué querramos hacer hincapié.

Grafos

La clase de estructuras con la que más trabajaremos es la de *grafos*. Si bien no precisaremos de resultados muy sofisticados de la teoría de grafos y daremos todas las definiciones pertinentes, el libro de Bondy [3] es una buena referencia para saldar cualquier duda. Un grafo $\mathcal{G} = \langle G, E^{\mathcal{G}} \rangle$ es una $\{E\}$-estructura, donde E es un símbolo de relación binaria. A los elementos de G los llamamos *vértices* y a cada par en $E^{\mathcal{G}}$, *arista*. La relación $E^{\mathcal{G}}$ es simétrica y antireflexiva (el grafo no tiene bucles). Para referirnos a grafos donde E puede no ser simétrica, hablamos de grafos *dirigidos*. En ese contexto, una *raíz* es un vértice al que no llega ninguna arista. Dados dos vértices, v_1 y v_2, decimos que v_1 es *hijo* de v_2, si vale Ev_2v_1. En ese caso decimos que

v_2 es *padre* de v_1. Una *línea* es un grafo tal que sus vértices pueden ser ordenados de modo que dos son adyacentes si y sólo si son consecutivos. Un *ciclo* es similar a una linea pero además el primer y último vértice también son adyacentes. Un *paseo* en un grafo es una secuencia alternada de vértices y aristas, $v_0e_0v_1e_1...v_k$ tal que v_i y v_{i+1} son los extremos de e_i. Los vértices v_0 y v_k se dicen *extremos* del camino. Un grafo es *conexo* si dado cualquier par de vértices existe un camino que los tiene como extremos. Un *árbol* es un grafo conexo y sin ciclos. Más adelante daremos descripciones lógicas de éstas clases.

Órdenes

Decimos que una relación binaria $\leq$ es un *orden parcial* en un conjunto, si es reflexiva, antisimétrica y transitiva. Si además vale que para todo par de elementos x, y vale $x \leq y$ o $y \leq x$, decimos que es un *orden total*. Si lo notamos con el símbolo $<$, consideramos el orden estricto. Un *preorden* es una relación reflexiva, transitiva y total pero no necesariamente antisimétrica. En general la notamos con $\preceq$. Podemos pensarlo como un orden de clases determinadas por una relación de equivalencia. Si dados a y b, vale que $a \preceq b$ y $b \preceq a$, diremos que a y b son equivalentes para el preorden. Hacemos referencia, a menudo, al orden lexicográfico de tuplas, que notamos con $\leq_{lex}$. Utilizamos $<_{lex}$ cuando consideramos el orden estricto. Dadas dos tuplas distintas $\bar{a}$ y $\bar{b}$, vale $\bar{a} <_{lex} \bar{b}$ si, en el primer lugar donde $\bar{a}$ y $\bar{b}$ difieren, digamos i, vale $a_i < b_i$. El resto de la notación que utilizaremos será introducida a lo largo del trabajo.

Capítulo 1

Lógicas

En este capítulo introducimos las principales lógicas que utilizamos a lo largo del trabajo, junto con herramientas que permiten estudiar su poder expresivo. A lo largo del capítulo, presentamos una serie de ejemplos con el fin de ilustrar su poder expresivo y sus limitaciones.

Comenzamos con la lógica de primer orden, que suponemos conocida para el lector, con el fin de fijar la notación y mostrar, mediante algunos ejemplos, su poder expresivo.

En la Sección 1.2 introducimos los juegos de Ehrenfeucht-Fraïssé, que sirven para mostrar limitaciones en cuanto a la expresividad. Ilustramos esta técnica, con ejemplos de los límites de la lógica de primer orden.

Luego introducimos las lógicas de punto fijo, entre las que destacamos a FO(IFP), pues, como veremos, captura PTIME sobre la clase de estructuras ordenadas.

En la Sección 1.4 repasamos las definiciones básicas de la lógica $\mathcal{L}_{\infty\omega}$, una extensión de la lógica primer orden, que permite conjunciones y disyunciones infinitas. Nos es útil su estudio, pues las lógicas de punto fijo resultan ser un fragmento de ésta.

En la siguiente sección definimos el fragmento existencial de la lógica de segundo orden. Resulta importante, pues, como veremos en el Teorema de Fagin, captura la clase NPTIME.

Por último, introducimos una familia de lógicas que admiten dos tipos de variables, unas que se interpretan sobre conjuntos finitos, y otras que toman valores numéricos, es decir, sobre un segmento inicial de $\mathbb{N}$. Estas lógicas tienen la capacidad de expresar operaciones aritméticas.

A lo largo de este trabajo cada vez que nos refiramos a una lógica $\mathcal{L}$ (a menos que se aclare lo contrario) se estará haciendo referencia a alguna de las ya presentadas.

1.1. Lógica de primer orden

Sintaxis

Para empezar, vale recordar la sintaxis y semántica de la lógica de primer orden (FO), aunque sea sólo para fijar la notación. Dado un vocabulario σ una FO[σ]-fórmula es una expresión en el lenguaje que consiste de:

- los símbolos de σ,
- las variables individuales x_1,x_2,... ,
- los conectivos $\neg$, $\vee$,
- el cuantificador existencial $\exists$,
- los paréntesis),(y
- y el símbolo de igualdad $=$.

Definimos el conjunto de FO-fórmulas inductivamente. Construimos primero las fórmulas *atómicas*.

- Si x e y son variables, la expresión $x = y$ es una fórmula atómica.
- Si $R^k \in \sigma$ y $\bar{x}$ es una tupla de variables de longitud k, $R^k\bar{x}$ es una fórmula atómica.

A partir de éstas el resto de las fórmulas.

- Si φ es una fórmula $\neg\varphi$ lo es.
- Si φ y ψ son fórmulas $(\varphi \vee \psi)$ lo es.
- Si φ es una fórmula y x una variable $\exists x\varphi$ lo es.

Utilizaremos $(\varphi \wedge \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \leftrightarrow \psi)$ y $\forall x\varphi$ para abreviar las expresiones $\neg(\neg\varphi \vee \neg\psi)$, $(\neg\varphi \vee \psi)$, $((\neg\varphi \vee \psi) \wedge (\neg\psi \vee \varphi))$ y $\neg\exists x\neg\varphi$ respectivamente. Escribimos $\exists x_1...x_k$ en lugar de $\exists x_1...\exists x_k$.

Decimos que una aparición de una variable en una fórmula es *libre* si no está cuantificada, caso contrario decimos que está *ligada*. Notamos con $free(\varphi)$ al conjunto de variables que aparecen libres en φ. Tenemos entonces

- $free(x = y) := \{x, y\}$,
- $free(R\bar{x}) := \{\bar{x}\}$,

- $free(\varphi \vee \psi) := free(\varphi) \cup free(\psi)$,
- $free(\neg\varphi) := free(\varphi)$,
- $free(\exists x\varphi) := free(\varphi) - \{x\}$.

Observacion 1.1.1. Una variable en una fórmula puede tener a la vez apariciones libres y ligadas, como es el caso de x en la fórmula:

$$\forall x((y \leq x) \wedge (x = y)).$$

Una *sentencia* es una fórmula sin variables libres. Escribimos $\varphi(x_1, x_2, ..., x_n)$ para indicar que las variables libres de φ se encuentran entre $x_1, x_2, ..., x_n$.

Semántica

Interpretamos las FO-fórmulas sobre estructuras. Utilizaremos *valuaciones* para tratar el caso de variables libres. Dada una estructura $\mathcal{A}$, una valuación ν, es una función que asocia a cada variable x un elemento del universo A. Con $\nu[\frac{x}{a}]$ sos referimos a la valuación que coincide con ν salvo, quizás, en x donde vale a. Con $(\mathcal{A}, a_1, ..., a_k)$ nos referimos a la estructura $\mathcal{A}$ y alguna valuación ν tal que $\nu(x_i) = a_i$. Definimos la relación $\models$ entre valuaciones y fórmulas inductivamente.

- $(\mathcal{A}, \nu) \models x_i = x_j$ sii $\nu(x_i) = \nu(x_j)$.
- $(\mathcal{A}, \nu) \models R\bar{x}$ sii $\nu(\bar{x}) \in R^{\mathcal{A}}$.
- $(\mathcal{A}, \nu) \models \psi_1 \vee \psi_2$ sii $(\mathcal{A}, \nu) \models \psi_1$ o $\mathcal{A} \models \psi_2$.
- $(\mathcal{A}, \nu) \models \neg\varphi$ sii no $(\mathcal{A}, \nu) \models \varphi$.
- $(\mathcal{A}, \nu) \models \exists x\varphi$ sii para algún $a \in A$ vale que $(\mathcal{A}, \nu[\frac{x}{a}]) \models \varphi$.

En general notaremos con $\mathcal{A} \models \varphi(\bar{a})$ a $(\mathcal{A}, \nu) \models \varphi(\bar{x})$, donde $\bar{a} = \nu(\bar{x})$. Decimos que una sentencia φ *define* una clase de estructuras $\mathcal{K}$ si

$$\mathcal{A} \models \varphi \quad \text{sii} \quad \mathcal{A} \in \mathcal{K}.$$

En ese caso decimos que $\mathcal{K}$ es *definible*. Dada φ con variables libres $x_1, ..., x_k$ la *relación definida* por φ es

$$R := \{(a_1, ..., a_k) \in A \mid \mathcal{A} \models \varphi(a_1, ..., a_k)\}.$$

En este caso decimos que R es *definible*. Una fórmula φ *define una relación sobre* $\mathcal{K}$ si la define sobre cada estructura de $\mathcal{K}$. Usamos las abreviaturas $\top$ (true) y $\bot$ (false) para $x = x$ y $\neg x = x$ respectivamente.

Ejemplo 1.1.2 (*Grafos*). *Tomando $\sigma = \{E\}$, con E una relación binaria, podemos definir la clase de grafos mediante la fórmula:*

$$\varphi_{grafo} := \forall x \neg E(x,x) \wedge \forall x \forall y (E(x,y) \rightarrow E(y,x)).$$

Ejemplo 1.1.3 (*Orden*). *Sea σ con $\leq \in \sigma$. La clase de estructuras* ordenadas *(por $\leq$) es la definida por la conjunción de las siguientes fórmulas, que notamos con φ_{ord}.*

$$\begin{aligned} \forall xyz(((x \leq y) \wedge (y \leq z)) \rightarrow (x \leq z)) &\quad \text{(transitiva)} \\ \forall xy(((x \leq y) \wedge (y \leq x)) \rightarrow (x = y)) &\quad \text{(antisimétrica)} \\ \forall x(x \leq x) &\quad \text{(refelxiva)} \\ \forall xy((x \leq y) \vee (y \leq x)) &\quad \text{(total)} \end{aligned}$$

Siempre que tengamos estructuras ordenadas por $\leq$, nos reservamos el símbolo $<$ para expresar el orden estricto. Con $x < y$ notamos $(x \leq y) \wedge (x \neq y)$.

Ejemplo 1.1.4 (*Sucesor*). *Sea σ igual que en el ejemplo anterior. Podemos definir la relación de sucesor sobre la clase de estructuras ordenadas mediante:*

$$\varphi_{suc}(x,y) := (x < y) \wedge \forall z((x \leq z) \rightarrow (y \leq z)).$$

Si dos estructuras $\mathcal{A}$ y $\mathcal{B}$ cumplen las mismas sentencias de primer orden, decimos que son *elementalmente equivalentes* y lo notamos $\mathcal{A} \equiv \mathcal{B}$.

Proposicion 1.1.5 (Teorema 1.1.10 en [23]). *Si $\mathcal{A} \approx \mathcal{B}$ entonces $\mathcal{A} \equiv \mathcal{B}$. Es más, si f es un isomorfismo, para toda fórmula $\varphi(x_1, ..., x_k)$, vale*

$$\mathcal{A} \models \varphi(a_1, ..., a_k) \quad sii \quad \mathcal{B} \models \varphi(f(a_1), ..., f(a_k)).$$

Podemos pensar en ésto como que $\models$ "respeta" isomorfismos. En general usamos la notación $\mathcal{A} \equiv^{\mathcal{L}} \mathcal{B}$, para decir que $\mathcal{A}$ y $\mathcal{B}$ satisfacen las mismas $\mathcal{L}$-sentencias. La siguiente proposición muestra como FO caracteriza (módulo isomorfismos) completamente a una estructura finita.

Proposicion 1.1.6. *Dada una estructura $\mathcal{A}$ existe una FO-fórmula $\varphi_{\mathcal{A}}$ tal que para toda estructura $\mathcal{B}$ vale*

$$\mathcal{B} \models \varphi_{\mathcal{A}} \quad sii \quad \mathcal{B} \approx \mathcal{A}.$$

Demostración. Dada $\mathcal{A}$, supongamos $A = \{a_1, ..., a_n\}$, consideremos para cada $R \in \sigma$ de aridad k, la fórmula

$$\varphi^R_{\mathcal{A}}(x_1, ..., x_k) := \bigwedge_{(a_{i_1},...,a_{i_k}) \in R^{\mathcal{A}}} Rx_{i_1}...x_{i_k} \bigwedge_{(a_{i_1},...,a_{i_k}) \notin R^{\mathcal{A}}} \neg Rx_{i_1}...x_{i_k}.$$

Luego,

$$\varphi_{\mathcal{A}} := \exists x_1...x_n((\bigwedge_{i \neq j}(x_i \neq x_j))(\bigwedge_{R \in \sigma} \varphi^R_{\mathcal{A}}(x_1, ..., x_k)) \wedge \forall y(\bigvee_{1 \leq i \leq n} y = x_i))$$

es la fórmula que buscábamos. □

Corolario 1.1.7. *Si $\mathcal{A}$ y $\mathcal{B}$ son estructuras, entonces $\mathcal{A} \equiv \mathcal{B}$ implica $\mathcal{A} \approx \mathcal{B}$.*

Observacion 1.1.8. El resultado anterior no es ciertos si consideramos estructuras infinitas.

1.2. Juegos de Ehrenfeucht-Fraïssé

Los *juegos* son una valiosa herramienta a la hora de estudiar los límites en la capacidad expresiva de una lógica. En particular, para probar resultados de expresabilidad o inexpresividad. Cobran un valor aún mayor cuando nos restringimos a estructuras finitas, donde otros recursos como el *Teorema de Compacidad*, la *Propiedad de Beth* y el *Teorema de Interpolación* no pueden utilizarse (ver [9]). Los juegos fueron introducidos por Eherenfeucht [10], a Fraïssé se debe la formulación algebraica del mismo resultado, diez años anterior [12].

1.2.1. Definiciones previas

Llamamos *rango cuantificacional* de una FO-fórmula φ, $qr(\varphi)$, a la cantidad de cuantificadores anidados que aparecen en φ. Lo definimos inductivamente.

- $qr(\varphi) = 0$ si φ es atómica.
- $qr(\neg\varphi) = qr(\varphi)$.
- $qr(\varphi \vee \psi) = max\{qr(\varphi), qr(\psi)\}$.
- $qr(\exists x\varphi) = qr(\varphi) + 1$.

Dado $m \in \mathbb{N}$, decimos $\mathcal{A}$ y $\mathcal{B}$ son *m-equivalentes* y lo notamos $\mathcal{A} \equiv_m \mathcal{B}$, si satisfacen las mismas sentencias de rango cuantificacional menor igual a m.

Definicion 1.2.1. Dadas dos estructuras $\mathcal{A}$ y $\mathcal{B}$, un *isomorfismo parcial* entre $\mathcal{A}$ y $\mathcal{B}$ es un una función inyectiva f, con dominio un subconjunto de A, tal que para todo $a_1, ..., a_k$ en el dominio de f y todo $R \in \sigma$ de aridad k, vale que,

$$R^{\mathcal{A}}a_1...a_k \quad \text{sii} \quad R^{\mathcal{B}}b_1...b_k. \tag{1.1}$$

Notamos como $\bar{a} \mapsto \bar{b}$ al isomorfismo parcial f tal que $f(a_i) = b_i$.

1.2.2. El juego

Sean $\mathcal{A}$ y $\mathcal{B}$ estructuras, $\bar{a} \in A^s$ y $\bar{b} \in B^s$, el juego $\mathcal{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$ es jugado por dos jugadores, Spoiler y Duplicator. En cada jugada Spoiler elige una estructura, $\mathcal{A}$ o $\mathcal{B}$ y un elemento en el universo correspondiente. Supongamos que en la jugada i-ésima elije $a_i \in A$. Luego Duplicator debe elegir un elemento $b_i \in B$ de manera que $\bar{a}a_1...a_i \mapsto \bar{b}b_1...b_i$ sea un isomorfismo parcial. Duplicator gana si $\bar{a}a_1...a_m \mapsto \bar{b}b_1...b_m$ resulta un isomorfismo parcial. Caso contrario gana Spoiler. Es decir, Duplicator intenta probar que dos estructuras son indistinguibles, Spoiler que no. Observemos que con esta definición Duplicator gana el juego $\mathcal{G}_0(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$ si y sólo si $\bar{a} \mapsto \bar{b}$ es un isomorfismo parcial, si y sólo si $\bar{a}$ y $\bar{b}$ satisfacen las mismas fórmulas libres de cuantificadores.

Observemos también que si $m > 0$, Duplicator gana $\mathcal{G}_m(\mathcal{A}, \bar{a}a, \mathcal{B}, \bar{b}b)$ si para todo $a \in A$, existe $b \in B$, tal que gana $\mathcal{G}_{m-1}(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$ y para todo $b \in B$, existe un $a \in A$, tal que gana $\mathcal{G}_{m-1}(\mathcal{A}, \bar{a}a, \mathcal{B}, \bar{b}b)$. La observación anterior da la pauta de como relacionar la cantidad de jugadas con el rango cuantificacional de una fórmula. Este hecho se formaliza en la siguiente proposición, obviaremos una demostración de ésta, que puede encontrarse en la literatura. Notamos con $\mathcal{G}_m(\mathcal{A}, \mathcal{B})$ al juego en el caso que $s = 0$.

Proposicion 1.2.2 (Teorema 2.2.8. en [9])**.** *Son equivalentes:*

(I) Duplicator *gana el juego* $\mathcal{G}_m(\mathcal{A}, \mathcal{B})$.

(II) $\mathcal{A} \equiv_m \mathcal{B}$.

El siguiente corolario es una consecuencia de la caracterización de los juegos mediante lógicas. Es un caso particular de lo que en teoría de juegos se conoce como composición de estrategias.

Corolario 1.2.3. *Si* Duplicator *gana el juego* $\mathcal{G}_m(\mathcal{A},\mathcal{B})$ *y el juego* $\mathcal{G}_m(\mathcal{B},\mathcal{C})$*, entonces gana* $\mathcal{G}_m(\mathcal{A},\mathcal{C})$*.*

Demostración. Basta notar que $\mathcal{A} \equiv_m \mathcal{B}$ y $\mathcal{B} \equiv_m \mathcal{C}$ implica $\mathcal{A} \equiv_m \mathcal{C}$. □

Proposicion 1.2.4. *Sea* $\mathcal{P}$ *una propiedad de estructuras. Si para todo* $m \in \mathbb{N}$*, existen* $\mathcal{A}_m \in \mathcal{P}$ *y* $\mathcal{B}_m \notin \mathcal{P}$ *tal que* $\mathcal{A}_m \equiv_m \mathcal{B}_m$*, entones* $\mathcal{P}$ *no es definible por una* FO*-fórmula.*

Demostración. Sea φ la fórmula que define a $\mathcal{P}$, supongamos $rq(\varphi) = k$. Luego $\mathcal{A} \models \varphi$ para todo $\mathcal{A} \in \mathcal{P}$ y $\mathcal{B} \models \neg\varphi$ para todo $\mathcal{B} \notin \mathcal{P}$. Si tomamos, para un $m > k$, $\mathcal{A}_m$ y $\mathcal{B}_m$ como en el enunciado, tenemos que:

$$\mathcal{A}_m \models \varphi \quad \text{sii} \quad \mathcal{B}_m \models \varphi,$$

lo que es absurdo. □

Los siguientes ejemplos muestran como pueden aplicarse estos juegos para mostrar que ciertas clases no son definibles en FO.

Ejemplo 1.2.5 (Paridad)**.** *Fijemos* $\sigma = \emptyset$*. Dadas dos* σ*-estructuras* $\mathcal{A}$ *y* $\mathcal{B}$ *(conjuntos), cualquier función inyectiva de* A *a* B *resulta un isomorfismo parcial. Por lo que, en cada jugada, elegir cualquier elemento (que no haya sido elegido aún) de la estructura correspondiente, es una estrategia ganadora para* Duplicator*. Sea* $m \in \mathbb{N}$*. Si* $|A|, |B| \geq m$*,* Duplicator *gana* $\mathcal{G}_m(\mathcal{A},\mathcal{B})$*. Luego* $\mathcal{A} \equiv_m \mathcal{B}$*. Aplicando la Proposición 1.2.4, la clase de conjuntos de cardinal par no es definible en* FO*.*

Ejemplo 1.2.6 (Paridad en estructuras ordenadas)**.** *Sea* $\sigma = \{\leq\}$ *y* $\mathcal{K}$ *la clase de estructuras ordenadas. Un isomorfismo parcial entre estructuras de* $\mathcal{K}$ *es una función inyectiva que respeta el orden (la única relación en* σ*). Veamos que si* $|A|, |B| \geq 2^m$ *entonces* $\mathcal{A} \equiv_m \mathcal{B}$*. Por lo que la paridad tampoco es* FO*-definible sobre la clase de órdenes.*

Dada $\mathcal{A} \in \mathcal{K}$*, definimos la función* distancia $d : A \times A \to \mathbb{N}$ *como*

$$d(a,b) := |\{a' \in A \mid (a < a' \leq b) \; o \; (b < a' \leq a)\}|. \tag{1.2}$$

Para cada $j \in \mathbb{N}$ *la distancia truncada*

$$d_j(a,b) := \begin{cases} d(a,b) & si \quad d(a,b) < 2^j \\ \infty & caso\ contrario \end{cases} \tag{1.3}$$

Sean $m \in \mathbb{N}$ *y* $\mathcal{A},\mathcal{B} \in \mathcal{K}$ *con* $|A|, |B| \geq 2^m$*. Vamos a dar una estrategia para* Duplicator *en* $\mathcal{G}_m(\mathcal{A},\mathcal{B})$*. Consiste en construir el isomorfismo parcial*

teniendo en cuenta la distancia. Para la primera jugada, supongamos que S elije $a \in A$. Existen dos posibilidades, o bien a está "cerca" de algún extremo, que notamos con min o max, es decir

$$d(a, min) < 2^{m-1} \quad o \quad d(a, max) < 2^{m-1},$$

o bien,

$$d(a, min) = d(a, max) = \infty.$$

En el primer caso Duplicator *responde con (el único) $b \in B$ a igual distancia. En el segundo caso* Duplicator *responde con algún $b \in B$ a distancia ∞ de los extremos, que existe por la suposición en el tamaño de las estructuras. Supongamos que el isomorfismo parcial construido hasta la jugada i-ésima respeta d_{m-i} y* Spoiler *eljie a_{i+1} en A. Si para algún a' jugado anteriormente,*

$$d(a, a') < 2^{m-i}$$

hay un único $b \in B$ a esa distancia de b' de manera tal que el orden se respete. Esa es la jugada de Duplicator*. Si, en cambio, para todo a'*

$$d_{m-i}(a, a') = \infty,$$

supongamos $a_j < a < a_{j+1}$ con a_j y a_{j+1} ya jugados. Como

$$d_{m-i}(a, a_j) = d_{m-i}(a, a_j + 1) = \infty, \tag{1.4}$$

debe ser $d_{m-(i-1)}(a_j, a_{j+1}) = \infty$. Por lo que $d_{m-(i-1)}(b_j, b_{j+1}) = \infty$. Luego existe $b \in B$ tal que

$$b_j < b < b_{j+1}$$
$$d_{m-i}(b, b_j) = d_{m-i}(b, b_j + 1) = \infty.$$

Esa es la jugada de Duplicator*.*

Ejemplo 1.2.7 (Conexión)**.** *Sean $\sigma = \{E\}$ y $\mathcal{K}$ la clase de grafos. Sea $\mathcal{C}_k$ el* ciclo *de k vértices, es decir, $\mathcal{C}_k$ tiene como universo al segmento $[0, k-1]$ e interpretamos la relación de adyacencia como*

$$E^{\mathcal{C}_k} := \{(i, i+1), (i+1, i), (0, k-1), (k-1, 0) \mid i \leq k-1\}.$$

Podemos adaptar el ejemplo anterior considerando la función d como la distancia en el grafo, es decir, el mínimo n tal que hay un camino de longitud n entre a y b, si no hay ningún camino fijamos d como ∞. Definimos la distancia truncada d_j, igual que en el ejemplo anterior.
La misma estrategia sirve para Duplicator*,obviando el orden, para ganar el juego $\mathcal{G}_m(\mathcal{C}_k, \mathcal{C}_k \sqcup \mathcal{C}_k)$ con $k > 2^m$. Por lo tanto la clase de grafos conexos tampoco es definible en* FO*. Tampoco lo es la clausura transitiva de la relación de adyacencia, pues la conexión es facilmente definible a partir de ésta.*

1.2.3. Comparación de lógicas

A continuación definiremos distintas lógicas, que extienden a FO agregando fórmulas que permiten definir propiedades que FO no puede. Si $\mathcal{L}_1$ y $\mathcal{L}_2$ son dos lógicas, notamos con $\mathcal{L}_1 \leqq \mathcal{L}_2$, que $\mathcal{L}_1$ es *más expresiva* que $\mathcal{L}_2$. Es decir, dada una $\mathcal{L}_1$-fórmula φ, existe una $\mathcal{L}_2$-fórmula ψ, tal que para toda estructura $\mathcal{A}$ y valuación ν,

$$(\mathcal{A}, \nu) \models_{\mathcal{L}_1} \varphi \quad \text{sii} \quad (\mathcal{A}, \nu) \models_{\mathcal{L}_2} \psi.$$

En ese caso decimos que φ y ψ son *equivalentes*. Notamos con $\mathcal{L}_1 \lneqq \mathcal{L}_2$ cuando vale $\mathcal{L}_1 \leqq \mathcal{L}_2$ y no vale $\mathcal{L}_2 \leqq \mathcal{L}_1$. Notamos $\mathcal{L}_1 \equiv \mathcal{L}_2$ cuando valen ambas desigualdades. En ese caso decimos que $\mathcal{L}_1$ y $\mathcal{L}_2$ son equivalentes.

1.3. Lógicas de punto fijo

Una de las falencias de la lógica de primer orden es su incapacidad para expresar recurrencias. Esto impide, por ejemplo, definir la clausura transitiva de una relación. Se puede salvar esta falencia extendiendo a FO con operadores de punto fijo, que permiten, justamente, definir relaciones vía recurrencias. Antes de definir la sintaxis y semántica de estas lógicas precisamos de algunas definiciones previas.

1.3.1. Operadores de punto fijo

Definicion 1.3.1. Sea B un conjunto finito y $\mathcal{F} : \mathcal{P}(B) \to \mathcal{P}(B)$ un operador.
Consideremos la sucesión:

$$\mathcal{F}_0 = \emptyset,$$
$$\mathcal{F}_n = \mathcal{F}(\mathcal{F}_{n-1}).$$

Decimos que $\mathcal{F}$ tiene un *punto fijo*, si para algún n_0, $\mathcal{F}_{n_0} = \mathcal{F}_{n_0+1}$. En ese caso definimos $\mathcal{F}_\infty := \mathcal{F}_{n_0}$ y lo llamamos punto fijo de $\mathcal{F}$. Si esto no ocurre definimos $\mathcal{F}_\infty := \emptyset$.

Si para todo $X \in \mathcal{P}(B)$ vale que $X \subset \mathcal{F}(X)$, decimos que $\mathcal{F}$ es un *operador inflacionario*. En ese caso para todo n vale que $\mathcal{F}_n \subset \mathcal{F}_{n+1}$.

Proposicion 1.3.2. *Valen las siguientes afirmaciones:*

(a) Si $\mathcal{F}$ tiene un punto fijo $\mathcal{F}_\infty = \mathcal{F}_{2^{|B|}-1}$.

(b) Si $\mathcal{F}$ es inflacionario tiene un punto fijo y $\mathcal{F}_\infty = \mathcal{F}_{|B|}$.

Demostración. (a) Como $|\mathcal{P}(B)| = 2^{|B|}$ debe ser $\mathcal{F}_{2^{|B|}} = \mathcal{F}_n$ para algún $n < 2^{|B|}$, entonces $\mathcal{F}_{2^{|B|}} = \mathcal{F}_\infty$. Si, en cambio, $\mathcal{F}_{2^{|B|}} \neq \mathcal{F}_{2^{|B|}-1}$ el operador oscila.

(b) Como $\mathcal{F}_0 \subset \mathcal{F}_1 \subset ... \subset B$ la sucesión debe estancarse en no más de $|B|$ pasos. □

Observacion 1.3.3. Dado un operador cualquiera $\mathcal{F}$ el operador $\widehat{\mathcal{F}}(X) := \mathcal{F}(X) \cup X$ resulta inflacionario.

Sea $\varphi(\bar{x}, X)$ una fórmula en el vocabulario σ, donde X es una variable de relación con aridad la longitud de $\bar{x}$ (llamémosla k). Consideremos una σ-estrucutra $\mathcal{A}$. Podemos definir el siguiente operador sobre $\mathcal{P}(A^k)$:

$$\mathcal{F}^\varphi(R) := \{\bar{a} \in A^k \mid \mathcal{A} \models \varphi(\bar{a}, R)\}.$$

Ejemplo 1.3.4 (Clausura transitiva)**.** *Sean $\sigma = \{E\}$ y $\mathcal{G} = \langle G, E^\mathcal{G}\rangle$ un grafo. Si tomamos*

$$\varphi(x, y, X) := Exy \vee (\exists z(Xxz \wedge Xzy)),$$

resulta: $\mathcal{F}_0^\varphi = \emptyset$, $\mathcal{F}_1^\varphi = E^\mathcal{G}$, luego

$$\mathcal{F}_2^\varphi = \{(a, b) \mid Eab \vee (\exists z Eaz \wedge Ezb)\},$$

es decir, los pares (a,b) tales que existe un camino de longitud a lo sumo 2 entre a y b. Siguiendo,

$$\mathcal{F}_n^\varphi = \{(a, b) \mid \textit{existe un camino de lo longitud a lo sumo n entre a y b}\}.$$

Luego $\mathcal{F}_\infty^\varphi$ resulta la clausura transitiva de la relación E.

1.3.2. FO(IFP) y FO(PFP)

Podemos extener a FO mediante operadores de punto fijo obteniendo FO(PFP). Por otra parte FO(IFP) será el fragmento de ésta que resulta de restringirse a operadores inflacionarios.

Sintaxis

Construiremos FO(IFP)- y FO(PFP)-fórmulas de manera similar. Además de variables de individuales, el lenguaje tiene para cada $k \in \mathbb{N}$ variables de relación k-arias $X_1^k, X_2^k, ..., X_2^k$. Extendemos al conjunto de fórmulas de primer orden agregando las del tipo $X^k x_1 x_2 ... x_k$ y una nueva regla de formación

para el operador de punto fijo. Si X^k es una variable de relación, $\bar{x}$ y $\bar{t}$ tuplas de variables individuales de longitud k y φ una fórmula, entonces:

$$\mathsf{IFP}\,[X\bar{x} \leftarrow \varphi]\,(\bar{t}),$$
$$\mathsf{PFP}\,[X\bar{x} \leftarrow \varphi]\,(\bar{t}).$$

son $\mathsf{FO}(\mathsf{IFP})$ y $\mathsf{FO}(\mathsf{PFP})$ fórmulas respectivamente. Observemos que, con las reglas definidas, sólo las variables individuales pueden aparecer cuantificadas.

Las variables libres las definimos como:

- $free(X\bar{x}) := \{\bar{x}, X\}$,
- $free(\mathsf{IFP}\,[X\bar{x} \leftarrow \varphi]\,(\bar{t})) := (free(\varphi) - \{X, \bar{x}\}) \cup \{\bar{t}\}$.

Semántica

Interpretamos las $\mathsf{FO}(\mathsf{IFP})$ o $\mathsf{FO}(\mathsf{PFP})$ fórmulas en estructuras, donde ahora una valuación ν no sólo asigna elementos del universo a variables individuales, si no también, una relacion de aridad k a cada variable de relación X^k. La relación $\models$ la definimos inductivamente igual que en el caso de FO, agregando los siguientes casos:

- $(\mathcal{A}, \nu) \models X\bar{x} \quad sii \quad \nu(\bar{x}) \in \nu(X)$.
- $(\mathcal{A}, \nu) \models \mathsf{IFP}\,[X\bar{x} \leftarrow \varphi]\,(\bar{t}) \quad sii \quad \nu(\bar{t}) \in \mathcal{F}_\infty^{X\bar{x}\vee\varphi}$.
- $(\mathcal{A}, \nu) \models \mathsf{PFP}\,[X\bar{x} \leftarrow \varphi]\,(\bar{t}) \quad sii \quad \nu(\bar{t}) \in \mathcal{F}_\infty^{\varphi}$.

Observemos que el operador $\mathcal{F}^{X\bar{x}\vee\varphi}$ resulta inflacionario cualquiera sea φ.

Ejemplo 1.3.5 (Grafos conexos en $\mathsf{FO}(\mathsf{IFP})$)**.** *Con la misma idea que en el Ejemplo 1.3.4, la fórmula:*

$$\mathsf{IFP}\,[Xxy \leftarrow Exy \vee (\exists z Xxz \wedge Xzy)]\,(s,t),$$

define la clausura transitiva de E. Luego,

$$\varphi_{con} := \forall st\, \mathsf{IFP}\,[Xxy \leftarrow Exy \vee (\exists z Xxz \wedge Xzy)]\,(s,t),$$

define la clase de grafos conexos, lo que prueba que $\mathsf{FO} \lneqq \mathsf{FO}(\mathsf{IFP})$.

Ejemplo 1.3.6 (Grafos acíclicos)**.** *Tomando nuevamente el vocabulario de los grafos, consideremos la siguiente fórmula:*

$$\varphi := \forall y(E(y,x) \to Xy).$$

Dado un grafo dirigido $\mathcal{G}$ y $X \subset G$, $\mathcal{F}^{\varphi}(X)$ es el conjunto de vértices $a \in G$, tales que todo vertice $b \in G$ desde el que hay un paseo hacia a está en X. Luego $\mathcal{F}_1^{\varphi}$ será el conjunto de raices, $\mathcal{F}_2^{\varphi}$ el de vértices a los que sólo llegan aristas desde raices. Iterando este razonamiento, $\mathcal{F}_{\infty}^{\varphi}$ resulta el conjunto de vértices tales que todos los paseos que terminan en ellos son finitos. Por lo que la FO(IFP)*-fórmula*

$$\varphi_{aciclico} := \forall t\, \mathsf{IFP}\,[Xx \leftarrow \varphi]\,(t),$$

expresa que el grafo (dirigido) es acíclico. Observar que en el caso de grafos no dirigidos, expresa que todos sus vértices son aislados.

Ejemplo 1.3.7 (Árboles dirigidos)**.** *Un grafo dirigido es un árbol si es acíclico y:*

(I) hay un único vértice sin predecesores (la raíz),

(II) todo vértice tiene a lo sumo un predecesor.

Notemos que la conexión se deduce de las condiciones anteriores (porque no hay ciclos). La conjunción de las siguientes FO(IFP)*-fórmulas define a la clase de árboles dirigidos.*

(I) $\varphi_{aciclico}$*,*

(II) $(\exists x \forall y \neg Eyx) \wedge \forall xy((\forall z \neg Ezx \wedge \forall z \neg Ezy) \to x = y)$*,*

(III) $\forall xyz((Eyx \wedge Ezx) \to y = z)$*.*

Ejemplo 1.3.8 (Aritmética en FO(IFP))**.** *Otro ejemplo del poder expresivo de* FO(IFP) *es su capacidad para definir operaciones aritméticas. Fijemos el vocabulario $\sigma = \{\leq, S\}$ y estructuras ordenadas donde S denota la relación de sucesor, que podemos definir a partir de $\leq$ mediante una* FO*-fórmula (Ejemplo 1.1.4). Observemos que dada una estructura $\mathcal{A}$, cada $a \in A$ es definible mediante una fórmula. Con $min(x)$ definimos al primero y si $\varphi(x)$ define un elemento $\exists y(\varphi(y) \wedge Syx)$ define al siguiente. Mediante esta identificación podemos pensar a estas estructuras como segmentos iniciales de $\mathbb{N}$.*

Definamos ahora $+ := \{(i,j,k) \mid i+j=k\}$ $y \times := \{(i,j,k) \mid i \times j = k\}$. *En el caso de la suma, podemos definirla por recurrencia mediante:*

$$\begin{aligned} 0 + j &= j \\ S(i) + j &= S(i+j). \end{aligned}$$

Traduciendo esta recursion a FO(IFP)*, resulta la fórmula:*

$$\varphi_+(ijk) = \mathsf{IFP}\,[Xxyz \leftarrow \varphi(X,x,y,z)]\,(ijk),$$

donde

$$\varphi := ((min(x)) \wedge (y=z)) \vee (\exists uv(Sux \wedge Svz \wedge Xuyv)).$$

La multiplicación podemos definirla recursivamente a partir de la suma de la siguiente manera:

$$\begin{aligned} 0 \times j &= 0 \\ (i+1) \times j &= i \times j + j. \end{aligned}$$

Se traduce en la FO(IFP)*-fórmula:*

$$\varphi_\times = \mathsf{IFP}\,[Xxyz \leftarrow \varphi]\,(ijk),$$

donde

$$\varphi := (min(x) \wedge min(z)) \vee (\exists uv(Sux \wedge \varphi_+(vyz) \wedge Xuyv))$$

Nota. La formulación original de muchos de resultados que veremos, no es en términos de FO(IFP) si no de la lógica *Least fixed-point logic* (FO(LFP))(ver [9] para una definición). Como éstas resultan equivalentes (ver [17]), la sintaxis de FO(IFP) es más simple y se adapta mejor a las extensiones que haremos más adelante, elegimos esta última.

1.4. Lógicas con fórmulas infinitas

Para poder analizar el poder expresivo de FO(IFP) y FO(PFP) resulta útil estudiar lógicas que permiten conjunciones o disyunciones infinitas, pues las primeras son un fragmento natural de éstas. Introducimos $\mathcal{L}_{\infty\omega}$ extendiendo a FO con las siguientes reglas de formación de fórmulas. Si Ψ es un conjunto (arbitrario) de $\mathcal{L}_{\infty\omega}$-fórmulas, entonces

- $\bigwedge \Psi$,

- $\bigvee \Psi$,

son $\mathcal{L}_{\infty\omega}$-fórmulas. Si Ψ está indexado por un conjunto de índices Λ, notaremos $\bigwedge_{\alpha\in\Lambda} \psi_\alpha$ en lugar de $\bigwedge \Psi$. La semántica es la natural, es decir,

- $(\mathcal{A}, \nu) \models \bigwedge \Psi$ sii $(\mathcal{A}, \nu) \models \psi$ para cada $\psi \in \Psi$.
- $(\mathcal{A}, \nu) \models \bigvee \Psi$ sii $(\mathcal{A}, \nu) \models \psi$ para alguna $\psi \in \Psi$.

Ejemplo 1.4.1 (Poder expresivo de $\mathcal{L}_{\infty\omega}$)**.** *Sea $\varphi_\mathcal{A}$ la FO-fórmula que para cada σ-estructura $\mathcal{A}$, caracteriza a las estructuras isomorfas a $\mathcal{A}$. Dada una propiedad cualquiera de σ-estructuras $\mathcal{P}$ la $\mathcal{L}_{\infty\omega}$-fórmula*

$$\bigvee_{\mathcal{A}\in\mathcal{P}} \varphi_\mathcal{A} \tag{1.5}$$

define a $\mathcal{P}$.

El ejemplo anterior muestra los alcances del poder expresivo de $\mathcal{L}_{\infty\omega}$. En particular, si tomamos $W \subset \mathbb{N}$ un conjunto cualquiera de números, la clase de conjuntos cuyo cardinal está en W es definible, incluso aunque W no sea computable.

Observemos que la fórmula $\varphi_\mathcal{A}$, requiere en general tantas variables como el tamaño de A (en realidad una más). Por lo que la fórmula 1.5 podría utilizar infinitas variables. Esto sugiere estudiar el fragmento de $\mathcal{L}_{\infty\omega}$ que surge al restringirse a fórmulas que usan sólo finitas variables.

1.4.1. Finitas variables

Notamos con $\mathcal{L}^k_{\infty\omega}$ al fragmento de $\mathcal{L}_{\infty\omega}$ que consiste de las fórmulas con hasta k variables, digamos $x_1...x_k$. La lógica FO^k es el fragmento análogo de FO. Definimos

$$\mathcal{L}^\omega_{\infty\omega} := \bigcup_{k\in\mathbb{N}} \mathcal{L}^k_{\infty\omega}. \tag{1.6}$$

Veamos algunos ejemplos.

Ejemplo 1.4.2 (Clausura transitiva en $\mathcal{L}^3_{\infty\omega}$)**.** *Sea $\sigma = \{E\}$ con E binaria. La fórmula*

$$\varphi_1(x, y) := (x = y) \vee Exy, \tag{1.7}$$

describe los pares que distan a lo sumo uno. Inductivamente,

$$\varphi_{i+1} := \exists z(\varphi_i(x, z) \wedge Ezy) \tag{1.8}$$

describe a los pares que distan a lo sumo $i+1$. Luego, podemos definir la clausura transitiva de E mediante la $\mathcal{L}^3_{\infty\omega}$-fórmula

$$\psi(x,y) := \bigvee_{i\geq 1} \varphi_i. \tag{1.9}$$

Ejemplo 1.4.3 (Poder expresivo de $\mathcal{L}^2_{\infty\omega}$)**.** *En el caso de $\sigma = \{\leq\}$ y la clase de conjuntos totalmente ordenados, con dos variables alcanza para definir cualquier elemento.*

$$\varphi_0(x) := \forall y(y \leq x) \tag{1.10}$$

define al primero e inductivamente,

$$\varphi_{i+1}(x) := \left(\bigwedge_{j\leq i} \neg\varphi_j(x)\right) \wedge \forall y(y < x \to \bigvee_{j\leq i} \varphi(y)),$$

define al $i+1$-ésimo.

Consideremos el lenguaje $\sigma := \{\leq, E\}$ con E binaria. Una σ-estructura ordenada $\mathcal{A}$ puede pensarse como un grafo sobre un segmento incial de $\mathbb{N}$. La siguiente FO^2 fórmula describe la clase de estructuras ordenadas isomorfas a $\mathcal{A}$.

$$\varphi_{\mathcal{A}} := \varphi_{ord} \wedge \bigwedge_{(i,j)\in E^{\mathcal{A}}} (\exists xy(\varphi_i(x) \wedge \varphi_j(y) \wedge Exy)) \\ \bigwedge_{(i,j)\notin E^{\mathcal{A}}} (\exists xy(\varphi_i(x) \wedge \varphi_j(y) \wedge \neg Exy))$$

Si repetimos la construcción del Ejemplo 1.4.1 concluimos que toda propiedad de σ-estructuras es definible en $\mathcal{L}^2_{\infty\omega}$.

Es decir que, sobre conjuntos ordenados, $\mathcal{L}^2_{\infty\omega}$ también tiene un gran poder expresivo.

Los Ejemplos 1.3.4 y 1.4.2 muestran cómo puede simularse una $\mathsf{FO}(\mathsf{IFP})$-fórmula mediante una conjunción o disyunción infinita. Formalizamos esto en la siguiente proposición.

Proposicion 1.4.4. *Para toda $\mathsf{FO}(\mathsf{PFP})$-fórmula φ existe una $\mathcal{L}^\omega_{\infty\omega}$-fórmula equivalente ψ.*

Demostración. Supongamos $\varphi(\bar{t}) = \mathsf{PFP}\,[X\bar{x} \leftarrow \tilde{\varphi}(X,\bar{x})]\,(\bar{t})$. Con $\tilde{\varphi}$ sin apariciones del operador PFP. Supongamos también que φ además de $x_1 ... x_k$

usa variables $z_1 ... z_l$. Sean $y_1, ..., y_k$ variables frescas. Consideremos el operador $\mathcal{F}^{\varphi}$. La relación definida por el operador en el paso i es definible mediante una FO-fórmula φ_i de la siguiente manera.

$$\varphi_0(\bar{x}) := \bot \tag{1.11}$$

y definimos $\varphi_{i+1}(\bar{x})$ a partir de $\tilde{\varphi}(X, \bar{x})$, reemplazando cada aparición de $Xu_1, ..., u_k$, donde $\{u_1, ..., u_k\} \subset \{x_1, ..., x_k, z_1, ..., z_l\}$, por:

$$\exists \bar{y}((\bar{y} = \bar{u}) \wedge \exists \bar{x}(\bar{x} = \bar{y} \wedge \varphi_i(\bar{x})). \tag{1.12}$$

Utilizamos $\bar{y}$ como variables auxiliares, para evitar que alguna variable que debería estar libre aparezca cuantificada en φ_i. Finalmente ψ resulta

$$\psi(\bar{t}) := \bigvee_{i \geq 0} (\forall \bar{x}(\varphi_i(\bar{x}) \leftrightarrow \varphi_{i+1}(\bar{x})) \wedge \varphi_i(\bar{t})). \tag{1.13}$$

La subfórmula $\forall \bar{x}(\varphi_i(\bar{x}) \leftrightarrow \varphi_{i+1}(\bar{x})$ indica que ya se alcanzó el punto fijo. Notemos que, si $\tilde{\varphi}$ utilizaba m variables, ψ utiliza, a lo sumo, $2m$.

Para extender el resultado a FO(PFP)-fórmulas con puntos fijos anidados, basta observar que en la construcción la fórmula 1.12 no es necesario pedir que φ sea de primer orden (es decir finita), sólo que use finitas variables. Por lo que podemos repetir el argumento inductivamente. □

Corolario 1.4.5. $\mathsf{FO(IFP)} \leqq \mathsf{FO(PFP)} \leqq \mathcal{L}^{\omega}_{\infty\omega}$.

El corolario anterior nos permite estudiar los límites en el poder expresivo de FO(IFP) y FO(PFP) a partir de los de $\mathcal{L}^{k}_{\infty\omega}$. Esta última puede ser caracterizada mediante una modificación de los juegos definidos en la Sección 1.2, como veremos a continuación.

1.4.2. Juegos con fichas

Introducimos una modificación a los juegos de Erehnfeucht-Fraïssé para adaptarlos a lógicas con finitas variables. Definimos el juego $\mathcal{G}^{k}_{m}(\mathcal{A}, \mathcal{B})$, el juego en m pasos con k fichas, de manera similar a $\mathcal{G}$. Los jugadores son los mismos, Duplicator y Spoiler. Hay k pares de fichas numerados de 1 a k. En cada jugada cada ficha puede estar asociada a algun elemento de A o B o estar *libre*. Usamos el símbolo distinguido $*$ para notar esta situación. Cada jugada consiste en una tupla $(a_1, ..., a_k, b_1, ..., b_k)$ con $a_i \in A \sqcup \{*\}$ y $b_i \in B \sqcup \{*\}$ de manera que $a_i = *$ si y sólo si $b_i = *$.

Diremos que $\bar{a} \mapsto \bar{b}$ es un ismorfismo parcial si lo es ignorando los $*$. En cada turno Spoiler elije un par de fichas, digamos el i-ésimo, (que puede estar

libre o haber sido jugado ya) y un elemento de una estructura, supongamos $a_i \in \mathcal{A}$. Luego Duplicator toma la otra ficha del mismo par y un elemento de B de manera que $\bar{a}' \mapsto \bar{b}'$ sea un isomorfismo parcial, donde $\bar{a}'$ ($\bar{b}'$) es las tupla que resulta de cambiar a_i (b_i) por el elemento elegido por Spoiler(Duplicator). Duplicator gana el juego en m pasos si tiene una estrategia que le permite jugar m jugadas seguidas. Y gana el juego *infinito* si puede jugar indefinidamente. Dados $m \in \mathbb{N} \sqcup \{\infty\}$ y $l \leq k$, notamos con $\mathcal{G}_m^k(\mathcal{A}, a_1, ..., a_l, \mathcal{B}, b_1, ..., b_l)$, al juego en m pasos comenzando con $(a_1, ..., a_l *, ..., *, b_1, ..., b_l, *, ..., *)$.

Proposicion 1.4.6 (Teorema 3.3.5 en [9]). *Las siguientes afirmaciones son equivalentes.*

(I) Duplicator *tiene una estrategia para el juego* $\mathcal{G}_\infty^k(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$.

(II) $(\mathcal{A}, \bar{a}) \equiv^{\mathsf{FO}^k} (\mathcal{B}, \bar{b})$.

(III) $(\mathcal{A}, \bar{a}) \equiv^{\mathcal{L}_{\infty\omega}^k} (\mathcal{B}, \bar{b})$.

El resultado anterior no sólo nos da una forma de estudiar el poder expresivo de fragmentos de $\mathcal{L}_{\infty\omega}^\omega$, también muestra que estos juegos caracterizan tanto la relación $\equiv^{\mathcal{L}_{\infty\omega}^k}$ como $\equiv^{\mathsf{FO}^k}$, que notaremos simplemente como $\equiv^k$. Observemos que esto implica que si dos tuplas $\bar{a}$ y $\bar{b}$ cumplen las mismas fórmulas de primer orden con k variables, entonces cumplen las mismas $\mathcal{L}_{\infty\omega}^k$-fórmulas.

Ejemplo 1.4.7 (Paridad con FO(PFP)). *Consideremos $\sigma = \emptyset$ y $\mathcal{A}$ y $\mathcal{B}$ dos σ-estructuras. Como cualquier inyección de A en B es un isomorfismo parcial, si $|A|, |B| \geq s$,* Duplicator *tiene una estrategia para $\mathcal{G}_\infty^s(\mathcal{A}, \mathcal{B})$. Por lo tanto $\mathcal{A} \equiv^s \mathcal{B}$. Luego paridad no es expresable por ninguna $\mathcal{L}_{\infty\omega}^\omega$-fórmula y por lo tanto por ninguna* FO(PFP)*-fórmula.*

1.5. Fragmento existencial de la lógica de segundo orden

Llamamos *lógica de segundo orden* (SO) a la lógica que extiende a FO agregando variables de relación y permitiendo cuantificar sobre éstas. Las fórmulas atómicas son las mismas que en el caso de FO(IFP). Agregamos a las reglas de FO la siguiente. Si X^k es una variable de relación y φ una SO-fórmula entonces,

$$\exists X^k \varphi \tag{1.14}$$

es una SO-fórmula. La interpretamos como

$$(\mathcal{A}, \nu) \models \exists X^k \varphi \quad \text{sii} \quad \text{para alguna relación } R \subset A^k \quad (\mathcal{A}, \nu[\tfrac{X^k}{R}]) \models \varphi.$$

1.5.1. Σ^1_1

Notamos con Σ^1_1 al fragmento de SO que surge de restringirse a las fórmulas del tipo $\exists \bar{X}\varphi$ donde φ no tiene variables de relación cuantificadas.

Ejemplo 1.5.1 (Paseo hamiltoniano)**.** *Un* paseo hamiltoniano *en un grafo $\mathcal{G}$ es un paseo que pasa por cada vértice del grafo una única vez. La Σ^1_1-fórmula*

$$\exists X(\forall x \exists z w(Xzw \wedge Xxw) \wedge \forall xy(Xxy \to Exy) \wedge$$
$$\forall xyz(Xxz \wedge Xyz \to x = y) \wedge \forall xyz(Xzx \wedge Xzy \to x = y)),$$

expresa que $\mathcal{G}$ admite un paseo hamiltoniano.

Proposicion 1.5.2. *Sea σ tal que $\leq \in \sigma$. Sea φ una Σ^1_1-fórmula, existe una $\mathsf{FO}(\mathsf{PFP})$-fórmula ψ equivalente a φ sobre la clase de estructuras ordenadas.*

Demostración. Vamos a ver el caso donde $\varphi = \exists X \varphi'(X)$, con X una variable de aridad uno. La demostración se generaliza a cualquier aridad, modificando el orden $\leq$ por el lexicográfico en las tuplas. Construiremos un operador que, dada una estructura $\mathcal{A}$, recorre, en orden, todos los subconjuntos posibles de A y verifica si satisfacen φ' o no. Definimos un orden entre los predicados a partir de una fórmula ξ, de la siguiente manera. Si P es un conjunto, el siguiente de P, P', es

$$P' := \{a \in A \mid \mathcal{A} \models \xi(P, a)\}.$$

La fórmula en cuestión es

$$\xi(X, x) := \forall y(y < x \to Xy) \wedge \forall y(x \leq y \to \neg Xy).$$

Luego, ψ recorre los conjuntos hasta que alguno satisfaga φ. Si encuentra uno, el proceso se estanca, si no, sigue con el siguiente. Como el sucesor de A es $\emptyset$, el proceso sólo se estanca en un predicado que cumpla φ. Basta entonces chequear, que $\mathcal{F}_\infty$ sea no vacío.

$$\psi := \exists x \mathsf{PFP}\left[Xx \leftarrow \varphi(X) \vee (\neg\varphi(X) \wedge \xi(X, x))\right](x). \tag{1.15}$$

□

Claro que, si el predicado que buscábamos era justamente $\emptyset$, el método no funciona.

Ejemplo 1.5.3 (Paridad en Σ^1_1)**.** *La siguiente Σ^1_1-fórmula expresa paridad sobre estructuras ordenadas.*

$$\varphi_{par} := \exists P((\exists x(min(x) \wedge Px)) \wedge (\forall xy(\varphi_{suc}(xy) \to (Px \leftrightarrow \neg Py))$$
$$\wedge(\exists x(max(x) \wedge \neg Px))))$$

Ejemplo 1.5.4 (Fijar un orden con Σ^1_1)**.** *En el caso de* Σ^1_1*, a diferencia de* $\mathsf{FO}(\mathsf{IFP})$*, restringirse a estructuras ordenadas no le da mayor poder expresivo. La razón de ésto es que puede fijar un orden en la estructura mediante la fórmula*

$$\exists R(\varphi_{ord}[\tfrac{R}{\leq}]). \tag{1.16}$$

La fórmula $\varphi_{ord}[\frac{R}{\leq}]$ *es la que resulta de reemplazar cada aparición de* $\leq$ *en* φ_{ord} *por* R*.*

El ejemplo anterior prueba que, sobre cualquier clase de estructuras, no es cierto que $\mathsf{FO}(\mathsf{PFP})$ sea más expresiva que Σ^1_1.

A continuación definimos las últimas lógicas que trataremos.

1.6. Lógicas que saben contar

Otra de las falencias de la lógica de primer orden, que no se salva con operadores de punto de fijo, como se ve en el Ejemplo 1.2.5, es su incapacidad para referirse al cardinal de estructuras o relaciones. Surge entonces, naturalmente, la necesidad de definir extensiones de ésta que tienen la capacidad de contar. Es decir, queremos poder expresar propiedades del tipo *existen* n *vértices que satisfacen* φ, donde n puede ser variable. Para esto, no alcanza con disponer de los cuantificadores adecuados, si no también de variables numéricas.

1.6.1. Punto fijo con contadores ($\mathsf{FO}(\mathsf{IFP})$+C)

Construiremos $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$ como la siguiente extensión de $\mathsf{FO}(\mathsf{IFP})$.

Sintaxis

Vamos a diferenciar dos tipos de variables individuales,

- las de *vértices* o de tipo v, que notamos con $x, y, z..$ y tomarán valores en el universo de la estructura,
- las *numéricas* de tipo n, que notamos con $i, j, k...$ y tomarán valores en un segmento inicial de $\mathbb{N}$.

Las variables en una tupla $\bar{y}$ pueden tener tipos distintos. El tipo de una tupla, entonces, es $\bar{k} \in \{v, n\}^*$. Las variables de relación $X^{\bar{k}}$ pueden ser de aridad *mixta*.

También agregamos los símbolo distinguidos $\leqslant$ y $\#$ que no figuran en ningún vocabulario σ. Agregamos a las fórmulas de $\mathsf{FO}(\mathsf{IFP})$ las de la forma $i \leqslant j$ donde i, j son variables individuales de tipo n. Extendemos las reglas de formación de $\mathsf{FO}(\mathsf{IFP})$-fórmulas con la siguiente regla: Si φ es una $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$-fórmula y x e i variables de tipo v y n respectivamente,

$$\#x\varphi = i \tag{1.17}$$

es una $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$-fórmula. Sus variables libres son:

$$free(\#x\varphi = i) := (free(\varphi) - x) \cup \{i\}. \tag{1.18}$$

Semántica

Dada una estructura $\mathcal{A}$, definimos $Num(A) := [0, |A|]$. Observemos, para evitar futuras confusiones, que $|Num(A)| = |A| + 1$.

A la hora de interpretar fórmulas, consideramos al universo A como un par $\langle V(A), Num(A)\rangle$. El conjunto $V(A)$ es el de vértices (lo que hasta ahora notabamos como A) y $Num(A)$ es el universo numérico (el que utilizaremos para referirnos a cardinales). Cuando hablamos de tamaño de la estructura, seguimos haciendo referencia a $|V(A)|$, $Num(A)$ es, en algún sentido, "virtual".

Si tenemos $\bar{k} \in \{v, n\}^*$, notamos con $A^{\bar{k}}$ al conjunto

$$U_1 \times U_2 \times ... \times U_s,$$

donde $U_i = V(A)$ si $k_i = v$ y $U_i = Num(A)$ si $k_i = n$. Notaremos también $A^{\bar{y}}$ en lugar de $A^{\bar{k}}$, si $\bar{y}$ es una tupla de tipo $\bar{k}$.

Dada una estructura $\mathcal{A}$, una valuación ν asigna a las variables de tipo v un elemento en $V(A)$ y a las de tipo n elementos en $Num(A)$. Además la valuación asigna a cada variable de relación $X^{\bar{k}}$ una relación $R \subset A^{\bar{k}}$. Para completar la definición de satisfacción, establecemos que

$$(\mathcal{A}, \nu) \models i \leqslant j \quad \text{sii} \quad \nu(i) \leq \nu(j),$$

donde $\leq$ en el lado derecho hace referencia al orden usual de $\mathbb{N}$. Por último $\#x\varphi = i$ debe interpretarse como *hay exáctamente i vértices x que satisfacen* φ. Es decir:

$$(\mathcal{A}, \nu) \models \#x\varphi = i \quad \text{sii} \quad |\{a \in V(A) \mid \mathcal{A} \models \varphi(a)\}| = \nu(i).$$

Utilizaremos la fórmula $\#x\varphi \geqslant i$ para abreviar $\exists j(i \leqslant j \wedge \#x\varphi = j)$, que expresa: *al menos i vértices cumplen* φ.

Observacion 1.6.1. En el Ejemplo 1.3.8 se ve cómo definir aritmética mediante $\mathsf{FO}(\mathsf{IFP})$ sobre segmentos iniciales de $\mathbb{N}$. Por lo tanto en el caso de $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$ para expresar propiedades de elementos de $Num(A)$ disponemos no sólo de $\leqslant$ si no también de $+$ y $\times$ que utilizaremos sin necesidad de volver a definirlos. Asimismo utilizaremos los predicados max y min como: $min(i) = \forall j \neg i \leq j$ y $max(i) = \forall j i \leq j$. De hecho cada uno los elementos de $Num(A)$ es distinguible y podemos hacer referencia a ellos de ser necesario como 1,2,3, etc.

Veamos algunos ejemplos para ilustrar el poder expresivo de $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$.

Ejemplo 1.6.2 (Paridad en $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$)**.** *La siguiente* $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$*-fórmula define la clase de estructuras de universo par:*

$$\varphi_{par} := \exists ij(max(j) \wedge i + i = j).$$

El ejemplo anterior muestra que $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$ es estrictamente más expresiva que $\mathsf{FO}(\mathsf{IFP})$.

Ejemplo 1.6.3 (Circuito euleriano)**.** *Un* circuito euleriano *en un grafo $\mathcal{G}$ es un paseo que empieza y termina en el mismo vértice, tal que por cada arista se pasa una única vez. Basta para esto que el grafo sea conexo y que cada vértice tenga una cantidad par de aristas. Adaptando los Ejemplos 1.6.2 y 1.3.5 probamos que la clase de grafos que admiten un circuito euleriano es definible con* $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$.

Ejemplo 1.6.4 (Contar en base n)**.** *Dada una estructura $\mathcal{A}$, en principio,* $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$ *sólo permite contar en $[0, |A|]$. Podemos extenderlo a $[0, (|A| + 1)^k - 1]$ si consideramos tuplas de variables numéricas. Donde cada una se corresponde con una codificación en base $n + 1$. Dada $(i_1, ..., i_k)$ notamos:*

$$[(m_1, ..., m_k)]_n = \sum_{i=1}^{k} m_i (n+1)^{k-i}. \tag{1.19}$$

Si ordenamos las tuplas con el orden lexicográfico, $<_{lex}$, podemos extener la aritmética definida anteriormente a tuplas. Obtenemos fórmulas $\varphi_+(\bar{i}_1, \bar{i}_2, \bar{i}_3)$ y $\varphi_\times(\bar{i}_1, \bar{i}_2, \bar{i}_3)$ que definen la suma y el producto en base n.

Ejemplo 1.6.5 (Contar clases de equivalencia)**.** *Consideremos una fórmula $\varphi_\equiv(x, y)$ que define una relación de equivalencia $\equiv$ en A. Sea $\psi(x)$ una fórmula tal que*

$$\mathcal{A} \models \forall xy(\varphi_\equiv(x, y) \to (\psi(x) \leftrightarrow \psi(y))),$$

es decir, o bien todos los miembros de una clase satisfacen ψ o ninguno lo hace. Existe una $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$*-fórmula* $\psi^{\equiv}(i)$ *tal que,*

$$\mathcal{A} \models \psi^{\equiv}(m) \quad sii \quad \text{existen } m \text{ clases (módulo } \equiv) \text{ que satisfacen } \psi.$$

Para construir $\psi^{\equiv}$, vamos a contar (en orden) cuantas clases de cada tamaño satisfacen ψ. Sean

$$\alpha(i,j) := \#x(\psi(x) \wedge (\#y\varphi_{\equiv}(x,y) = j)) = i,$$
$$\beta(i,j) := \exists k(\alpha(i,k) \wedge j \times i = k).$$

La fórmula α cuenta cuantos x satisfacen ψ y están en una clase de tamaño i. Utilizamos a β para dividir por i. Por último, debemos sumar todas estas cantidades. Podemos definir esta sumatoria inductivamente.

$$\varphi(S,k,l) := ((k=1) \wedge \beta(k,l)) \vee \exists k'l'l''(suc(k,k') \wedge S(k',l') \wedge \beta(k,l'') \wedge (l' + l'' = l)).$$
$$\psi_{\equiv}(i) := \exists k(max(k) \wedge \mathsf{IFP}\,[Skl \leftarrow \varphi(S,k,l)]\,(ki)).$$

Notaremos con

$$\tfrac{\#}{\varphi} x\psi(x) = i$$

a la fórmula que expresa "hay i clases de equivalencia de la relación definida por φ que satisfacen ψ". Utilizando la aritmética obtenida en el Ejemplo 1.6.4, podemos extender esto a contar clases de equivalencia de una relación definida sobre alguna potencia de A.

1.6.2. FO+C

Claramente, para expresar ciertas propiedades numéricas no hace falta apelar a la recursión, sino, solamente a la aritmética. Definimos $\mathsf{FO} + \mathsf{C}$ como el fragmento de $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$ compuesto por fórmulas que sólo utilizan al operador IFP en subfórmulas de la forma φ_+ y $\varphi_\times$ (ver Ejemplo 1.3.8). Observemos que las operaciones $+$ y $\times$ no son definibles a partir de $\leq$ por ninguna fórmula de primer orden, ya que a partir de $+$ podemos definir paridad, contradiciendo al Ejemplo 1.2.6. En resumen, permitimos las fórmulas

$$i + j = k,$$
$$i \times j = k,$$

donde i, j, k son variables numéricas y se interpretan de manera natural, entendiendo a $+$ y $\times$ como la suma y el producto que determina el orden usual de $\mathbb{N}$. Con esta definición de $\mathsf{FO} + \mathsf{C}$ la fórmula φ_{par}, definida en el Ejemplo 1.6.2, resulta una $\mathsf{FO} + \mathsf{C}$-fórmula.

Capítulo 2

Complejidad Descriptiva

En este capítulo presentamos las principales nociones de la teoría de la complejidad de las que haremos uso.

Comenzamos fijando un modelo de cómputo e introduciendo las definiciones y resultados básicos de la teoría de la complejidad (un desarrollo más profundo se puede encontrar en [25, 21, 2]). Definimos las distintas clases de complejidad y estudiamos, brevemente, cómo se relacionan.

En la Sección 2.2 establecemos las relaciones entre las clases de complejidad definidas, y las lógicas presentadas en el capítulo anterior, dando la noción de lógica que captura una clase de complejidad. Presentamos los principales resultados que dieron comienzo a la complejidad descriptiva, como por ejemplo el Teorema de Fagin, y estudiamos algunas de sus consecuencias.

Por último, nos concentramos en el problema de encontrar una lógica que capture PTIME sobre todas las estructuras, uno de los principales problemas abiertos del área, pues permitiría traducir el problema de separar las clases PTIME y NPTIME a diferenciar el poder expresivo de dos lógicas.

2.1. Modelo de cómputo

Antes de poder introducir los principales conceptos y resultados de este capítulo necesitamos fijar un modelo de cómputo, para poder referirnos adecuadamente a las distintas clases de complejidad que tendrán nuestro interés. Elegimos para este fin a las *máquinas de Turing*, que permiten una simple descripción de dichas clases. Una máquina de Turing, $\mathfrak{M}$, consiste en una o varias cintas, acotadas a la izquierda e infinitas a la derecha. La cinta está divida en celdas y cada una tiene un símbolo de un alfabeto Σ. Habitualmente consideraremos $\Sigma = \{0, 1\}$ (pueden utilizarse algunos símbolos distinguidos para marcar inicio y final de las cintas). Cada celda tiene un cabezal de

lectura/escritura. Las primeras celdas no pueden escribirse y representan la *entrada* de la máquina. Las celdas restantes son de lectura/escritura y se denominan de *trabajo*. A su vez, $\mathfrak{M}$ tiene conjunto finito de estados, Q, que contiene los símbolos destacados: q_0, q_+ y q_-, estados inicial, de aceptación y de rechazo respectivamente. Llamamos *configuración* de $\mathfrak{M}$ a la siguiente información:

- el estado actual,
- el contenido de cada una de las cintas de trabajo,
- la posición de los cabezales.

La máquina también dispone de una tabla finita de instrucciones T, que determina si dadas dos configuraciones C y C', puede pasarse de una a la otra. En ese caso, C' se dice *sucesora* de C. Si cada configuración posible tiene a lo sumo una sucesora, $\mathfrak{M}$ se dice *determinística*. Caso contrario se dice *no determinística*. La máquina empieza en una configuración con un estado inicial y opera paso a paso, pasando siempre de una configuración a otra sucesora. Si llega a una configuración con un estado de aceptación o rechazo, se detiene.

Formalemente, entonces, una máquina de Turing $\mathfrak{M}$ consiste en una tupla:

$$\mathfrak{M} = \langle \Sigma, T, Q \rangle.$$

Las máquinas que consideraremos son de *decisión*. Dada una palabra $w \in \Sigma^*$ decimos que $\mathfrak{M}$ *acepta* a w si, en alguna corrida con w como entrada, $\mathfrak{M}$ termina en un estado de aceptación. Decimos que $\mathfrak{M}$ *rechaza* a w si todas las corridas con w como entrada terminan en un estado de rechazo. Dado $W \subset \Sigma^*$, decimos que $\mathfrak{M}$ *decide* W si para toda paralabra $w \in \Sigma^*$ vale que:

(I) si $w \in W$ $\mathfrak{M}$ acepta a w,

(II) si $w \notin W$ $\mathfrak{M}$ rechaza a w.

2.1.1. Clases de complejidad

Con *clase de complejidad*, nos referimos a una familia de problemas que son decidibles por una máquina de Turing con determinadas propiedades. Las variables a tener en cuenta, que dan forma a las clases más habituales, son el *espacio* (cantidad de celdas utilizadas como máximo en cada una de las cintas de trabajo) y el *tiempo* (cantidad máxima de pasos necesaria para realizar el cómputo). Estas cotas vienen determinadas por funciones del tamaño de la entrada, que denotamos con n.

Definicion 2.1.1. Dada una familia de funciones $\mathcal{F} \subset \mathbb{N}^{\mathbb{N}}$ y una máquina Turing $\mathfrak{M}$, decimos que $\mathfrak{M}$ es $\mathcal{F}$-*Space*, si existe una función $f \in \mathcal{F}$ tal que cada cinta de trabajo utiliza a lo sumo $f(n)$ celdas, donde n es la cantidad de celdas utilizadas en las cintas de entrada. Decimos que es $\mathcal{F}$-*Time* si el cómputo se realiza en a lo sumo $f(n)$ pasos.

Las familias de funciones[1]

$$\mathfrak{Log} := \{f \in \mathbb{N}^{\mathbb{N}} | f = d\log(n) \quad \text{con} \quad d \in \mathbb{N}\}, \tag{2.1}$$

$$\mathfrak{P} := \{f \in \mathbb{N}^{\mathbb{N}} | f = kn^d \quad \text{con} \quad d, k \in \mathbb{N}\}. \tag{2.2}$$

Dan pie a las clases de complejidad que consideraremos en este trabajo, a saber:

LOGSPACE clase de problemas decidibles por una máquina de Turing determinística $\mathfrak{Log}$-Space,

NLOGSPACE clase de problemas decidibles por una máquina de Turing no determinística $\mathfrak{Log}$ - Space,

PTIME clase de problemas decidibles por una máquina de Turing determinística $\mathfrak{P}$-Time,

NPTIME clase de problemas decidibles por una máquina de Turing no determinística $\mathfrak{P}$-Time,

PSPACE clase de problemas decidibles por una máquina de Turing determinística $\mathfrak{P}$-Space.

En general usaremos la misma notación para máquinas con una determinada cota que para la clase de complejidad que determinan. Por ejemplo, diremos $\mathfrak{M}$ es PSPACE en lugar de $\mathfrak{M}$ es $\mathfrak{P}$-Space. Pueden establecerse las siguientes relaciones entre las clases mencionadas,

$$\text{LOGSPACE} \subseteq \text{NLOGSPACE} \subseteq \text{PTIME} \subseteq \text{NPTIME} \subseteq \text{PSPACE}. \tag{2.3}$$

Las únicas que no son consecuencia directa de la definición son NPTIME $\subseteq$ PSPACE y NLOGSPACE $\subseteq$ PTIME.

[1]Con log nos referimos al logaritmo en base dos como función a valores naturales, asignando el primer natural mayor.

Nota. Para entender estas inclusiones pensemos a una máquina $\mathfrak{M}$, dada una entrada x, como un grafo dirigido. Cada vértice del grafo representa una configuración y dos son adyacentes si una es sucesora de la otra. El hecho de que la máquina es no determinística, se traduce en que el grafo tiene bifurcaciones. La cantidad de configuraciones posibles es exponencial en la cantidad de celdas (de cada cinta) utilizadas. En el primer caso, como en tiempo polinomial no se pueden leer más que una cantidad polinomial de celdas, podemos suponer esta cota también para el espacio. La cota temporal se traduce a la profundidad del grafo, es decir, la longitud máxima de los caminos. Cada sucesión de decisiones, puede codificarse entonces en una cinta utilizando una cantidad polinomial de celdas. Para simular la máquina no determinística mediante una determinística basta recorrerlas todas en orden, reutilizando una cinta. Aplicando los resultados que veremos más adelante, se puede traducir esta inclusión a una desigualdad de lógicas, como muestra el Corolario 2.2.11.

En cuanto a $\mathsf{NLOGSPACE} \subseteq \mathsf{PTIME}$, si la máquina sólo utiliza $\log(n)$ celdas, el grafo que describe su comportamiento, dada una determinada entrada, tiene una cantidad polinomial de vértices. El problema se traduce entonces a decidir, si es posible llegar desde una configuración inicial a una de aceptación, es decir, si existe un camino entre estos dos vértices del grafo. Como esto puede hacerse mediante un algoritmo en tiempo polinomial (ver [25] Sección 1), puede construirse una máquina con la cota deseada, que dada una entrada y a partir de la máquina original, decide si existe el camino buscado.

Vale también aclarar, que de las inclusiones mencionadas, la única que se sabe estricta es $\mathsf{NLOGSPACE} \subsetneq \mathsf{PSPACE}$ (ver [25]). Siendo todas las demás importantes problemas abiertos.

2.1.2. Codificación de estructuras

Para trabajar con estructuras como entradas de un máquina de Turing, necesitamos primero traducir éstas a expresiones en un alfabeto. Elegimos para esto a $\{0, 1\}$. Si fijamos un orden en la estructura (numeramos al universo), es fácil expresar una estructura como una tira de ceros y unos, construyendo para cada relación, algo similar a la matriz de adyacencia de un grafo. En el caso de una relación de aridad k, y estructura de tamaño n, podemos codificar la relación como una tira de longitud n^k, con un uno en la i-ésima celda si la tupla correspondiente está en la relación y un cero caso contrario. Concatenando las tiras estamos codificando cada σ-estructura como una sola palabra. Dado un vocabulario σ, una máquina para σ-estructuras cuenta

con una cinta de lectura para el universo de la estructura (un uno por cada elemento) y una adicional para cada relación. Consideraremos, en este caso, como tamaño de la entrada al cardinal del universo (no teniendo en cuenta el lugar ocupado en las cintas correspondientes a las relaciones). Hacemos esto por simplicidad, ya que las clases de complejidad definidas anteriormente se mantienen invariantes por cambios de este tipo (ver [9]).

Nótese que con este método, necesariamente al codificar las relaciones le estamos dando un orden (arbitrario) a la estructura, que en principio no viene provisto de uno. La lógica tiene la capacidad de referirse a las estructuras sin necesidad de una representación de éstas. Es por esto que el problema de elegir un orden de forma canónica, ocupa un lugar central en la teoría de la complejidad descriptiva.

Observacion 2.1.2. En el caso de estructuras ordenadas, esta codificación resulta canónica. Es decir, dos estructuras ordenadas isomorfas (como estructuras ordenadas) tienen la misma codificación. Lo que nos permite identificar una estructura ordenada con su codificación y pensar que un algoritmo trabaja directamente sobre la estructura.

A partir de este momento vamos a considerar $\leq$ como un símbolo distinguido, que no forma parte de ningún vocabulario σ.

Definicion 2.1.3. Dada una clase $\mathcal{K}$ de σ-estructuras, definimos:

$$\mathcal{K}_{\leq} := \{\langle \mathcal{A}, \leq^{\mathcal{A}} \rangle | \mathcal{A} \in \mathcal{K} \quad \text{y} \quad \mathcal{A} \models \varphi_{ord}\}.$$

$\mathcal{K}_{\leq}$ consiste en todas las estructuras de $\mathcal{K}$ con todos los ordenes posbiles.

Dadas dos estructuras $\mathcal{A}$ y $\mathcal{B}$ isomorfas, los conjuntos $\mathcal{A}_{\leq}$ y $\mathcal{B}_{\leq}$ resultan iguales (como conjuntos de codificaciones). Un algoritmo que decida una propiedad $\mathcal{P}$ sobre una estructura $\mathcal{A}$, debe hacerlo para cualquier codificación (orden) de ésta. Por lo que decimos que una máquina de Turing decide una propiedad $\mathcal{P}$ si decide $\mathcal{P}_{\leq}$. Decimos que una máquina decide una fórmula φ si decide la propiedad definida por φ. En el caso de fórmulas con variables libres, consideramos como entrada a pares $(\mathcal{A}, \bar{a})$, donde $\bar{a}$ indica la valuación en las variables libres de φ. En ese caso la máquina tiene una cinta adicional que indica los elementos en $\bar{a}$.

2.2. Lógicas que capturan clases de complejidad

La primera noción de *capturar* una clase de complejidad que tendremos en cuenta, sirve a fines de introducir el problema y entender la relación entre

las lógicas presentadas en el Capítulo 1 y las clases de complejidad definidas en la Sección 2.1.1. Ésta resulta poco precisa como definición más general. Retomaremos esto más adelante para estudiar sus limitaciones y dar con una definición que se ajuste mejor a nuestras necesidades.

La relación entre las clases de complejidad y las lógicas tiene dos aspectos. Por un lado tenemos la complejidad de la relación de satisfacción, es decir, dada una fórmula φ, qué poder de cómputo se necesita para decidir si una estructura la satisface o no. En este caso la fórmula está fija y consideramos como entrada del algoritmo a la estructura. Esto en general es conocido como *data complexity*[2]. Por otro lado, dada una propiedad decidible por una máquina con una determinada cota en su complejidad, qué lógica se necesita para definirla. Cuando decimos que una propiedad $\mathcal{P}$ pertenece o está en una determinada clase de complejidad, queremos decir que el problema de decidir $\mathcal{P}_{\leq}$ lo está. Tenemos entonces la siguiente definición:

Definicion 2.2.1. Sea $\mathcal{L}$ una lógica y $\mathfrak{C}$ una clase de complejidad. Decimos que $\mathcal{L}$ captura $\mathfrak{C}$, si para toda propiedad $\mathcal{P}$ vale:

$$\mathcal{P} \quad \text{es } \mathcal{L}\text{-definible sii} \quad \mathcal{P}_{\leq} \in \mathfrak{C}.$$

En muchos casos no es posible encontrar una lógica que capture una clase de complejidad. Por lo que nos restringimos a algunas clases de estructuras. Vamos a enfocar primero nuestra atención al caso de lenguajes $\sigma \sqcup \{\leq\}$ y la clase de estructuras ordenadas que notamos $\mathcal{O}(\sigma)$. En el Capítulo 3 veremos qué sucede con otras clases.

Definicion 2.2.2. Sea $\mathcal{K}$ una clase de σ-estructuras, $\mathcal{L}$ una lógica y $\mathfrak{C}$ una clase de complejidad. Decimos que $\mathcal{L}$ captura $\mathfrak{C}$ sobre $\mathcal{K}$, si para toda propiedad $\mathcal{P}$ vale:

$$\mathcal{P} \cap \mathcal{K} \quad \text{es } \mathcal{L}\text{-definible sii} \quad \mathcal{P}_{\leq} \in \mathfrak{C}.$$

2.2.1. La complejidad de la relación de satisfacción para FO

Como primer ejemplo podemos estudiar el caso de la lógica de primer orden.

Proposicion 2.2.3. *Toda propiedad sobre estructuras finitas definible en* FO *es decidible por una máquina de Turing en* LOGSPACE.

[2]También podemos fijar la estructura y considerar a la fórmula como entrada (*expresion complexity*) o a ambas (*combined complexity*).

Demostración. Veámoslo por inducción en las fórmulas. En el caso de fórmulas atómicas sólo hace falta comparar las celdas correspondientes de las cintas que codifican la relación en cuestión (para el caso $R\bar{x}$) o verificar los lugares correspondientes a las valuaciones (en el caso $x = y$). Si tenemos una máquina determinística $\mathfrak{M}$ para φ y una $\mathfrak{M}'$ para ψ con las cotas adecuadas, podemos construir una para $\varphi \vee \psi$ corriendo una y luego la otra y finalizando en un estado de aceptación si alguna lo hace. Para $\neg\varphi$, basta intercambiar los estados de aceptación y rechazo. En el caso de $\exists x\varphi(x)$, si $\mathfrak{M}$ es la máquina para $\varphi(x)$, la iteramos utilizando una cinta para guardar el valor de x en cada iteración, hasta llegar a un estado de aceptación (o no). Para esto solo usamos $\log(n)$ celdas. □

Si bien toda propiedad FO-definible es decidible en LOGSPACE, FO carece del poder para expresar propiedades computables con muy pocos recursos. Podemos construir una máquina que decide si una estructura tiene universo par, utilizando únicamente una celda de trabajo. Consiste en una que recorre el universo alternando el contenido de dicha celda entre 1 y 0. Ya vimos que la clase de estructuras de universo par no es definible en primer orden, más aún, la clase de estructuras ordenadas de universo par tampoco. Para poder capturar LOGSPACE y NLOGSPACE, al menos sobre estructuras ordenadas, es necesario extender a FO con operadores de clausura transitiva. Construyendo así las lógicas FO(TC) y FO(DTC) (ver [9] para una definición).

2.2.2. El caso de FO(IFP) y FO(PFP)

El poder de cómputo necesario para decidir propiedades definidas por fórmulas de lógicas que extienden a FO con operadores de punto fijo es un poco mayor.

Proposicion 2.2.4. *Valen las siguientes afirmaciones:*

(a) Toda propiedad definible en FO(IFP) *es decidible en* PTIME.

(b) Toda propiedad definible en FO(PFP) *es decidible en* PSPACE.

Demostración. Ya vimos que para fórmulas de primer orden alcanza con LOGSPACE. Falta ver qué pasa con fórmulas construidas a partir de los operadores IFP y PFP. Sea $\varphi = \mathsf{IFP}\,[X\bar{x} \leftarrow \psi(X, \bar{x})]\,(\bar{t})$ y $\mathfrak{M}$ una máquina determinística para ψ. Construimos una máquina determinística $\mathfrak{M}'$ que opera de la siguiente manera: Recibe como entrada un par $(\mathcal{A}, \bar{a})$. Utilizamos dos cintas de trabajo W_1 y W_2, W_1 contiene una relación R de aridad k (la de X) y mediante $\mathfrak{M}$ escribe en W_2 la relación $R' = \{\bar{b} \in A^k | (\mathcal{A}, \bar{b}) \models R\bar{x} \vee \psi(R, \bar{x})\}$. Comienza con R como la relación vacía. En cada paso chequea si $R = R'$, si es

así basta ver si $R\bar{a}$. Si $R \neq R'$ sigue. Por la Proposición 1.3.2 el procedimiento se itera como mucho n^k veces.

Lo anterior es fácilmente adaptable al caso FO(PFP). Como sabemos que, de existir, el punto fijo se alcanza en 2^n-1 pasos, basta iterar el procedimiento esa cantidad de veces, utilizando n lugares de una cinta extra como contador. Claro que en cada paso, la relación a construir es $R' = \{\bar{b} \in A^k | (\mathcal{A}, \bar{b}) \models \psi(R, \bar{x})\}$. Si cuando finaliza vale $R = R'$ y $R\bar{a}$, acepta. □

Recordemos, del Ejemplo 1.4.3, que $\mathcal{L}^{\omega}_{\infty\omega}$ puede definir propiedades que no son computables. Como cualquier propiedad definible en FO(PFP) está en PSPACE, concluimos (aplicando la Proposición 1.4.4) el siguiente corolario.

Corolario 2.2.5. $\mathsf{FO(PFP)} \leqq \mathcal{L}^{\omega}_{\infty\omega}$.

Nuevamente, estas extensiones de FO fallan a la hora de expresar propiedades sobre cardinalidad del universo. No capturan por lo tanto PTIME ni PSPACE, al menos no sobre todas las estructuras. Sin embargo, al restringirnos a estructuras ordenadas su poder expresivo crece notoriamente.

2.2.3. El teorema de Immerman-Vardi

El Teorema de I-V, junto con el de Fagin, es uno de los principales en el área. Originalmetne fue enunciado para la lógica FO(LFP) en lugar de FO(IFP), pero como son equivalentes y FO(IFP) es la que usamos, lo enunciaremos para ésta última.

Teorema 2.2.6 (I-V)**.** FO(IFP) *captura* PTIME *sobre estructuras ordenadas.*

Una demostración rigurosa de este teorema puede encontrarse en [9]. A continuación presentaremos los principales argumentos que se utilizan en la demostración.

La principal idea de esta demostración (y de muchas de los resultados de este estilo) es la de encontrar los recursos lógicos adecuados para describir la configuración de una máquina de Turing. Fijemos un vocabulario $\sigma \sqcup \{\leq\}$. El hecho de que toda propiedad definible mediante FO(IFP) es decidible en PTIME está demostrado en la Proposición 2.2.4. Fijemos entonces una propiedad $\mathcal{P}$ de σ-estructuras ordenadas que es decidible por una máquina $\mathfrak{M}$ en PTIME. Recordemos que una configuración C contiene toda la información relevente de la máquina. Podemos suponer $\mathfrak{M}$ polinomialmente acotada en tiempo y en espacio (ver Sección 2.1.1). Fijemos $d \in \mathbb{N}$ tal que n^d acota el contenido de cada cinta de trabajo y a la vez es mayor que la aridad de cualquier relación en σ. Dada una estructura de universo n, una variable puede representar n elementos, o sea, con una variable alcanza para codificar lo que

en una cinta necesita $\log(n)$ celdas. Con d variables entonces codificamos el equivalente a $d\log(n)$ celdas, y con una relación de ardidad d alcanza para codificar $2^{d\log(n)} = n^d$ celdas, que es el contendio relevante de cada cinta. Para el caso de la posición de los cabezales, mediante un razonamiento similar podemos codificar cada una en una relación de la misma aridad. El estado no representa un problema, pues hay una cantidad finita de éstos. Si resumimos estas tres en una sola, tenemos para cada configuración posible, una relación.

No es difícil, pero si requiere cierto trabajo, construir las fórmulas que describen estas relaciones. Notemos con C_i a la configuración en el i-ésimo paso. Como $\mathfrak{M}$ está acotada en tiempo, podemos suponer $i \leq n^d$, por lo que, aumentando la aridad de cada relación (para indicar el tiempo), podemos, no sólo codificar en C_i la configuración actual, si no también todas las anteriores. De esa manera $C_i \subseteq C_{i+1}$, siendo la configuración final el punto fijo de un proceso inflacionario. La FO(IFP)-fórmula que buscamos es la que define este punto fijo. Basta ver si en éste hay un estado de aceptación.

Proposicion 2.2.7. FO(PFP) *captura* PSPACE *sobre la clase de estructuras ordenadas.*

A diferencia del caso anterior, la cantidad de configuraciones podría ser exponencial. No podemos expresarlo como un proceso inflacionario, por lo cual necesitamos FO(PFP) en lugar de FO(IFP). Como suponemos que una configuración de aceptación es sucesora de sí misma, de llegar a una, el proceso se estanca, caso contrario, oscila.

Teorema 2.2.8 (Fagin)**.** Σ^1_1 *captura* NPTIME.

Veámoslo primero para el caso de estructuras ordenadas. Razonando igual que en el Teorema 2.2.6, consideremos la secuencia C_i. Al tratarse de una máquina no determinística, no existe una única sucesión de configuraciones posible. Se trata entonces de expresar "existe una sucesión de configuraciones que termina en un estado de aceptación", lo que resulta naturalmente expresable en Σ^1_1.

Generalicemos el resultado a todas las estructuras. La clave para esto es la capacidad de Σ^1_1 de fijar un orden sobre el universo (Ejemplo 1.5.4). Sea ψ es la Σ^1_1-fórmula que define una propiedad NPTIME sobre estructuras ordenadas y φ_{ord} la fórmula definida en el Ejemplo 1.1.3. Definimos $\psi[\frac{R}{\leq}]$ y $\varphi_{ord}[\frac{R}{\leq}]$ las fórmulas que resultan reemplazar cada aparición de $x \leq y$ por Rxy. Luego,

$$\exists R(\varphi_{ord}[\tfrac{R}{\leq}] \wedge \psi[\tfrac{R}{\leq}]) \tag{2.4}$$

define la propiedad sobre todas las estructuras.

2.2.4. Algunas consecuencias

Los resultados vistos en la sección anterior permiten traducir las desigualdades mostradas en la Sección 2.1.1 a desigualdades en el poder expresivo de las lógicas correspondientes. Asimismo los problemas de separar las clases de complejidad se traducen en problemas de separar las lógicas. Esto se expresa en los siguientes corolarios.

Corolario 2.2.9. *Son equivalentes:*

(I) $\mathsf{FO(IFP)} \equiv \Sigma^1_1$ *sobre estructuras ordenadas.*

(II) $\mathsf{PTIME} = \mathsf{NPTIME}$.

Corolario 2.2.10. *Son equivalentes:*

(I) $\mathsf{FO(IFP)} \lneqq \mathsf{FO(PFP)}$ *sobre estructuras ordenadas.*

(II) $\mathsf{PTIME} \neq \mathsf{PSPACE}$.

El siguiente corolario es la Proposición 1.5.2 en términos de clases de complejidad.

Corolario 2.2.11. $\mathsf{NPTIME} \subseteq \mathsf{PSPACE}$.

2.2.5. $\mathsf{FO(IFP)} + \mathsf{C}$ sobre estructuras ordenadas

$\mathsf{FO(IFP)} + \mathsf{C}$ extiende a $\mathsf{FO(IFP)}$ con contadores. En cuanto a la complejidad necesaria para decidir una $\mathsf{FO(IFP)} + \mathsf{C}$-fórmula, ésta no difiere de la de $\mathsf{FO(IFP)}$. Basta ver qué recursos se necesitan para decidir una fórmula del tipo $\#x\varphi(x) = i$, donde φ es una $\mathsf{FO(IFP)}$-fórmula. Si $\mathfrak{M}$ es la máquina que decide $\varphi(x)$ en tiempo polinomial, la iteramos haciendo que x recorra el universo. Guardando en un contador la cantidad de aceptaciones. Esto ocupa sólo $\log(n)$ celdas de una cinta de trabajo. Por lo tanto una $\mathsf{FO(IFP)} + \mathsf{C}$-fórmula también puede decidirse en PTIME. Lo que prueba, mediante el Teorema de I-V, que $\mathsf{FO(IFP)} + \mathsf{C}$ y $\mathsf{FO(IFP)}$ son equivalentes sobre estructuras ordenadas. Deducimos entonces la siguiente proposición.

Proposicion 2.2.12. $\mathsf{FO(IFP)} + \mathsf{C}$ *captura* PTIME *sobre estructuras ordenadas.*

2.3. Una lógica para PTIME

El problema de encontrar una lógica que capture PTIME (sobre todas las estructuras, no sólo ordenadas) es uno de los pincipales problemas abiertos de la teoría de la complejidad descriptiva. Una respuesta positiva, permitiría traducir el problema de comparar PTIME con NPTIME a comparar dos lógicas. La pregunta tiene sus origenes en el trabajo de Chandra y Harel [5] y la formulación que aquí presentamos se debe a Gurevich [16] quien conjetura una respuesta negativa. Como el Teorema de Fagin (2.2.8) prueba que sí existe una lógica que captura NPTIME, si esta conjetura es cierta, implica que PTIME $\neq$ NPTIME.

Para poder enunciar el problema adecuadamente, es necesario primero dar una definición más precisa de *lógica*. Basandonos en los trabajos de Gurevich [16] y Grohe [15] elejimos la que presentamos a continuación. Ebbinghaus [8] formula una serie de generalizaciones y definiciones que permiten refinar esta definición.

Definicion 2.3.1. Una *lógica abstracta* es un par $\mathcal{L} = (\mathcal{SEN}, \models_{\mathcal{L}})$, donde $\mathcal{SEN}$ asigna a cada vocabulario finito σ, un conjunto decidible de expresiones, $\mathcal{SEN}[\sigma]$, cuyos elementos se llaman $\mathcal{L}[\sigma]$-sentencias. La función $\models_{\mathcal{L}}$, asigna a cada $\mathcal{L}[\sigma]$-sentencia, φ, una propiedad de σ-estructuras $\mathcal{P}_\varphi$.

Decimos que φ define $\mathcal{P}_\varphi$ y que $\mathcal{P}_\varphi$ es $\mathcal{L}$-definible. Dada una σ-estructura $\mathcal{A}$ notamos $\mathcal{A} \models_{\mathcal{L}} \varphi$ si $\mathcal{A} \in \mathcal{P}_\varphi$.

Es claro que las lógicas definidas en el Capítulo 1 son lógicas abstractas si nos restringimos a las sentencias.

Definicion 2.3.2. Sea $\mathcal{L}$ una lógica abstracta. Decimos que $\mathcal{L}$ *captura* PTIME si cumple las siguientes condiciones:

P.1 Para cada vocabulario σ, existe una máquina de Turing, $\mathfrak{M}$, que para cada sentencia $\varphi \in \mathcal{SEN}[\sigma]$, devuelve una máquina $\mathfrak{M}_\varphi$ PTIME que decide φ.

P.2 Para cada vocabulario σ y para cada máquina $\mathfrak{M}$ en PTIME que decide una propiedad de σ-estructuras $\mathcal{P}_{\mathfrak{M}}$, existe $\varphi \in \mathcal{SEN}[\sigma]$ que define $\mathcal{P}_{\mathfrak{M}}$.

Observacion 2.3.3. La máquina a la que hacemos referencia en (P.1), difiere de las definidas en la Sección 2.1. Pues las nuevas computan una función, mientras que las anteriores sólo decidían una propiedad. Para adaptar la definición a este contexto, basta agregar una cinta de *salida* que, al finalizar el algoritmo, contiene una palabra que codifica a una máquina. La codificación de una máquina como palabra en $\{0,1\}^*$ es similar a la de una estructura.

Notemos que la condición (P.1) es más fuerte que la de la Definición 2.2.2 que sólo pedía una máquina para cada fórmula. La razón para esto es evitar lógicas como la del siguiente ejemplo.

Ejemplo 2.3.4 (Lógica trivial)**.** *Para cada vocabulario σ existen sólo numerables máquinas $\mathfrak{M}$ en* PTIME*, por lo tanto sólo existen numerables propiedades de σ-estructuras en* PTIME*. Sean estas $\mathcal{P}_1^\sigma$, $\mathcal{P}_2^\sigma$, $\mathcal{P}_3^\sigma$,... Definamos la lógica $\mathcal{L}$ donde $\mathcal{SEN}[\sigma] := \mathbb{N}$ y $\mathcal{A} \models_{\mathcal{L}} n$ sii $\mathcal{A} \in \mathcal{P}_n$. Es claro que cumple (P.2) y que toda sentencia tiene un algoritmo que la decide (que no podemos computar). Claramente ésta no es una lógica que queremos aceptar en nuestra definición.*

En cuanto a FO, FO(IFP), FO(IFP) + C todas cumplen (P.1). Como la definición de $\mathcal{L}$-fórmula para estas lógicas es recursiva, la máquina en cuestión debe construir el algoritmo de decisión recursivamente a partir de los algorítmos construidos en la Sección 2.2. Sin embargo FO y FO(IFP) no cumplen (P.2), pues no expresan propiedades como la paridad. El caso de FO(IFP) + C es más difícil. De hecho se conjeturó que esta lógica capturaba PTIME. Cai, Fürer e Immerman [4] demostraron que existe una propiedad de grafos decidible en tiempo polinomial que no es definible en FO(IFP) + C, descartando así esta conjetura. Sin embargo otros resultados positivos fueron demostrados, extendiendo las clases sobre las que FO(IFP) + C captura PTIME, algunos de estos serán presentados en el próximo capítulo.

Capítulo 3

Órdenes definibles y canonizaciones

En este capítulo vemos como extender el teorema de Immerman Vardi, encontrando algunas clases de estructuras no ordenadas donde las lógicas $\mathsf{FO(IFP)}$ y $\mathsf{FO(IFP)} + \mathsf{C}$ capturan PTIME.

Comenzamos tratando el caso de estructuras que no vienen provistas de un orden, pero puede definirse uno tomando algunos parámetros. Presentamos ejemplos de clases de estructuras donde esto es posible y clases donde no.

En la Sección 3.2 nos concentramos en el caso de estructuras rígidas, es decir, que no admiten automorfismos. Estudiamos la relación entre la rigidez de una estructura y la capacidad de una lógica de distinguir sus elementos y de definir un orden sobre ésta.

Luego, definimos las transducciones, una herramienta lógica que nos será de gran utilidad para obtener, de manera simple, diversos resultados. La razón de esto es que permite relacionar diversas clases de estructuras, incluso en vocabularios distintos.

Finalmente, presentamos una generalización de la noción de orden, la de canonización. Se trata de, dada una estructura, en lugar de darle un orden, asociarle una estructura ordenada isomorfa de forma canónica. Ilustramos este método con diversos ejemplos y presentamos algunas técnias que permiten extender las clases de estructuras canonizables.

3.1. Órdenes definibles

Definicion 3.1.1. Una $\mathsf{FO(IFP)} + \mathsf{C}$-fórmula $\varphi(\bar{t}, x, y)$ define un orden con k parámetros, sobre una clase de estructuras $\mathcal{K}$, si para cada σ-estructura

$\mathcal{A} \in \mathcal{K}$, existe $\bar{p} \in A^k$, tal que la relación defeinida por $\varphi(\bar{p}, x, y)$ resulta un orden total.

Proposicion 3.1.2. *Si existe una* $\mathsf{FO(IFP)} + \mathsf{C}$*-fórmula* φ *que define un orden sobre una clase* $\mathcal{K}$ *de estructuras, entonces* $\mathsf{FO(IFP)} + \mathsf{C}$ *captura* PTIME *sobre* $\mathcal{K}$*.*

Demostración. Sea $\varphi(\bar{t}, x, y)$ la $\mathsf{FO(IFP)} + \mathsf{C}$-fórmula que define un orden sobre $\mathcal{K}$ y $\mathcal{P}$ una propiedad de σ-estructuras en PTIME. Consideremos $\mathcal{P}_{\leq}$ la clase de $\sigma \sqcup \{\leq\}$ estructuras ordenadas $\mathcal{A}$ tales que $\mathcal{A}|_{\sigma} \in \mathcal{P}$. Por el teorema de I-V existe una σ-fórmula ψ que define a $\mathcal{P}_{\leq}$. Tomemos $\tilde{\psi}$ la fórmula que resulta de reemplazar todas las apariciones de $x \leq y$ en ψ por $\varphi(\bar{t}, x, y)$. Luego,

$$\exists \bar{t}(\varphi(\bar{t}, x, y) \text{ “define un orden total”}) \wedge \tilde{\psi}) \tag{3.1}$$

define $\mathcal{P}$. □

Ejemplo 3.1.3 (Líneas)**.** *Consideremos la clase de líneas (ver Figura 3.1). La siguiente* $\mathsf{FO(IFP)}$*-fórmula:*

$$\varphi(p, x, y) = \mathsf{IFP}\left[Xxy \leftarrow (x = p) \vee (\exists z(Xzx \wedge z \neq y \wedge Exy))\right](xy),$$

“dirige” la relación de adyacencia E *desde* p*. Si tomamos* p *como uno de los dos extremos, la clausura transitiva de la relación definida por* φ *es un orden lineal sobre la estructura.*

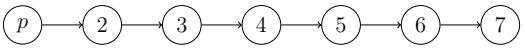


Figura 3.1: Línea ordenada

Ejemplo 3.1.4 (Ciclos)**.** *Si consideramos, en cambio, la clase de los ciclos un parámetro no basta para determinar un orden. Pero alcanza con fijar dos. El primer y segundo elemento. La relación de sucesor que genera la relación de orden buscada, viene dada por (ver Figura 3.2):*

$$\varphi(p_1, p_2, x, y) := \mathsf{IFP}\left[Xxy \leftarrow \psi(X, x, y)\right](xy),$$

con

$$\psi(X, x, y) := ((x = p_1) \wedge (y = p_2)) \vee \\ ((y \neq p_1) \wedge (\exists x'(Xx'x \wedge x' \neq y \wedge Exy))).$$

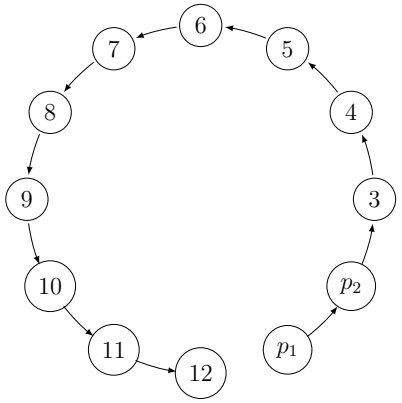


Figura 3.2: Ciclo ordenado

Es importante recalcar que los casos interesantes son clases con infinitas estructuras. Como permitimos parámetros a la hora de definir órdenes, podemos dar, trivialmente, con una fórmula que defina un orden sobre una estructura o sobre una clase finita (módulo isomorfismos). Como muestran las siguientes proposiciones.

Proposicion 3.1.5. *Sea $\mathcal{K}$ una clase de estructuras de cardinal acotado, es decir, que existe un $k \in \mathbb{N}$ tal que $|A| \leq k$ para toda $\mathcal{A} \in \mathcal{K}$. Entonces $\mathcal{K}$ admite un orden* FO*-definible.*

Demostración. Definimos la siguiente fórmula con k parámetros.

$$\varphi(t_1, ..., t_k, x, y) := \bigvee_{i \leq j} ((x = t_i) \wedge (y = t_j)). \tag{3.2}$$

□

Corolario 3.1.6. *Sea $\mathcal{K}$ una clase que contiene finitas estructuras módulo isomorfismos, $\mathcal{K}$ admite un orden* FO*-definible.*

Demostración. Sea $k = max\{|A| \mid \mathcal{A} \in \mathcal{K}\}$, k es una cota para $\mathcal{K}$. □

El siguiente ejemplo muestra que no siempre puede definirse un orden.

Ejemplo 3.1.7 (Los grafos completos no admiten órdenes)**.** *Sea $\mathcal{K}$ la clase de grafos* completos*. Grafos $\mathcal{G} = \langle G, E^{\mathcal{G}} \rangle$ donde todo par de vértices está conectado mediante una arista. Supongamos que una fórmula $\varphi(\bar{t}, x, y)$ define un orden sobre $\mathcal{K}$ con k parámetros. Como para todo par de vértices u, v vale Euv, en un grafo completo cualquier permutación de los vértices resulta un automorfismo. Sea $n = k+2$, $\mathcal{K}_n$ el grafo completo de n vértices y $\bar{p} \in K_n^k$ tal*

que $\varphi(\bar{p}, x, y)$ define un orden lineal en $\mathcal{K}_n$. Tomemos dos vértices u, v que no hayan sido tomados como parámetros. Definimos el automorfismo f que permuta dos vértices distintos u y v y deja fijo a los demás vértices. Luego,

$$\mathcal{G} \models \varphi(\bar{p}, u, v) \quad sii \quad \mathcal{G} \models \varphi(f(\bar{p}), f(u), f(v)) \quad sii \quad \mathcal{G} \models \varphi(\bar{p}, v, u).$$

Lo que es absurdo pues supusimos que φ define un orden lineal.

El ejemplo anterior, por un lado, muestra los límites de este método. La clase de grafos completos es, en algún sentido, simple y sin embargo no podemos definir un orden sobre ésta. Esto lo trataremos más adelante introduciendo otras herramientas. Por otro lado, sugiere una relación entre la cantidad de automorfismos que admiten las estructuras de una determinada clase y la posibilidad de definir órdenes sobre ésta. Estudiaremos esto más en profundidad en la siguiente sección.

3.2. Estructuras rígidas

Una estructura se dice *rígida* si el único automorfismo que admite es la identidad. La pregunta de si la rigidez es una condición suficiente para poder definir un orden mediante $\mathsf{FO}(\mathsf{IFP})$ en una clase de estructuras, fue planteada por Dawar [6], quien conjeturó que esto era cierto. Gurevich [18] refuta ésto, construyendo una familia de estructuras rígidas, en la cual ningún orden es definible, ni siquiera con una $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$-fórmula. Una condición más fuerte debe pedírsele a una clase de estructuras para poder garantizar que se puede definir un órden sobre ésta.

3.2.1. Orden de tipos

Recordemos que FO^k denota el fragmento de FO que consiste de fórmulas con variables entre $x_1 ... x_k$.

Definicion 3.2.1. Dada una estructura $\mathcal{A}$ y una tupla $(a_1, a_2, ..., a_l) \in A^l$, con $l \leq k$. Llamamos *k-tipo* de $(a_1, a_2, ..., a_l)$ al conjunto de FO^k-fórmulas φ tales que $\mathcal{A} \models \varphi(a_1, a_2, ..., a_l)$. Si nos restringimos a fórmulas libres de cuantificadores lo llamamos *k-tipo básico*.

Recordemos los juegos con fichas definidos en la Sección 1.4.2. La Proposición 1.4.6 asegura que que $\bar{a}$ y $\bar{b}$ tienen el mismo k-tipo si $\mathsf{Duplicator}$ tiene una estrategia para $\mathcal{G}^k_\infty(\mathcal{A}\bar{a}, \mathcal{A}\bar{b})$, el juego jugado en dos copias de $\mathcal{A}$. Como consecuencia de esto, si $\bar{a}$ y $\bar{b}$ tienen el mismo k-tipo, también satisfacen las mismas $\mathcal{L}^k_{\infty\omega}$-fórmulas.

Observemos que dado un vocubulario finito σ, sólo hay (módulo equivalencia) finitas $\mathsf{FO}^k[\sigma]$-fórmulas, libres de cuantificadores, distintas. Por lo cual sólo hay finitos k-tipos básicos posibles y cada uno queda definido por una fórmula. Ordenemos éstas como $\alpha_1(\bar{x}), ..., \alpha_n(\bar{x})$.

Definiremos un orden en los k-tipos de k-tuplas inductivamente. El orden de los tipos básicos viene dado por el que le dimos a las fórmulas que los definen, es decir mediante

$$\theta_0(\bar{x}, \bar{y}) := \bigvee_{1 \leq i < j \leq n} (\alpha_i(\bar{x}) \wedge \alpha_j(\bar{y})). \tag{3.3}$$

Extenderemos este orden a todos los tipos (no sólo los básicos) basándonos en los métodos de coloreo de tuplas definidos en el libro de Immerman [21] y sus formulaciones en $\mathsf{FO}(\mathsf{IFP})$ dadas por Dawar [7]. Definimos inductivamente una relación R de aridad $2k$, que será un orden estricto, parcialmente definido. La siguiente fórmula expresa que dos tuplas no son distinguibles por el orden que define R.

$$\begin{aligned}\varphi_{eq}(x_1, ..., x_k, y_1, ..., y_k) := \neg R(x_1, ..., x_k, y_1, ..., y_k) \wedge \\ \neg R(y_1, ..., y_k, x_1, ..., x_k).\end{aligned}$$

Extenderemos el orden de manera que en el paso j-ésimo, exactamente las tuplas tales que Spoiler gana el juego en $j-1$ pasos ya estén ordenadas. En el primer paso queremos que valga $R\bar{a}\bar{b}$ si y sólo si $\theta_0(\bar{a}, \bar{b})$. Es decir, comenzamos ordenando sólo tipos básicos.

El paso inductivo es el siguiente. Sean $\bar{a}$ y $\bar{b}$ tales que Spoiler gana el juego en j pasos y no están ordenadas aún en el paso j. Sea i ficha de menor índice que puede mover Spoiler para ganar en las $j-1$ jugadas siguientes. La fórmula δ_i expresa esa propiedad. Notaremos, para cada i fijo, con $x_1, ..., x, ..., x_k$, a la tupla que resulta de cambiar x_i por x en $x_1, ..., x_k$. Entonces,

$$\delta_i(\bar{x}, \bar{y}) := \bigwedge_{j<i} \beta_j(\bar{x}, \bar{y}) \wedge \neg\beta_i(\bar{x}, \bar{y}),$$

donde

$$\begin{aligned}\beta_i(\bar{x}, \bar{y}) := \forall x \exists y \varphi_{eq}(x_1, ..., x, ..., x_k, y_1, ..., y, ..., y_k) \wedge \\ \forall y \exists x \varphi_{eq}(x_1, ..., x, ..., x_k, y_1, ..., y, ..., y_k).\end{aligned}$$

Llamemos $M_i(\bar{a})$, al conjunto formado por todas las tuplas que resultan de cambiar el lugar i-ésimo de $\bar{a}$. Consideremos el conjunto de todas las posibles jugadas de Spoiler, que consisten en mover la ficha i-ésima y le permiten

ganar en las siguientes $j-1$ movidas. Es decir, el conjunto de tuplas de $M_i(\bar{a})$ que son distinguibles (en el orden parcial definido hasta ahora) de cualquiera tupla de $M_i(\bar{b})$ y viceversa. Llamémoslo S_i.

Tomemos un elemento minimal en S_i (con el orden definido hasta este paso). Supongamos que es $(a_1, ..., \tilde{a}, ..., a_k)$. Esta tupla debe ser menor que todas las de $M_i(\bar{b})$, pues justamente supusimos que era comparable con todas. Por lo que, el menor elemento de S_i se corresponde, o bien con uno de $M_i(\bar{a})$, o bien con uno de $M_i(\bar{b})$. De si se da uno u otro caso dependerá el orden relativo de $\bar{a}$ y $\bar{b}$. Las siguientes fórmulas identifican los $\tilde{a}$ o $\tilde{b}$ que dan lugar a los elementos de S_i.

$$\begin{aligned}\sigma_i^1(x, \bar{x}, \bar{y}) := \forall y(&R(x_1, ..., x, ..., x_k, y_1, ..., y, ..., y_k)\\ &\vee R(y_1, ..., y, ..., y_k, x_1, ..., x, ..., x_k)),\\ \sigma_i^2(y, \bar{x}, \bar{y}) := \forall x(&R(x_1, ..., x, ..., x_k, y_1, ..., y, ..., y_k)\\ &\vee R(y_1, ..., y, ..., y_k, x_1, ..., x, ..., x_k)).\end{aligned}$$

Formalizando lo dicho anteriormente, la iteración es definida por:

$$\begin{aligned}\theta(R, \bar{x}, \bar{y}) := \theta_0(\bar{x}, \bar{y}) \vee (\neg R\bar{x}\bar{y} \wedge &\left(\bigvee_{1\leq i\leq k} \delta_i(\bar{x}, \bar{y})\right)\\ &\wedge(\bigvee_{1\leq i\leq k} (\delta_i(\bar{x}, \bar{y}) \wedge \exists x(\sigma_i^1(x, \bar{x}, \bar{y})\\ \wedge\forall y(\sigma_i^2(y, \bar{x}, \bar{y})) \rightarrow R(x_1, ..., x, ..., x_k, y_1, ..., y, ..., y_k))))).\end{aligned}$$

Y por lo tanto el orden de tipos viene dado por la FO(IFP)-fórmula:

$$\varphi_{k-ord}(\bar{x}, \bar{y}) := \mathsf{IFP}\,[R\bar{x}\bar{y} \leftarrow \theta(R, \bar{x}, \bar{y})]\,(\bar{x}\bar{y}). \tag{3.4}$$

Notemos con $<_k$ a esta relación y $\leq_k$ a su clausura reflexiva. Veamos que efectivamente este método define un preorden en los tipos, es decir, que vale la siguiente proposición.

Proposicion 3.2.2. *Valen las siguientes afirmaciones:*

(a) El tipo de $\bar{a}$ es el mismo que el de $\bar{b}$ si y sólo si $\bar{a} \leq_k \bar{b}$ y $\bar{b} \leq_k \bar{a}$.

(b) Si $\bar{a} \leq_k \bar{b}$ y $\bar{b} \leq_k \bar{c}$ entonces $\bar{a} \leq_k \bar{c}$.

Demostración. Veamos primero (a). Basta ver que, si $\bar{a}$ y $\bar{a}'$ tienen el mismo tipo, entonces para todo $\bar{b}$, $\bar{a} \leq_k \bar{b}$ si y sólo si $\bar{a}' \leq_k \bar{b}$. Sean $\bar{a} \leq_k \bar{b}$. Supongamos que fueron ordenados en el paso j. Eso quiere decir que Spoiler gana el juego $\mathcal{G}_j^k(\mathcal{A}\bar{a}, \mathcal{A}\bar{b})$. Como $\bar{a}$ y $\bar{a}'$ tienen el mismo tipo, Duplicator gana el

juego comenzando con dichas tuplas. Luego Spoiler debe ganar $\mathcal{G}_j^k(\mathcal{A}\bar{a}', \mathcal{A}\bar{b})$, pues, de lo contrario, Duplicator ganaría $\mathcal{G}_j^k(\mathcal{A}\bar{a}, \mathcal{A}\bar{b})$ (ver Corolario 1.2.3).

Veamos (b), es decir, que la relación definida es transitiva. Basta ver que lo es en cada paso de la iteración. Es claro para el primero (el de los tipos básicos). Los demás pasos los deducimos inductivamente.

Supongamos que valen $R\bar{a}\bar{b}$ y $R\bar{b}\bar{c}$, queremos ver que vale $R\bar{a}\bar{b}$. Por la construcción de R, existen $i, j \leq k$, $\bar{a}' \in M_i(\bar{a})$ y $\bar{b}' \in M_j(\bar{b})$, tales que para todo $\bar{b}_2 \in M_i(\bar{b})$ y todo $\bar{c}_2 \in M_i(\bar{c})$, vale $R\bar{a}'\bar{b}_2$. Supongamos además que $i < j$ (los demás casos son análogos). Como j era el menor índice tal que pueden diferenciarse $\bar{b}$ y $\bar{c}$, todo $\bar{c}' \in M_i(\bar{c})$ resulta equivalente (con el orden definido hasta ahora) a $\bar{b}'$, lo que a su vez, aplicando la parte (a), implica que vale $R\bar{a}'\bar{c}'$ para todo $\bar{c}' \in M_i(\bar{c})$. Esto prueba la transitividad. □

A continuación analizamos el caso donde el preorden definido resulta un orden.

3.2.2. Estructuras k-rígidas

Definicion 3.2.3. Sea $k \in \mathbb{N}$, una σ-estructura se dice *k-rígida* si no hay dos elementos con el mismo k-tipo.

Decimos que una clase de estructuras $\mathcal{K}$ es k-rígida, si toda $\mathcal{A} \in \mathcal{K}$ es k-rígida.

Observemos que, si una estructura es k-rígida, FO^k y $\mathcal{L}_{\infty\omega}^k$ son capaces de distinguir a cada elemento del universo.

Ejemplo 3.2.4. *(Estructura 2-rígida) A partir de Ejemplo 1.4.3 deducimos que una clase de estructuras ordenadas es 2-rígida.*

Proposicion 3.2.5. *Sea $\mathcal{K}$ una clase de estructuras. $\mathcal{K}$ es k-rígida para algún $k \in \mathbb{N}$ si y sólo si admite un orden* FO(IFP) *definible.*

Demostración. Sea $\mathcal{K}$ una clase de estructuras k-rígidas, para definir un orden basta tomar la fórmula φ_{k-ord} definida en la sección anterior. Dicha fórmula define, para cada $\mathcal{A} \in \mathcal{K}$, un orden sobre A (identificando cada $a \in A$ con $(a, ..., a)$).

Veamos la otra implicación. Sea $\varphi(x, y)$ la FO(IFP)-fórmula que define un orden sobre $\mathcal{K}$. Existe una $\mathcal{L}_{\infty\omega}^{\omega}$-fórmula $\tilde{\varphi}(x, y)$ equivalente (Proposición 1.4.4). Supongamos que utiliza m variables. Reemplazando las apariciones de $\leq$ en 1.4.3 por $\tilde{\varphi}$, podemos distinguir a cada elemento mediante una $\mathcal{L}_{\infty\omega}^{\omega}$-fórmula que utiliza $m+2$ variables, o equivalentemente $\mathcal{K}$ es $m+2$-rígida. □

Corolario 3.2.6. *Sea $\mathcal{K}$ una clase de estructuras k-rígida, entonces* FO(IFP) *captura* PTIME *sobre $\mathcal{K}$.*

Si bien toda estructura rígida y finita es k-rigida para algún $k \in \mathbb{N}$ (ver [7]), una clase de estructuras rígidas podría no ser k-rígida para ningún k. Esta es la razón por la que la rigidez no es una condición suficiente para definir un orden en la estructura. Pueden encontrarse ejemplos de clases de estructuras rígidas que no son k-rígidas para ningún $k \in \mathbb{N}$ y sin embargo, FO(IFP) captura PTIME sobre ellas [26]. Esto muestra que la condición de que un orden sea definible no es necesaria ni en clases de estructuras que no admiten automorfismos.

El Teorema de Abiteboul y Vianu

Como una aplicación de las herramientas que definimos hasta ahora, presentamos este resultado. Es una combinación de resultados de [27] y [1]. De la Proposición 2.2.7 se deduce que el problema de separar las clases de complejidad PTIME y PSPACE es equivalente al de separar las lógicas FO(IFP) y FO(PFP) sobre estructuras ordenadas. El teorema en cuestión afirma que esto es equivalente a separarlas sobre cualquier clase de estructuras.

Teorema 3.2.7 (Abiteboul y Vianu, Teorema 12.50 en [21])**.** *Son equivalentes:*

(I) PTIME = PSPACE*.*

(II) FO(IFP) $\equiv$ FO(PFP)*.*

La demostración del teorema se basa en, para cada estructura $\mathcal{A}$, la construcción de la estructura ordenada $E^k(\mathcal{A})$, el $\mathcal{L}^k$*-invariante de* $\mathcal{A}$. Consiste en la estructura formada por los k-tipos de $\mathcal{A}$ con algunos predicados adicionales que permiten rescatar información de $\mathcal{A}$ a partir de ésta. Muchas propiedades de $\mathcal{A}$ se traducen en propiedades de $E^k(\mathcal{A})$ y viceversa.

3.3. Transducciones

Las transducciones (también llamadas interpretaciones sintácticas) son una importante herramienta lógica. Permiten definir a partir de una estructura, otra, incluso en un vocabulario distinto. Dónde está definido el universo de la nueva estructura da pie a diversos tipos de transducciones (ver [24]), la que acá presentamos es la versión más general. Definiremos el nuevo universo como un subconjunto de alguna potencia del original, identificando tuplas mediante una relación de equivalencia. Antes de definir transducción precisamos fijar ciertos términos.

Definicion 3.3.1. Sea $\mathcal{A}$ una σ-estructura y $\sim$ una relación de equivalencia definida sobre A. Dada $R \in \sigma$ de aridad k, decimos que $\sim$ es *R-compatible* si para cada $\bar{a} = (a_1, a_2, ..., a_k), \bar{b} = (b_1, b_2, ..., b_k) \in A^k$ vale que $\bar{a} \sim \bar{b}$ implica:

$$R\bar{a} \quad \text{sii} \quad R\bar{b}. \tag{3.5}$$

Decimos que es *compatible* si lo es para cada $R \in \sigma$.

La condición de compatibilidad es la necesaria para poder definir una σ-estructura en el cociente por la relación $\sim$.

Definicion 3.3.2. Sean σ y τ dos vocabularios, y $\mathcal{L}$ una lógica. Una $\mathcal{L}[\sigma, \tau]$-transducción, Θ, de *dimensión* s y con t *parámetros* es una tupla

$$\Theta = (\theta_p(\bar{x}), \theta_{uni}(\bar{x}, \bar{y}), \theta_{\equiv}(\bar{x}, \bar{y}_1, \bar{y}_2), \{\theta_R(\bar{x}, \bar{y}_R)\}_{R \in \tau}), \tag{3.6}$$

de $\mathcal{L}[\sigma]$-fórmulas, donde

- $\bar{y}$, $\bar{y}_1$ e $\bar{y}_2$ son s-tuplas de variables del mismo tipo,
- $\bar{x}$ es una t-tupla de variables individuales,
- para cada relación R de aridad k, $\bar{y}_R := \bar{y}_{R,1}, \bar{y}_{R,2}$ ... $\bar{y}_{R,k}$, con $\bar{y}_{R,i}$ una tupla de variables de igual longitud y tipo que $\bar{y}$.

A $\bar{y}$ las llamamos *variables de universo* y a $\bar{x}$ *parámetros*. Pedimos además que, para cada estructura $\mathcal{A}$ y cada $p \in A^{\bar{x}}$, $\theta_{\equiv}(\bar{p}, \bar{y}_1, \bar{y}_2)$ defina[1] una relación de equivalencia compatible en $A^{\bar{x}}$. Es decir, satisface las fórmulas:

$$\left(\bigwedge_{1 \leq i \leq k} \theta_{\equiv}(\bar{x}, \bar{y}_{R,i}, \bar{z}_{R,i}) \right) \rightarrow (\theta_R(\bar{x}, \bar{y}_R) \leftrightarrow \theta_R(\bar{x}, \bar{z}_R)). \tag{3.7}$$

Sea Θ una transducción con t parámetros y de dimensión s, llamamos *dominio* de Θ a la clase de pares $(\mathcal{A}, \bar{p})$, tales que $\mathcal{A} \models \theta_p(\bar{p})$ y lo notamos $\mathcal{D}_\Theta$.

La transducción Θ define una aplicación de σ-estructuras en τ-estructuras (que también notaremos Θ):

$$\begin{aligned} \mathcal{D}_\Theta &\rightarrow \text{Struc}[\tau] \\ (\mathcal{A}, \bar{p}) &\mapsto \Theta(\mathcal{A}, \bar{p}), \end{aligned}$$

[1] Si la lógica tiene la capacidad de expresar la relación de equivalencia generada por una relación (como es el caso de FO(IFP) o FO(IFP) + C), el requisito de que $\theta_\equiv$ la defina puede cambiarse por que la genere.

donde $\Theta(\mathcal{A}, \bar{p})$ es la τ-estructura de universo:

$$V(\Theta(\mathcal{A}, \bar{p})) := \{\bar{a} \in A^{\bar{y}} \mid \mathcal{A} \models \theta_{uni}(\bar{p}, \bar{a})\}/\equiv \tag{3.8}$$

con $\equiv$ la relación definida por $\theta_{\equiv}$.

Con las relaciones definidas por las fórmulas correspondientes. La fórmula 3.7 garantiza la buena definición de cada relación. En el caso en que el universo de la estructura es directamente alguna potencia del original (sin ninguna identificación), decimos que la transducción es *directa* y en ese caso obviaremos la fórmula correspondiente a la relación de equivalencia en la definición.

Veamos algunos ejemplos.

Ejemplo 3.3.3 (Grafo lineal)**.** *Dado un grafo $\mathcal{G} = \langle G, E^{\mathcal{G}} \rangle$, su* grafo lineal *es el que tiene como vértices a las aristas de $\mathcal{G}$, donde dos aristas son adyacentes, si tienen un vértice en común. La siguiente $\mathsf{FO}[\{E\}, \{E\}]$-transducción de dimensión dos y sin parámetros asigna a cada grafo su grafo lineal correspondiente.*

$$\begin{aligned}
&\theta_p := \top,\\
&\theta_{uni}(x_1, x_2) := Ex_1x_2,\\
&\theta_{\equiv}(y_1, y_2, z_1, z_2) := (y_1 = z_2) \wedge (y_2 = z_1),\\
&\theta_E := (y_1, y_2, z_1, z_2) := ((y_1 = z_1) \wedge (y_2 \neq z_2)) \vee ((y_1 = z_2) \wedge (y_2 \neq z_1))\\
&\vee ((y_2 = z_1) \wedge (y_1 \neq z_2)) \vee ((y_2 = z_2) \wedge (y_1 \neq z_1)).
\end{aligned}$$

La relación $\equiv$ identifica a dos pares que representan la misma arista y θ_E expresa "son aristas distintas que comparten un vértice".

Ejemplo 3.3.4 (Supresión de k vértices)**.** *Sea $k \in \mathbb{N}$ y $\bar{x}$ de longitud k. Consideremos la $\mathsf{FO}[\{E\}, \{E\}]$-transducción dada por:*

$$\begin{aligned}
&\theta_p(\bar{x}) := \top,\\
&\theta_{uni}(\bar{x}, y) := \bigwedge_{1 \leq i \leq k} (y \neq x_i),\\
&\theta_E(y, z) := Eyz.
\end{aligned}$$

En este caso dado un grafo $\mathcal{G}$, y vértices $v_1 ... v_k$, $\Theta(\mathcal{G}, (v_1, v_2, ..., v_k))$ es el grafo que resulta de suprimir dichos vértices.

Ejemplo 3.3.5 (Componentes conexas)**.** *Sea $\varphi_{camino}(x, y)$ la* FO(IFP)*-fórmula que interpretada sobre un grafo, expresa "existe un camino de x a y" La siguiente* FO(IFP)$[\{E\}, \{E\}]$*-transducción, con un parámetro x, asigna a cada grafo $\mathcal{G}$ la componente conexa de x.*

$$\theta_p(x) := \top,$$
$$\theta_{uni}(x, y) := \varphi_{camino}(x, y),$$
$$\theta_E(x, y_1, y_2) := Ey_1y_2.$$

Ejemplo 3.3.6 (Árbol dirigido)**.** *Definiremos una transducción que a cada árbol $\mathcal{T}$ y $t_0 \in T$ le asigna una copia dirigida de $\mathcal{T}$, que tiene a t_0 como raiz. La fórmula $\varphi(x, y_1, y_2)$ es la definida en el Ejemplo 3.1.3, que dirige la relación de adyacencia desde x.*

$$\theta_p(x) := \top,$$
$$\theta_{uni}(x, y) := \top,$$
$$\theta_E(x, y_1, y_2) := \varphi(x, y_1, y_2).$$

Observemos que si bien en la definición de transducción permitimos que las variables de universo no sean todas del mismo tipo en los ejemplos usuales (y todos los que daremos) lo son. Si son todas de tipo n decimos que la transducción es *numérica*.

3.3.1. Transducciones y complejidad

Una $[\sigma, \tau]$-transducción Θ respeta isomorfismos, es decir, $\mathcal{A} \approx \mathcal{B}$, implica que $\Theta(\mathcal{A}) \approx \Theta(\mathcal{B})$. Si $f : \mathcal{A} \to \mathcal{B}$ es una isomorfismo, construimos la función $\tilde{f^k}$ que determina el isomorfismo entre $\Theta(\mathcal{A})$ y $\Theta(\mathcal{B})$ como indica el siguiente diagrama.

$$\begin{array}{ccc} A^k & \xrightarrow{f^k} & B^k \\ {\scriptstyle\pi}\downarrow & & \downarrow{\scriptstyle\pi} \\ A^k/\equiv & \xrightarrow[\tilde{f^k}]{} & B^k/\equiv \end{array}$$

Con π denotamos a la proyección al cociente y f^k la función que resulta de aplicar f en cada coordenada. La buena definición de $\tilde{f^k}$ y el hecho de que resulta un isomorfismo, son consecuencias directas de que la relación $\models$ respeta isomorfismos. Deducimos entonces que Θ define una aplicación de

σ-propiedades en τ-propiedades. Fijemos una σ-propiedad $\mathcal{P}$ y supongamos $\Theta(\mathcal{P}) \in \mathsf{PTIME}$, podemos construir una máquina que decida $\Theta^{-1}(\Theta(\mathcal{P}))$ (la preimagen de $\Theta(\mathcal{P})$) de la siguiente manera. Notemos primero que dada una estructura $\mathcal{A}$, el tamaño de $\Theta(\mathcal{A})$ es a lo sumo una potencia del original. A partir de cada relación en $\mathcal{A}$, podemos construir cada una de las de $\Theta(\mathcal{A})$, simulando las $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$-fórmulas que las definen (recorriendo cada una de las tuplas de la correspondiente longitud). Una vez que tenemos una copia de $\Theta(\mathcal{A})$, corremos la máquina que decide $\Theta(\mathcal{P})$.

Definicion 3.3.7. Decimos que una $[\sigma, \tau]$-transducción es *fiel*, si para todo par de estructuras $\mathcal{A}$ y $\mathcal{B}$, vale que

$$\Theta(\mathcal{A}) \approx \Theta(\mathcal{B}) \quad \text{sii} \quad \mathcal{A} \approx \mathcal{B}. \tag{3.9}$$

Pedirle a una transducción que sea fiel es equivalente a pedirle que para toda propiedad $\mathcal{P}$, $\Theta^{-1}(\Theta(\mathcal{P})) = \mathcal{P}$. De lo que se deduce la siguiente proposición.

Proposicion 3.3.8. *Sea Θ una $[\sigma, \tau]$-transducción fiel y $\mathcal{P}$ una propiedad de σ-estructuras. Si $\Theta(\mathcal{P}) \in \mathsf{PTIME}$, entonces $\mathcal{P} \in \mathsf{PTIME}$.*

3.3.2. Todo es un grafo

La mayoría de los ejemplos y resultados que presentamos son para clases de grafos. A continuación vamos a mostrar que estos son lo suficientemente generales.

Veremos cómo, dada una estructura cualquiera, podemos codificarla mediante un grafo. Es decir, podemos construir, en tiempo polinomial, un grafo que codifique todas las relaciones de la estructura original, y, recíprocamente, dado el grafo que codifica la estructura podemos reconstruir ésta en tiempo polinomial.

Fijemos un lenguaje σ, vamos a mostrar la construcción para el caso de $\sigma = \{T, R\}$, con T unaria y R binaria, ya que alcanza para entender en qué consiste y cómo adaptarla a otros vocabularios. Escribir el caso general, además de tedioso para el lector, lo haría parecer más complicado de lo que en verdad es.

Sea $\mathcal{A}$ una estructura de universo de cardinal n. La idea es construir el grafo con ciertos vértices distinguibles que representan al universo de $\mathcal{A}$ y "marcarlos" con subgrafos completos, de tamaños distintos, para indicar cada tupla en una relación. Obviamente, para esto necesitaremos mas de $|A|$ vértices. En el caso particular que analizaremos, los vértices del grafo serán elementos de A^4. Consideremos el siguiente subconjunto de A^4:

(1) Para cada $a \in A$, (a, a, a, a),

(2) para cada $a \in A$, el conjunto $\{(a, b, a, a) \mid b \neq a\}$,

(3) para cada $(a, b) \in R^{\mathcal{A}}$ con $a \neq b$, los conjuntos $\{(a, c, b, a) \mid c \neq a\}$ y $\{(a, b, c, a) \mid c \notin \{a, b\}\}$,

(4) para cada $(a, a) \in R^{\mathcal{A}}$, el conjunto $\{(a, a, b, a) \mid b \neq a\}$,

(5) para cada $a \in T^{\mathcal{A}}$, el conjunto $\{(a, a, b, c) \mid b \neq a\}$.

Usaremos los elementos en (1) como una copia de A, para reconocerlos los hacemos adyacentes a los respectivos elementos de (2). De esa manera, los vértices de la estructura original serán los únicos que tienen $n-1$ vértices adyacentes que no están a su vez relacionados con otros vértices.

Los de (3) los usamos para marcar los pares (distintos) que están en la relación, formando un grafo completo entre todos los $2n-3$ vértices. Conectamos a a con los primeros $n-1$ y b con los otros $n-2$. De esa manera Rab lo codificamos como: "a es un elemento de un subgrafo completo maximal (no incluido en otro) de n elementos y b de uno de $n-1$ elementos que a su vez forman uno de $2n-3$ elementos".

Usamos los vértices de (4) para el caso de elementos relacionados consigo mismo, formando, para cada a tal que vale Raa, un subgrafo completo de $n-1$ elementos, que conectamos a a.

Por último usamos los vértices de (5) para T, formando, para cada a tal que vale Ta, un subgrafo completo de $(n-1)^2$ elementos que conectamos a a.

Observacion 3.3.9. La construcción anterior presupone que las estructuras tienen un universo de cardinal mayor a dos. Para que funcione correctamente para cualquier estructura hay que tratar los casos "pequeños" aparte.

Proposicion 3.3.10 (Todo es un grafo)**.** *Dado un lenguaje σ, existen una $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}[\sigma, \{E\}]$-transducción Γ y una $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}[\{E\}, \sigma]$-transducción Λ, ambas sin parámetros, tales que para cada σ-estructura $\mathcal{A}$:*

(I) $\Gamma(\mathcal{A})$ *es un grafo.*

(II) $\Lambda(\Gamma(\mathcal{A})) \approx \mathcal{A}$.

Demostración. Claramente la construcción dada anteriormente puede expresarse mediante fórmulas. Necesitamos poder contar, porque el tamaño de

los subgrafos completos que queremos detectar depende del tamaño de la estructura. Entonces para el caso de Γ,

$$\begin{aligned}\gamma_{uni}(x_1,x_2,x_3,x_4) := (x_1 = x_2 = x_3 = x_4) \vee \\ ((x_1 = x_3 = x_4) \wedge x_2 \neq x_1)) \vee \\ ((x_1 = x_4) \wedge (x_2 \neq x_1) \wedge Rx_1x_3) \vee \\ ((x_1 = x_4) \wedge (x_3 \neq x_1) \wedge (x_3 \neq x_2) \wedge Rx_1x_2) \vee \\ ((x_1 = x_2 = x_3) \wedge (x_2 \neq x_1) \wedge Rx_1x_1) \vee \\ ((x_1 = x_2) \wedge (x_3 \neq x_1) \wedge (x_3 \neq x_2).\end{aligned}$$

Para definir γ_E hay que tener en cuenta de cuales de las tuplas definidas en la construcción se trata (depende de que disyunciones de γ_{uni} satisfagan) y seguir las reglas dadas en la construcción.

En el caso de Λ,

$$\lambda_{uni}(x) := \#y(Exy \forall z(Eyz \rightarrow z = x)) = n - 1,$$

donde $n-1$ se refiere al anteúltimo elemento de $[0, |A|]$, definible en $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$. Para λ_R y λ_T hay que decodificar el grafo, es decir, verificar que los vértices pertenezcan a subgrafos con las características mencionadas, dichas características son fácilmente expresables mediante $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$-fórmulas. Lo que finaliza la construcción de las transducciones que buscábamos. □

Observacion 3.3.11. Γ y Λ son, más precisamente, $\mathsf{FO} + \mathsf{C}$ transducciones.

Ejemplo 3.3.12. *Sea $\mathcal{A}$ con $|A| = 4$. La Figura 3.3 muestra los nodos correspondientes a dos elementos a y b tales que vale Rab y Tb.*

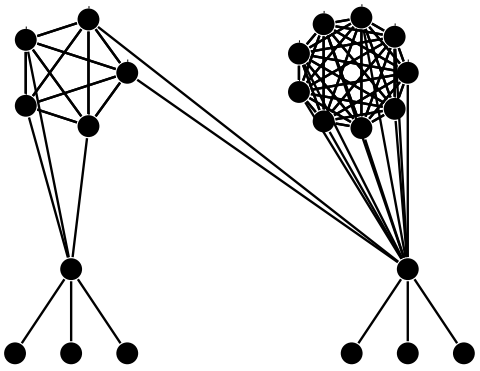


Figura 3.3: *Rab* y *Tb*

Notemos que, $\Lambda(\Gamma(\mathcal{A})) \approx \mathcal{A}$ implica que Γ es fiel. Para que Λ también sea fiel, precisamos restringirla a la clase de grafos que codifican estructuras, es decir, la imagen de Γ. Esa clase es definible por una $\mathsf{FO(IFP)} + \mathsf{C}$ fórmula digamos φ_Γ (la fórmula expresa que existen determinada cantidad de vértices, subgrafos completos y las relaciones de adyacencia adecuadas). Si modificamos Λ agregando la fórmula

$$\lambda_p := \varphi_\Gamma, \tag{3.10}$$

resulta fiel. Aplicando la Proposición 3.3.8, obtenemos una correspondencia entre las propiedades PTIME de cualquier clase de estructuras y las de la clase correspondiente de grafos. Es más, dada una propiedad $\mathcal{P}$, podemos dar un algoritmo para decidirla a partir de Γ y uno para $\Gamma(\mathcal{P})$.

3.3.3. Transducciones y definibilidad

El siguiente lema muestra que una transducción no solo establece una relación entre estructuras, sino también entre fórmulas. El caso de $\mathsf{FO(IFP)} + \mathsf{C}$-fórmulas requiere particular atención. Por un lado debido a que el universo de la nueva estructura puede ser más grande que el de la original y por otro, el hecho de que las transducciones numéricas tienen como variables de vértices, variables numéricas. Para que la exposición sea más clara, lo enunciamos y demostramos para $\mathsf{FO(IFP)}$ (y por lo tanto para FO) y $\mathsf{FO(IFP)} + \mathsf{C}$ de forma separada. Si quisieramos una versión para una lógica abstracta, tendríamos que pedirle ciertas condiciones de regularidad a ésta (ver [8]).

Lema 3.3.13 (de la transducción para $\mathsf{FO(IFP)}$)**.** *Sea Θ una $\mathsf{FO(IFP)}[\sigma,\tau]$-transducción de dimensión s con t parámetros. Para cada τ-fórmula $\varphi(z_1, z_2, ..., z_k)$ existe una σ-fórmula $\psi(\bar{x}, \bar{y}_1, \bar{y}_2, ..., \bar{y}_k)$, tal que para toda $(\mathcal{A}, \bar{p}) \in \mathcal{D}_\Theta$ y todo $\bar{a}_1, \bar{a}_2, ..., \bar{a}_k \in A^s$:*

$$\mathcal{A} \models \psi(\bar{p}, \bar{a}_1, \bar{a}_2, ..., \bar{a}_k) \quad sii \quad \{[\bar{a}_i]\}_{i\leq k} \subset V(\Theta(\mathcal{A}, \bar{p})) \quad y \quad \Theta(\mathcal{A}, \bar{p}) \models \varphi([\bar{a}_1], [\bar{a}_2], ..., [\bar{a}_k]).$$

Demostración. Dada φ construimos ψ inductivamente.

- $\varphi = z_1 = z_2$, definimos $\psi := \theta_{uni}(\bar{x}, \bar{y}_1) \wedge \theta_{uni}(\bar{x}, \bar{y}_2) \wedge \theta_{\equiv}(\bar{x}, \bar{y}_1, \bar{y}_2)$.
- $\varphi = Rz_1...z_k$, $\psi := \left(\bigwedge_{i\leq k} \theta_{uni}(\bar{x}, \bar{y}_k)\right) \wedge \theta_R(\bar{x}, \bar{y}_1...\bar{y}_k)$.
- $\varphi = \neg\varphi'(z_1, ..., z_k), \psi := \neg\psi'(\bar{y}_1, ..., \bar{y}_k)$.
- $\varphi = \varphi_1(z_1, ..., z_k) \vee \varphi_2(z'_1, ..., z'_k)$, $\psi := \psi_1(\bar{y}_1, ..., \bar{y}_k) \vee \psi_2(\bar{y'}_1, ..., \bar{y'}_k)$.

- $\varphi = \exists z \varphi'(z_1, ..., z_k, z)$, $\psi := \exists \bar{y}(\theta_{uni}(\bar{y}) \wedge \psi'(\bar{y}_1, ..., \bar{y}_k, \bar{y}))$.
- $\varphi = \mathsf{IFP}\left[Z^k \bar{z} \leftarrow \varphi'(\bar{z})\right](\bar{t})$,
  $\psi := \mathsf{IFP}\left[Y^{k\times s} \bar{y}_1, ..., \bar{y}_k \leftarrow \psi'(\bar{y}_1, ..., \bar{y}_k)\right](\bar{t'}_1, ..., \bar{t'}_k)$.

□

Lema 3.3.14 (De la transducción para $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$)**.** *Sea Θ una $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}[\sigma, \tau]$-transducción de dimensión s con t parámetros. Para cada τ-fórmula $\varphi(z_1, z_2, ..., z_k)$ existe una σ-fórmula $\psi(\bar{x}, \bar{y}_1, \bar{y}_2, ..., \bar{y}_k)$, con $\bar{y}_i$ numérica en el caso de una transducción numérica y del mismo tipo que z_i en el caso contrario, tal que para toda $(\mathcal{A}, \bar{p}) \in \mathcal{D}_\Theta$ y $\bar{a}_1, \bar{a}_2, ..., \bar{a}_k \in A^s \sqcup [0, |A|]^s$:*

$$\mathcal{A} \models \psi(\bar{p}, \bar{a}_1, \bar{a}_2, ..., \bar{a}_k) \quad sii \quad \{[\bar{a}_i]\}_{i\leq k} \subset V(\Theta(\mathcal{A}, \bar{p})) \quad y \quad \Theta(\mathcal{A}, \bar{p}) \models \varphi([\bar{a}_1], [\bar{a}_2], ..., [\bar{a}_k]).$$

En el caso de $\bar{a}_i \in [0, |A|]^s$, $[\bar{a}_i]$ denota $[\bar{a}_i]_n$.

Demostración. Veamos el caso

$$\varphi = \#z\varphi'(z) = i. \tag{3.11}$$

Construimos $\psi := \frac{\#}{\theta_{\equiv}} \bar{y} \psi'(\bar{y}) = (i_1, i_2, ..., i_s)$. El significado de $\frac{\#}{\theta_{\equiv}}$ es el del Ejemplo 1.6.5. Los demás casos fueron tratados en la proposición anterior (traduciendo las variables como explica el enunciado de este lema). □

Si bien el siguiente corolario no es más que un caso particular del lema, vale la pena enunciarlo, ya que es el que utilizaremos y su enunciación es más amena.

Corolario 3.3.15 (transducción para sentencias)**.** *Sea Θ una $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}[\sigma, \tau]$-transducción de dimensión s con t parámetros. Para cada τ-sentencia φ, existe una σ-fórmula $\psi(\bar{x})$ tal que para cada $(\mathcal{A}, \bar{p}) \in \mathcal{D}_\Theta$ vale:*

$$\mathcal{A} \models \psi(\bar{p}) \quad sii \quad \Theta(\mathcal{A}, \bar{p}) \models \varphi. \tag{3.12}$$

El Lema 3.3.14 muestra que una $\mathcal{L}[\sigma, \tau]$-transducción Θ, no sólo define una aplicación de σ-estructuras a τ-estructuras, sino también una de τ-fórmulas σ-fórmulas. Llamamos *dual* de Θ a dicha aplicación y la notamos $\hat{\Theta}$.

Ejemplo 3.3.16 (k-conexión)**.** *Un grafo $\mathcal{G}$ se dice k-conexo si $|V| \geq k$ y para cualquier elección de $k-1$ vértices, el subgrafo que resulta de suprimir estos vértices es conexo. Sea Θ la transducción definida en el ejemplo 3.3.4 y φ_{con} la $\mathsf{FO}(\mathsf{IFP})$-fórmula que expresa que un grafo es conexo (ejemplo 1.3.5).*

Luego $\mathcal{G} \models \hat{\Theta}(\varphi_{con}(v_1, ..., v_k))$ *si suprimiendo* $(v_1, ..., v_k)$ *el grafo es conexo. Entonces,*

$$(\exists x_1 x_2 ... x_k \left(\bigwedge_{i \neq j} x_i \neq x_j \right)) \wedge (\forall x_1 x_2 ... x_k \hat{\Theta}(\varphi_{con}(x_1, ..., x_k))) \tag{3.13}$$

define la clase de grafos $k+1$*-conexos.*

Observacion 3.3.17. Es fácil extender el resultado anterior a otras lógicas más expresivas que FO(IFP) como por ejemplo FO(PFP). Es decir, una FO(IFP)-transducción Θ, define también una aplicación de FO(PFP)-fórmulas en FO(PFP)-fórmulas. Si quisieramos, en cambio, aplicarlo a una fórmula en una lógica menos expresiva (como FO), $\hat{\Theta}(\varphi)$ resultaría una FO(IFP)-fórmula, no una FO-fórmula.

Si juntamos estos lemas con los resultados de la sección 3.3.2 obtenemos la siguiente proposición, que profundiza el significado de "todo es un grafo".

Proposicion 3.3.18. *Sea* $\mathcal{L}$ *una lógica en el sentido de la sección 2.3. Supongamos que* $\mathcal{L}$ *es tan expresiva como* FO(IFP) + C. *Si* $\mathcal{L}$ *captura* PTIME *sobre la clase de grafos, entonces lo hace sobre cualquier clase de estructuras.*

Demostración. Sea $\mathcal{P}$ una propiedad de σ-estructuras y Γ la transducción definida en la Sección 3.3.2. Ya vimos que $\Gamma(\mathcal{P}) \in$ PTIME. Luego existe una $\mathcal{L}$-fórmula que la define. Aplicando $\hat{\Gamma}$ tenemos una $\mathcal{L}$-fórmula que define $\mathcal{P}$. □

3.3.4. Composición de transducciones

Para que la definición sea más clara supondremos ambas transducciones de dimensión uno. El caso general no tiene mayor dificultad, sólo es cuestión de escribir tuplas de la longitud adecuada. Sea Θ una $[\sigma, \tau]$-transducción con s parámetros y Υ una $[\tau, \nu]$-transducción con t parámetros, definiremos la $[\sigma, \nu]$-transducción $\Lambda := \Upsilon \circ \Theta$, con $s + t$ parámetros, de manera que $\Upsilon \circ \Theta(\mathcal{A}, \bar{p}, \bar{q})$ resulte $\Upsilon(\Theta(\mathcal{A}, \bar{p}), [\bar{q}])$. Dicha transducción viene dada por las siguientes fórmulas.

$$\lambda_p(\bar{x}, \bar{w}) := \hat{\Theta}(\upsilon_p(\bar{w})),$$
$$\lambda_{uni}(\bar{x}, \bar{w}, y) := \hat{\Theta}(\upsilon_{uni}(\bar{w}, y)),$$
$$\lambda_{\equiv}(\bar{x}, \bar{w}, y_1, y_2) := \hat{\Theta}(\upsilon_{\equiv}(\bar{w}, y_1, y_2)),$$
$$\lambda_R(\bar{x}, \bar{w}, \bar{y}) := \hat{\Theta}(\upsilon_R(\bar{w}, \bar{y})) \quad \text{para cada } R \in \nu.$$

Para que Λ esté bien definida, $\lambda_{\equiv}$ debe definir una relación de equivalencia compatible, es decir, $\bar{a}$ y $\bar{b}$ equivalentes implica que cumplen exactamente las mismas relaciones. Veámoslo por simplicidad para el caso de R unaria.

Demostración.

$$\begin{aligned} \mathcal{A} \models \lambda_{\equiv}(\bar{p}, \bar{q}, a, b) \quad \text{sii} \quad & \Theta(\mathcal{A}, \bar{p}) \models \upsilon_{\equiv}([\bar{q}], [a], [b]). \\ \text{Luego } & \Theta(\mathcal{A}, \bar{p}) \models \upsilon_R([\bar{q}], [a]) \leftrightarrow \upsilon_R([\bar{q}], [b]) \\ \text{sii} \quad & \mathcal{A} \models \hat{\Theta}(\upsilon_R(\bar{p}, \bar{q}, a)) \leftrightarrow \hat{\Theta}(\upsilon_R(\bar{p}, \bar{q}, b)) \\ \text{sii} \quad & \mathcal{A} \models \lambda_R(\bar{p}, \bar{q}, a) \leftrightarrow \lambda_R(\bar{p}, \bar{q}, b). \end{aligned}$$

□

Ejemplo 3.3.19. *Sean Θ y Υ las transducciones definidas en los Ejemplos 3.3.5 y 3.3.4 respectivamente, y φ_{k-con} la $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$-fórmula que define la $k+1$-conexión (Ejemplo 3.3.16). La fórmula $\widehat{\Upsilon \circ \Theta}(\varphi_{k-con})(x)$ expresa "la componente conexa de x es $(k+1)$-conexa".*

3.4. Canonizaciones

Como se ve en el Ejemplo 3.1.7, que una clase de estructuras sea "sencilla" no alcanza para poder definir un órden sobre ésta. Una alternativa a ésto es asociar a cada estructura, de forma canónica, una *copia ordenada*. Esto es lo que llamamos una *canonización*.

Definicion 3.4.1. Dada una σ-estructura $\mathcal{A}$, una *copia ordenada* de $\mathcal{A}$ es una estructura $\mathcal{A}' \in \mathcal{O}(\sigma \sqcup \leq)$, tal que $\mathcal{A}'|_{\sigma} \approx \mathcal{A}$.

Definicion 3.4.2. Dada una clase $\mathcal{K}$ de σ-estructuras, una *canonización* ζ de $\mathcal{K}$, es una aplicación de σ-estructuras en $\mathcal{O}(\sigma \sqcup \{\leq\})$, tal que para cada σ-estructuras $\mathcal{A}$,

(C.I) $\zeta(\mathcal{A})$ es una copia ordenada de $\mathcal{A}$,

(C.II) si $\mathcal{A} \approx \mathcal{B}$ entonces $\zeta(\mathcal{A}) \approx \zeta(\mathcal{B})$.

Ejemplo 3.4.3 (Canonización trivial)**.** *Fijemos un vocabulario σ. Recordemos la codificación de σ-estructuras definida en la Sección 2.1.2. Dada una estructura $\mathcal{A}$, el conjunto $\mathcal{A}_{\leq}$ es a su vez un conjunto ordenado (de palabras). Podríamos asignar a cada $\mathcal{A}$ el primer elemento de $\mathcal{A}_{\leq}$. Esto es un ejemplo de canonización, pero no nos resulta útil, por que no podemos (no sabemos como) computarlo en* PTIME *y por lo tanto no podemos usarlo a la hora de extender el teorema de I-V. Por eso nos concentraremos en las que son definibles mediante la lógica* $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$.

Definicion 3.4.4. Sea Θ una $\mathcal{L}[\sigma, \sigma \sqcup \{\leq\}]$-transducción. Dada una clase $\mathcal{K}$ de σ-estructuras, decimos que Θ *canoniza* $\mathcal{K}$, si para cada $\mathcal{A} \in \mathcal{K}$, existe $\bar{p} \in A^k$, tal que $(\mathcal{A}, \bar{p}) \in \mathcal{D}_\Theta$ y $\Theta(\mathcal{A}, \bar{p})$ es una copia ordenada de $\mathcal{A}$. En ese caso a Θ la llamamos una $\mathcal{L}$-canonización de $\mathcal{K}$.

A partir de ahora $\mathcal{L}$ será $\mathsf{FO(IFP)}$ o $\mathsf{FO(IFP)} + \mathsf{C}$.

Lema 3.4.5. *Sea $\mathcal{K}$ una clase de estructuras que admite una $\mathcal{L}$-canonización, entonces $\mathcal{L}$ captura* PTIME *sobre $\mathcal{K}$.*

Demostración. Queremos ver que para cada propiedad $\mathcal{P} \in \mathsf{PTIME}$, existe $\varphi_\mathcal{P}$, tal que para cada estructura $\mathcal{A}$, vale:

$$\mathcal{A} \models \varphi_\mathcal{P} \quad \text{sii} \quad \mathcal{A} \in \mathcal{P}. \tag{3.14}$$

Fijemos $\mathcal{P} \in \mathsf{PTIME}$. Aplicando el teorema de I-V obtenemos una $\sigma \sqcup \{\leq\}$-fórmula $\varphi_\leq$ que define $\mathcal{P}_\leq$. Sea Θ la transducción que canoniza a $\mathcal{K}$. Para cada $(\mathcal{A}, \bar{p}) \in \mathcal{D}_\Theta$ tenemos:

$$\begin{aligned} \Theta(\mathcal{A}, \bar{p}) \models \varphi_\leq \quad & \text{sii} \quad \Theta(\mathcal{A}, \bar{p}) \in \mathcal{P}_\leq \\ & \text{sii} \quad \Theta(\mathcal{A}, \bar{p})|_\sigma \in \mathcal{P} \\ & \text{sii} \quad \mathcal{A} \in \mathcal{P} \quad (\Theta(\mathcal{A}, \bar{p})|_\sigma \approx \mathcal{A}). \end{aligned}$$

Luego,

$$\exists \bar{x}(\theta_p(\bar{x}) \wedge \hat{\Theta}(\varphi_\leq)(\bar{x})) \tag{3.15}$$

define $\mathcal{P}$. □

La noción de canonización generaliza la de orden definible, en el sentido de la siguiente proposición.

Proposicion 3.4.6. *Si una clase $\mathcal{K}$ de estructuras admite un orden $\mathcal{L}$-definible, entonces admite una $\mathcal{L}$-canonización.*

Demostración. Sea $\varphi(\bar{t}, x, y)$ la fórmula que define un orden sobre $\mathcal{K}$. Definamos la transducción Θ:

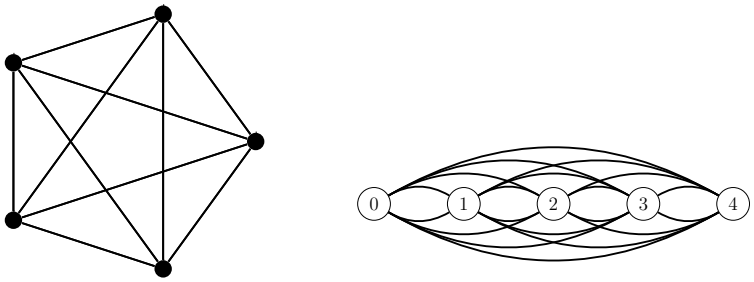
$$\begin{aligned} &\theta_p(\bar{t}) := \text{“}\varphi(\bar{t}, x, y) \quad \text{define un orden”}, \\ &\theta_{uni}(\bar{t}) := \top, \\ &\theta_{R_i}(\bar{t}, x_1, ..., x_{k_i}) := Rx_1, ..., x_{k_i}, \\ &\theta_\leq(\bar{t}, x, y) := \varphi(\bar{t}, x, y). \end{aligned}$$

□

Ejemplo 3.4.7. *(Canonización de grafos completos) En el Ejemplo 3.1.7 vimos que la clase de grafos completos no admitía ordenes. Sin embargo si admite una* $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$*-canonización, definida por la siguiente transducción* Θ *(Figura 3.4):*

$$\theta_p := \top,$$
$$\theta_{uni}(i) := \neg max(i),$$
$$\theta_E(i,j) := i \neq j,$$
$$\theta_{\leq}(i,j) := i \leqslant j.$$

Las variables i, j *son numéricas,* $max(i)$ *es la fórmula que indica que* i *es el máximo (recordar que* $|Num(A)| = |A|+1$*) y* $\leqslant$ *denota el orden en* $Num(A)$*. Es decir, el universo de* $\Theta(\mathcal{A})$ *es un segmento inicial de* $\mathbb{N}$*, que ya viene con su propio orden. Como se trata de grafos completos, para definir* E*, basta agregar una arista a cada par de números distintos. La fórmula* θ_E *no "mira" la relación de adyacencia original.*



(a) $\mathcal{K}_5$ (b) $\Theta(\mathcal{K}_5)$

Figura 3.4: Canonización de $\mathcal{K}_5$

3.4.1. Canonizaciones normales

El tipo de canonizaciones definidas en el ejemplo 3.4.7 es del tipo que más nos interesan. Están definidas por transducciones numéricas, donde el orden es el propio de $\mathbb{N}$.

Definicion 3.4.8. Decimos que una transducción Θ es *normal*, si θ_{uni} es equivalente a $\neg max(i)$ y $\theta_{\leq}$ a $i \leqslant j$. Decimos que una canonización es normal si está definida por una transducción normal.

Notemos que en el caso de canonizaciones normales, en la condición (C.II), puede cambiarse $\Theta(\mathcal{A}) \approx \Theta(\mathcal{B})$ por $\Theta(\mathcal{A}) = \Theta(\mathcal{B})$, ya que estamos considerando siempre el mismo universo. Por otro lado, si tenemos una estructura

ordenada cualquiera, existe un único isomorfismo con un segmento inicial de $\mathbb{N}$, lo que permite construir una transducción normal que copie la estructura.

Proposicion 3.4.9. *Sea σ con $\leq \in \sigma$ y $\mathcal{K}$ una clase de estructuras ordenadas. Existe una transducción normal que canoniza a $\mathcal{K}$.*

Demostración. Para construir la transducción Θ precisamos construir la biyección entre el universo de la estructura y un segmento inicial de $\mathbb{N}$. Lo hacemos inductivamente.

$$\begin{aligned}\varphi(X,i,x) :=& min(i) \wedge min(x) \vee (\exists i x' \varphi_{suc}(i',i) \\ & \wedge \varphi_{suc}(x',x) \wedge Xi'x'), \\ \varphi_{biy}(i,x) :=& \mathsf{IFP}\left[Xix \leftarrow \varphi(X,i,x)\right](ix).\end{aligned}$$

Luego para cada relación R definimos

$$\varphi_R(i_1,...,i_k) := \exists x_1,...,x_k(Rx_1...x_k) \bigwedge_{j\leq k} \varphi_{biy}(i_j,x_j). \tag{3.16}$$

El resto de las fórmulas de Θ son las usuales para una transducción normal. □

La construcción de la proposición anterior nos permite *normalizar* cualquier transducción componiendo con la recién construida.

La idea del Ejemplo 3.4.7 puede adaptarse para canonizar, de forma normal, otras clases igualmente "simples" de grafos (como las estrellas). A continuación presentamos un caso un poco más interesante, la clase de árboles.

3.4.2. Canonización de árboles

Sea $\mathfrak{T}$ la clase de árboles dirigidos (Ejemplo 1.3.7), vamos a construir una $\mathsf{FO(IFP)} + \mathsf{C}[\{E\},\{E,\leq\}$-transducción Θ, sin parámetros que canoniza a $\mathfrak{T}$. Para cada árbol $\mathcal{T}$, construiremos una copia de $\mathcal{T}$ sobre $Num(T)$, inductivamente, mediante una relación (mixta), R, de aridad tres. Dado $t \in T$, notamos con $S(t)$ al subárbol de raiz t. Construiremos R de manera que el árbol

$$\mathcal{R}_t := \langle [0,|S(t)|-1], \{(i,j)|Rtij\}\rangle,$$

sea isomorfo a $S(t)$. Para el caso de las hojas, $Rtij$ es vacía. Luego dado un vértice t, para cada hijo de t tenemos una copia ordenada de sus respectivos subárboles. Las ordenamos lexicográficamente, como algunos podrían ser isomorfos esto da lugar a un preorden, que notamos $\preceq$. Respetando ese preorden

hacemos una copia de cada uno. Finalmente agregamos un último elemento adyacente a cada una de las raíces de los subárboles copiados. Antes de dar las fórmulas necesarias vale hacer la siguiente observación.

Observacion 3.4.10. El orden lexicográfico requiere particular atención. Estamos considerano la matriz de adyacencia del grafo, concatenando cada fila de modo que se forme una tira de ceros y unos, que comparamos como si se tratara del desarrollo binario de un número. Con la particularidad de que ordenamos previamente por la longitud de la palabra. Esto es necesario para evitar ciertas ambigüedades, pues la palabra 0 y 0000 representan grafos distintos pero el mismo número.

A continuación describimos las fórmulas necesarias para definir esta relación. Definimos,

$$\varphi_S(t_0, t) := (t_0 = t) \vee \overline{E}t_0t,$$

donde $\overline{E}$ es la clausura transitiva de E (Ejemplo 1.3.4), φ_S expresa que t pertenece a $S(t_0)$. La fórmula

$$\varphi_{T<}(t_1, t_2) := \exists ij((i < j) \wedge (\#t'\varphi_S(t_1, t') = i) \wedge (\#t'\varphi_S(t_2, t') = j)),$$

compara los tamaños de los subárboles correspondientes a t_1 y t_2, necesaria para definir el orden acorde con la Observación 3.4.10. El orden de subárboles viene dado por

$$\varphi_{\preceq}(t, t_1, t_2) := Ett_1 \wedge Ett_2 \wedge (\varphi_{T<}(t_1, t_2) \vee (\varphi_{T=}(t_1, t_2) \wedge$$
$$\exists ij(\neg Rt_1ij \wedge Rt_2ij \wedge (\forall i'j'(i'j' <_{lex} ij) \rightarrow (Rt_1i'j' \leftrightarrow Rt_2i'j'))))).$$

La fórmula $\varphi_{T=}$ expresa que los subárboles son del mismo tamaño. La fórmula $\varphi_{\preceq}(t, t_1, t_2)$ expresa que t_1 y t_2 son hijos de t y $S(t_1) \preceq S(t_2)$. Usamos $<_{lex}$ para indicar el orden lexicográfico de pares. Notaremos $\varphi_{\prec}(t, t_1, t_2) := \varphi_{\preceq}(t, t_1, t_2) \wedge \neg\varphi_{\preceq}(t, t_2, t_1)$ y $\varphi_{\approx}(t, t_1, t_2) := \varphi_{\preceq}(t, t_1, t_2) \wedge \varphi_{\preceq}(t, t_2, t_1)$. Luego,

$$\varphi^{\#}_{\prec}(t_0, t, i) := \#t(\exists t_1\varphi_{\prec}(t_0, t_1, t) \wedge \varphi_S(t_1, t)) = i,$$

quiere decir que existen i vértices pertenecientes a subárboles estrictamente menores que t. La fórmula

$$\varphi_C(t_0, t, l) := \exists ijk(\varphi^{\#}_{\prec}(t_0, t, i) \wedge (\#t'\varphi_{\approx}(t_0, t, t') = j) \wedge$$
$$(\#t'\varphi_S(t, t') = k) \wedge (\exists j'(j' \leqslant j) \wedge (i + k \times j' = l))),$$

determina si a partir del lugar l, del segmento donde estamos construyendo la etapa actual, corresponde una copia del subárbol de raiz t. Como varios subárboles son isomorfos, varios l pueden satisfacer la condición. Por último

$$\varphi_r(t_0, i) := \exists t(Et_0t \wedge \exists jk(\varphi_C(t_0, t, k) \wedge (\forall l\neg Rtlj) \wedge (j + k = i))),$$

afirma que i se corresponde con un raiz.

Finalmente la fórmula que define la recursión,

$$\varphi(t,i,j) := (\exists t'(Ett' \wedge \exists i'j'k(Rt'i'j' \wedge \varphi_C(t,t',k) \wedge (i'+k=i) \\ \wedge(j'+k=j))))\vee \\ (max(i) \wedge \exists t'j'k(\varphi_r(t',i) \wedge \varphi_C(t,t',k) \wedge (j'+k=j))).$$

La $\mathsf{FO}(\mathsf{IFP})+\mathsf{C}$-fórmula

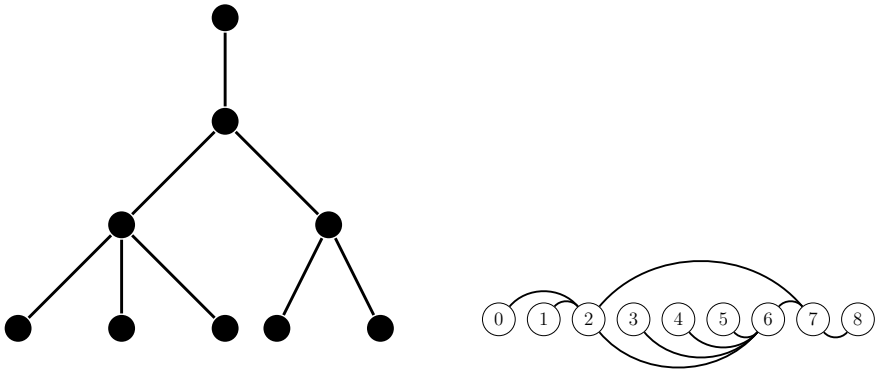
$$\varphi_R(t,i,j) := \mathsf{IFP}\,[Rxij \leftarrow \varphi(R,x,i,j)]\,(t,i,j), \tag{3.17}$$

define la relación que estábamos buscando. Entonces, Θ es la transducción dada por:

$$\theta_p := \top,$$
$$\theta_{uni}(i) := \neg max(i),$$
$$\theta_E(i,j) := \exists t_0((\forall t \neg Ett_0) \wedge \varphi_R(t_0,i,j)),$$
$$\theta_{\leq}(i,j) := i \leqslant j.$$

Corolario 3.4.11. $\mathsf{FO}(\mathsf{IFP})+\mathsf{C}$ *captura* PTIME *sobre la clase de árboles dirigidos.*

Si quisieramos extender el resultado anterior a árboles (donde la relación de ayacencia es simétrica), basta componer Θ con la transducción definida en el Ejemplo 3.3.6.



(a) $\mathcal{T}$ (b) $\Theta(\mathcal{T})$

Figura 3.5: Canonización de un árbol

3.4.3. Levantamiento de canonizaciones

El siguiente lema es un caso muy particular de un recurso que permite extender las clases canonizables. La idea es que, si se puede descomponer a una estructura de manera que cada componente sea canonizable, entonces toda la estructura resulta canonizable.

Lema 3.4.12. *Sea Θ una transducción normal que canoniza una clase de grafos $\mathcal{K}$. Sea $\mathcal{K}'$ otra clase de grafos con la propiedad de que todo $\mathcal{G} \in \mathcal{K}'$ es unión disjunta de grafos $\mathcal{G}_i \in \mathcal{K}$ (que llamamos componentes de $\mathcal{G}$) y que existe una fórmula $\chi(x, y)$ que expresa que x e y están la misma componente. Entonces existe una transducción Γ que canoniza a $\mathcal{K}'$.*

Demostración. Consideremos la transducción $\mathcal{X}$, con un parámetro x, tal que $\mathcal{X}(\mathcal{G}, p) = \mathcal{G}_i$ con $p \in \mathcal{G}_i$ (ver Ejemplo 3.3.5). Sea $\Lambda := \Theta \circ \mathcal{X}$, entonces $\Lambda(\mathcal{G}, p)$ es una copia ordenada de la componente de p. Llamemos $\lambda_E(x, i, j)$ a la fórmula que define la relación de adyacencia en Λ. La idea es similar a la de la canonización de árboles. Ordenamos lexicográficamente las copias ordenadas las componentes y las copiamos, en orden, sobre el segmento $[0, |G|-1]$. Definimos algunas fórmulas auxiliares, similares a las ya definidas. La siguiente fórmula cuenta el tamaño de la componente de x.

$$ord(x, i) := \#y\chi(x, y) = i. \tag{3.18}$$

El (pre)orden lexicografico queda definido por $\varphi_{\preceq}$, similar a la del caso anterior.

$$\begin{aligned}\varphi_{\preceq}(x, y) := (\exists ij(i < j \wedge ord(x, i) \wedge ord(y, j)))\vee \\ (\exists ij((i = j) \wedge ord(x, i) \wedge ord(y, j))\wedge \\ (\exists ij(\neg\lambda_E(x, i, j) \wedge \lambda_E(y, i, j) \wedge (\forall i'j'(i'j' <_{lex} ij)) \\ \rightarrow (\lambda_E(x, i', j') \leftrightarrow \lambda_E(y, i', j'))))).\end{aligned}$$

Las fórmulas $\varphi_{\prec}$ y $\varphi_{\approx}$ tienen el mismo significado que en el caso anterior. Nuevamente necesitamos determinar si el lugar k del segmento se corresponde con una componente isomorfa a la de x.

$$\begin{aligned}\varphi_C(x, k) := \exists ijl((\#y\varphi_{\prec}(y, x) = i)\wedge \\ (\tfrac{\#}{\chi}y\varphi_{\approx}(x, y) = j)\wedge \\ ord(x, l) \wedge (\exists j'(j' \leqslant j) \wedge (i + l \times j' = k))).\end{aligned}$$

Luego,

$$\begin{aligned}\gamma_E(i, j) := \exists xk(\varphi_C(x, k) \wedge (\exists i'j'(\lambda_E(x, i', j') \wedge (i' + k = i)\wedge \\ (j' + k = j)))).\end{aligned}$$

El resto de las fórmulas que definen la transducción son las usuales en una transducción normal. □

Llamamos *bosque* a una unión disjunta de árboles.

Corolario 3.4.13. $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$ *captura* PTIME *sobre la clase de bosques.*

Claro que clases más complejas no admiten descomposiciones en componentes que sean disjuntas, de manera que pueda definirse cada una mediante una fórmula. La intersección entre componentes representa un problema a la hora de definir una canonización, ya que impide hacerlo en cada una por separado. Para poder extender este resultado necesitamos otro tipo de descomposiciones. Un ejemplo clásico son las conocidas como *tree-decompositions*, introducidas por Halin [19] y utilizadas muy frecuentemente. El problema con éstas, es que no son invariantes por automorfismos, por lo tanto no son definibles mediante una lógica y no sirven para definir canonizaciones. Variantes de éstas, como *block-tree decompositons*, introducidas por Grohe [13], se adaptan mejor a este fin. Mediante éstas se puede dar una canonización para la clase de grafos planos, descomponiendo estos, sucesivamente, en sus componentes 1-, 2- y 3-conexas y definiendo un orden sobre las últimas, probando así, que $\mathsf{FO}(\mathsf{IFP}) + \mathsf{C}$ captura PTIME sobre la clase de grafos planos. Otra variante de descomposición son las *tree-like decompositions* [14], mediante las que se puede extender el resultado a cualquier clase que admita menores excluidos. Un ejemplo de este tipo de clases es el de los grafos embebibles en una superficie. Este resultado se debe a Grohe [15], quien además realiza un profundo estudio de las descomposiciones.

Índice alfabético

$<_{lex}$, 6
$\mathcal{A} \equiv \mathcal{B}$, 10
$\mathcal{A} \equiv^k \mathcal{B}$, 23
$\mathcal{A} \equiv^{\mathcal{L}} \mathcal{B}$, 10
$\mathcal{A} \equiv_m \mathcal{B}$, 12
$Num(A)$, 26
$V(A)$, 26
$\mathcal{F}_\infty$, 15
FO, 8
$\mathcal{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$, 12
$\mathcal{G}_m^k(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$, 22
$\mathcal{L}$-definible, 39
$\mathcal{L}_{\infty\omega}$, 19
$\mathcal{L}_{\infty\omega}^{\omega}$, 20
$\mathcal{L}_{\infty\omega}^{k}$, 20
SO, 23
$\bot$, 9
FO(IFP) + C, 25
LOGSPACE, 31
$\mathcal{K}_\leq$, 33
$\models_{\mathcal{L}}$, 39
NLOGSPACE, 31
NPTIME, 31
PSPACE, 31
PTIME, 31
Struc[σ], 5
$\top$, 9
φ_{ord}, 10
k-tipo, 44
k-tipo básico, 44
m-equivalentes, 12

bosque, 65
canonización, 58
ciclo, 5, 14
configuración, 30

ejemplos
- k-conexión, 56
- árbol dirigido, 18, 51
- artmética, 18
- canonización de grafos completos, 60
- canonización trivial, 58
- ciclos, 42
- circuito euleriano, 27
- clausura transitiva, 16
- clausura transitiva en $\mathcal{L}_{\infty\omega}^3$, 20
- componentes conexas, 51
- conexión, 14
- contar clases de equivalencia, 27
- contar en base n, 27
- estructura 2-rígida, 47
- fijar un orden con Σ_1^1, 25
- grafo lineal, 50
- grafos, 10
- grafos acíclicos, 18
- grafos completos, 43
- grafos conexos en FO(IFP), 17
- líneas, 42
- lógica trivial, 40
- orden, 10
- paridad, 13
- paridad con FO(PFP), 23
- paridad en FO(IFP) + C, 27
- paridad en Σ_1^1, 24

paridad en estructuras ordenadas, 13
paseo hamiltoniano, 24
poder expresivo de $\mathcal{L}^2_{\infty\omega}$, 21
poder expresivo de $\mathcal{L}_{\infty\omega}$, 20
sucesor, 10
supresión de k vértices, 50
equivalencia elemental, 10
estructura, 4
k-rígida, 47
rígida, 44

fórmula
atómica, 8
free, 8

grafo, 5
k-conexo, 56
completo, 43
lineal, 50

isomorfismo, 4
parcial, 12

juegos de Ehrenfeucht-Fraissé, 11

lógica abstracta, 39

máquina de Turing, 29
determinística, 30
no determinística, 30

operador inflacionario, 15
orden, 6
parcial, 6
total, 6

paseo, 5
preorden, 6
propiedad definible, 9
punto fijo, 15

qr, 11

rango cuantificacional, 11
relación compatible, 49

sentencia, 9

transducción, 49
directa, 50
dual, 56
fiel, 52
numérica, 51

valuación, 9
variable, 8
individual, 8
libre, 8
ligada, 8

Bibliografía

[1] Abiteboul, S. y V. Vianu: *Fixpoint Extensions of First-order Logic and Datalog-like Languages*. En *Proceedings of the Fourth Annual Symposium on Logic in Computer Science*, páginas 71–79. IEEE Press, 1989.

[2] Arora, S. y B. Barak: *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[3] Bondy, A. y U. S. R. Murty: *Graph Theory*. Graduate Texts in Mathematics. Springer, 2008.

[4] Cai, J. Y., M. Furer y N. Immerman: *An Optimal Lower Bound on the Number of Variables for Graph Identification*. En *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS '89, páginas 612–617, Washington, DC, USA, 1989. IEEE Computer Society.

[5] Chandra, A. y D. Harel: *Structure and Complexity of Relational Queries*. Journal of Computer and System Sciences, 25:99–128, 1982.

[6] Dawar, A.: *Feasible Computation through Model Theory*. Tesis de Doctorado, University of Pennsylvania, Philadelphia, 1993.

[7] Dawar, A., S. Lindell y S. Weinstein: *Infinitary Logic and Inductive Definability over Finite Structures*. Inf. Comput., 119(2):160–175, June 1995.

[8] Ebbinghaus, H. D.: *Extended Logics: The General Framework*. En Barwise, J. y S. Feferman (editores): *Model-Theoretic Logics*, páginas 25–76. Springer, New York, 1985.

[9] Ebbinghaus, H. D. y J. Flum: *Finite model theory*. Springer Monographs in Mathematics Series. Springer-Verlag Berlin and Heidelberg GmbH & Company, 2005.

[10] Erehnfeucht, A.: *An Application of Games to the Completeness Problem for Formalized Theories.* Fund. Math., (49):129–141, 1961.

[11] Fagin, R.: *Generalized first-order spectra and polynomial-time recognizable sets.* En Karp, R. (editor): *Complexity of Computation, SIAM-AMS Proceedings*, volumen 7, páginas 27–41, 1974.

[12] Fraïssé, R.: *Sur une nouvelle classification des systemes de relations.* CRAS, 230(1-2):1022–1024, 1950.

[13] Grohe, M.: *Fixed-Point Logics on Planar Graphs.* En *13th IEEE symposium on logic in computer science*, páginas 6–15. IEEE Computer Society, 1998.

[14] Grohe, M.: *Fixed-point definability and polynomial time on graphs with excluded minors.* J. ACM, 59(5):27:1–27:64, 2012.

[15] Grohe, M.: *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*, 2013. `http://www2.informatik.hu-berlin.de/~grohe/cap`.

[16] Gurevich, Y.: *Logic and the Challenge of Computer Science.* Current thrend in theoretical computer science, páginas 1–57, 1988.

[17] Gurevich, Y. y S. Shelah: *Fixed-point extensions of first-order logic.* Informe técnico CRL-TR-05-85, University of Michigan (Ann Arbor, MI US), 1985.

[18] Gurevich, Y. y S. Shelah: *On Finite Rigid Structures.* Journal of Symbolic Logic, 61:61–549, 1996.

[19] Halin, R.: *S-functions for graphs.* Journal of Geometry, 8(1-2):171–186, 1976.

[20] Immerman, N.: *Relational Queries Computable in Polynomial Time.* Information and Control, 68:86–104, 1986.

[21] Immerman, N.: *Descriptive complexity.* Graduate texts in computer science. Springer, 1999.

[22] Libkin, L.: *Elements of Finite Model Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[23] Marker, D.: *Model Theory : An Introduction.* Graduate Texts in Mathematics. Springer, 2002.

[24] Otto, M.: *Bounded variable logics and counting – A study in finite models*, volumen 9. Springer-Verlag, 1997.

[25] Papadimitriou, C.: *Computational complexity.* Addison-Wesley, Reading, Massachusetts, 1994.

[26] Seth, A.: *When do Fixed Point Logics Capture Complexity Classes?* En Kozen, Dexter (editor): *Proceedings of the Tenth Annual IEEE Symp. on Logic in Computer Science, LICS 1995*, páginas 353–365. IEEE Computer Society Press, June 1995.

[27] Vardi, M. Y.: *The Complexity of Relational Query Languages (Extended Abstract).* En *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, páginas 137–146, New York, NY, USA, 1982. ACM.