MATEMATICA

Acta Científica Venezolana, 53: 94–118, 2002

UNA PRESENTACION UNIFORME DE LA TEORIA DESCRIPTIVA DE LA COMPLEJIDAD COMPUTACIONAL

Argimiro Arratia
Departamento de Matemáticas
Universidad Simón Bolívar
Caracas, Venezuela
arratia@ma.usb.ve

Recibido: 11/12/01; Revisado: 07/05/02; Aceptado: 11/06/02

RESUMEN: La Teoría Descriptiva de la Complejidad Computacional estudia la Complejidad Computacional desde la perspectiva de la Lógica. Entre sus metas principales está el caracterizar mediante lógicas las clases de complejidad computacionales, tradicionalmente definidas en términos de máquinas de Turing con recursos acotados. La presentación que encontramos en la literatura actual de esta teoría se hace conforme a su desarrollo histórico, por lo que, para cada clase de complejidad computacional, se realiza una traducción particular del modelo computacional asociado a la clase en los elementos sintácticos que conforman la lógica con la que se pretende describir dicha clase. Esta larga y ardua labor intelectual se puede simplificar mediante un único esquema para vincular una clase de complejidad, espacial o temporal, con algún lenguaje formal, el cual involucra la única traducción de un modelo de máquina de Turing particular (específicamente, la máquina determinística con espacio de trabajo logarítmicamente acotado) en fórmulas de una lógica particular (la extensión de la lógica de primer orden con el operador de la clausura transitiva determinística). Esta versión uniforme de la Teoría Descriptiva de la Complejidad Computacional es la que presentamos en este trabajo. **Palabras claves:** Lógica, Teoría de Modelos Finitos, Complejidad Computacional.

A UNIFORM PRESENTATION OF THE THEORY OF DESCRIPTIVE COMPUTATIONAL COMPLEXITY

ABSTRACT: The Theory of Descriptive Computational Complexity deals with Computational Complexity from the perspective of Logic. Among its main goals it is the logical characterization of computational complexity classes, traditionally defined in terms of resource bounded Turing machines. The presentation often found of this theory in the current literature, follows its historical development; hence, in consequence, for each particular computational complexity class one finds a particular translation of the associated computational model into the syntactic elements belonging to the logic intended for describing the complexity class. This long and ardous intelectual job can be simplified with a scheme that links a time or space complexity class with a formal language, which requires learning just a single translation of a particular Turing machine model (namely, the logarithmic space bounded deterministic Turing machine) into formulas of a particular logic (the extension of first order logic with the deterministic transitive closure operator). It is this uniform version of the Theory of Descriptive Computational Complexity which we present in this paper. **Keywords:** Logic, Finite Model Theory, Computational Complexity.

1. INTRODUCCION

La teoría descriptiva de la complejidad computacional, o en forma más concisa, la Teoría Descriptiva de la Complejidad, estudia la Complejidad Computacional desde la perspectiva de la Lógica. Entre sus metas principales está el caracterizar mediante lógicas las clases de complejidad computacionales, tradicionalmente definidas en términos de máquinas de Turing con recursos acotados. De esta manera, el tiempo, el espacio u otra medida dependiente de la arquitectura de una máquina, necesarios para resolver un problema, se traducen en número de cuantificadores, variables y otros recursos sintácticos que permiten describir el problema en un lenguaje formal.

Además de librarnos así de toda referencia a un modelo de computadora particular, este marco teórico provisto por la Lógica nos permite importar los métodos y herramientas de ésta, para trabajar en las soluciones a los enigmas de la Complejidad Computacional. Es en ese sentido que apreciamos la Teoría Descriptiva de la Complejidad: no como sustituto, sino como complemento de la caja de herramientas que ya teníamos para trabajar en la complejidad computacional de problemas. Por esto último, la presentación que hago aquí de esta teoría no va de acuerdo a como se sucedió históricamente, ni tampoco soy fiel a las demostraciones originales de sus resultados principales. Estas desmostraciones las he sustituido por otras, acordes con un esquema único para vincular clases de complejidad, espacial o temporal, con lenguajes formales, el cual nos da una visión más uniforme de la teoría y que explicaré en la sección 3.4.

Un elemento necesario en este esquema para establecer puentes entre la Complejidad Computacional y la Lógica, es el de reducción descriptible por fórmulas de la

lógica de primer orden o PO–reducciones. El concepto de reducibilidad de primer orden y sus variantes se consideran de mucha importancia en la evolución de la complejidad en términos descriptivos y, por ello, en la sección 4, se desarrolla nuestro segundo paradigma, que complementa el primero mencionado antes. Este último concierne una manera de demostrar que un problema es un representante de la clase de problemas de igual complejidad PO-reducibles, partiendo de que nuestro problema es un representante para la clase determinada por reducciones dadas en términos de máquinas de Turing (e.g. log–reducción).

Una vez establecidos estos esquemas los pondremos en práctica reproduciendo, en la sección 5, algunos de los resultados que forjaron la Teoría Descriptiva de la Complejidad como, por ejemplo, la relación biunívoca entre la clase **NP** y el fragmento existencial de la lógica de segundo orden (resultado que es considerado la piedra fundacional de esta teoría), o vínculo similar entre **P** y **PESPACIO** y las extensiones de la lógica de primer orden por operadores inductivos.

En un intento por hacer esta presentación autocontenida, expondré en las secciones 2 y 3 los elementos básicos de la Complejidad Computacional y de la Lógica Matemática que son necesarios. Por razones de espacio esto lo hago de manera informal y bastante resumida, recomendando al lector interesado en los detalles leer[11,17] para lo referente a la Complejidad Computacional y [6] para una buena introducción a la Lógica Matemática.

2. Complejidad Computacional

La Complejidad Computacional es el área de las ciencias de la computación que intenta explicar por qué algunos problemas, algorítmicamente resolubles, requieren para este tipo de solución de grandes cantidades de recursos físicos, como son el tiempo y el espacio necesarios para realizar operaciones mediante un computador.

Para comprender y clasificar la complejidad de algoritmos resulta conveniente considerar sólo problemas de decisión: dado un conjunto K de objetos con cierta propiedad, nos interesa determinar membresía en K. Una vez resuelto este problema algorítmicamente nos interesa saber cuán "eficiente" es nuestra solución.

Comenzaremos por definir lo que entendemos por un "problema" y fijaremos nuestro modelo de algoritmo.

2.1. Estructuras Finitas y Máquinas de Turing

Un vocabulario finito $\tau = \{R_1, \ldots, R_r, C_1, \ldots, C_s\}$ es un conjunto de símbolos relacionales R_i y símbolos constantes C_j. Cada símbolo relacional tiene asociado un número entero positivo que indica su *aridad*: la longitud de los vectores de elementos que pertenecen a la relación. Una estructura finita sobre τ (o τ-estructura finita) es una tupla

$$\mathcal{A} = \langle A, R_1^{\mathcal{A}}, \ldots, R_r^{\mathcal{A}}, C_1^{\mathcal{A}}, \ldots, C_s^{\mathcal{A}} \rangle$$

que consiste de:

un universo $A = \{a_1, \ldots, a_n\}$;

constantes $C_j^{\mathcal{A}} \in A$, interpretaciones de cada símbolo constante $C_j \in \tau$;

relaciones $R_i^{\mathcal{A}} \subseteq A^{k_i}$, interpretaciones de cada símbolo relacional $R_i \in \tau$ de aridad k_i.

Denotaremos por $|A|$ la cardinalidad del universo de $\mathcal{A}$. Otra notación que importaremos de la tradición es escribir $R(a_1, \ldots, a_k)$ para indicar que $(a_1, \ldots, a_k) \in R$, para cualquier símbolo relacional R de aridad k. Frecuentemente omitiremos, por comodidad, el superíndice $\mathcal{A}$ en $R^{\mathcal{A}}$ y $C^{\mathcal{A}}$, y utilizamos el mismo símbolo R o C para referirnos a una relación o constante de un vocabulario y, a la vez, su interpretación en una estructura. Denotaremos por EST(τ) el conjunto de todas las estructuras finitas sobre τ.

Algunas veces necesitaremos hablar de subestructuras de una estructura dada. Una τ-estructura $\mathcal{B}$ es una *subestructura* de una τ- estructura $\mathcal{A}$, si el universo de $\mathcal{B}$ está contenido en el universo de $\mathcal{A}$, cualquier relación en $\mathcal{B}$ es la correspondiente relación en $\mathcal{A}$ restringida a B y cualquier símbolo constante tiene igual interpretación en $\mathcal{A}$ y $\mathcal{B}$.

Definición 1 Dado un vocabulario τ, un *problema* sobre τ es un subconjunto K de EST(τ), cerrado bajo isomorfismos. Una *instancia* de K es una τ-estructura y una *instancia positiva* de K es una instancia de K que pertenece a K. □

La abstracción teórica más popular de un algoritmo y, en general, de una computadora, es la *máquina de Turing*. Existen diversas variaciones de este modelo (por supuesto que sólo en papel) que se explican en forma detallada y extensa en muchos libros (e.g.,[11,17]) y, dada la gran cantidad de literatura existente sobre el tema, nos ahorraremos el espacio para definir máquinas de Turing aquí y nos limitaremos sólo a puntualizar aquellas de sus características que consideramos importante recordar. La versión de máquina de Turing que adoptaremos es la que tiene múltiples cintas con sus correspondientes cabezales: una cinta para sólo leer los datos de entrada, varias otras para realizar el trabajo necesario (por lo que los cabezales de estas cintas de trabajo leen y escriben) y, posiblemente, una cinta de sólo salida (y su cabezal sólo escribe).

De los elementos formales que definen una de estas máquinas de Turing M nos interesa recordar:

- que está constituida por un conjunto finito de estados de los cuales distinguimos algunos como estados de aceptación;

- que cada paso de la computación que realiza M es finitamente descriptible por una *configuración instantánea* CI: una tupla con la información del estado q en que se encuentra M en un instante de sus operaciones; la posición i del cabezal lector en la cinta de entrada; para cada $j = 1, \ldots, k$, la palabra u_j que posee escrita en su j–ésima cinta de trabajo (asumimos M tiene $k \geq 1$ cintas de trabajo) junto con la posición h_j del cabezal de la j–ésima cinta de trabajo. Simbólicamente, escribimos

$$CI := \langle q, i, u_1, h_1, u_2, h_2, \ldots, u_k, h_k \rangle$$

  Observe que no se incluyen en CI los contenidos de las cintas de entrada y salida.

- M *acepta* una secuencia finita de símbolos si al escribirse estos en su cinta de entrada se puede producir una sucesión finita de configuraciones, CI_0, CI_1, ..., CI_r, donde CI_0 es la configuración inicial, CI_r es una configuración final que contiene un estado q_a de aceptación y, para $i = 0, \ldots, r-1$, CI_{i+1} se obtuvo a partir de CI_i aplicando una de las reglas que rigen los movimientos de M. Una tal sucesión $CI_0, CI_1, \ldots, CI_r$ la denominaremos *una computación de* M.

- El tiempo de una computación $CI_0, CI_1, \ldots, CI_r$ es r. El espacio utilizado durante la computación $CI_0, CI_1, \ldots, CI_r$ es la longitud máxima de las configuraciones CI_i que la constituyen (donde la longitud de CI es $\sum_{i=1}^{k} |u_i|$).

- Sea $f : \mathbb{N} \to \mathbb{N}$ una función, donde $\mathbb{N}$ es el conjunto de los naturales. La máquina de Turing M opera en tiempo (respectivamente, espacio) $f(n)$ si para entradas de longitud n, el tiempo (resp. espacio) de cualquier computación de M es $O(f(n))$.

- La máquina de Turing M es:

  determinística, si para cualquier configuración CI_i existe a lo sumo una configuración CI_{i+1} siguiente;

  nodeterminística, si para cualquier configuración CI_i existe ninguna, una o más configuraciones CI_{i+1} siguiente.

Dado que una máquina de Turing sólo trabaja con secuencias finitas de símbolos y deseamos que procese estructuras finitas, debemos codificar nuestros problemas (i.e., clases de estructuras) como un conjunto de palabras de longitud finita. La codificación usual es como un subconjunto de $\{0,1\}^*$, el conjunto de palabras sobre el alfabeto $\{0,1\}$, y a continuación presentamos una manera de hacer esto. Téngase en cuenta que tal codificación presupone que las estructuras son *ordenadas*, es decir, que su universo está linealmente ordenado (esto nos permitirá, por ejemplo, hablar del "k–ésimo elemento").

Definición 2 Sea $\tau := \{R_1, \ldots, R_r, C_1, \ldots, C_s\}$ un vocabulario donde R_i es una relación de aridad k_i, para cada $i = 1, \ldots, r$, y cada C_j es un símbolo constante. Sea $\mathcal{A} = \langle \{0, \ldots, n-1\}, R_1^{\mathcal{A}}, \ldots, R_r^{\mathcal{A}}, C_1^{\mathcal{A}}, \ldots, C_s^{\mathcal{A}} \rangle$ una τ–estructura *ordenada* de tamaño n. La *codificación de* $\mathcal{A}$ sobre el alfabeto $\{0,1\}$, denotada $cod_\tau(\mathcal{A})$, es la palabra

$$0^{|A|} u_1 u_2 \cdots u_s w_1 \cdots w_r$$

en $\{0,1\}^*$, donde, para cada $i = 1, \ldots, s$, u_i es la representación en binario de $C_i^{\mathcal{A}}$ y, para cada $j = 1, \ldots, r$, w_j es la palabra en $\{0,1\}^*$ de longitud n^{k_j} tal que el k-ésimo dígito de w_j es 1 si el k-ésimo elemento de A^{k_j} (en el orden lexicográfico de k_j–tuplas), está en $R_j^{\mathcal{A}}$, o 0 en caso contrario. (Observe que si τ es vacío entonces $cod_\tau(\mathcal{A}) = 0^{|A|}$, por lo que siempre podemos recuperar la cardinalidad de $\mathcal{A}$ a partir de su codificación.)

Si K es un problema sobre τ, entonces $cod_\tau(K) := \{cod_\tau(\mathcal{A}) : \mathcal{A} \in K\}$ es la codificación de K como un lenguaje en $\{0,1\}^*$. □

Tenemos así que cuando queramos dar una τ–estructura $\mathcal{A}$ como dato de entrada para una máquina de Turing M, esto es sólo posible si los elementos de $\mathcal{A}$ están linealmente ordenados, en cuyo caso codificamos $\mathcal{A}$ como palabra en $\{0,1\}^*$ de acuerdo con el esquema de la Definición 2 y escribimos $cod_\tau(\mathcal{A})$ en la cinta de entrada de M. En el futuro, cuando escribamos expresiones como "máquina de Turing M recibe por entrada estructura $\mathcal{A}$", entiéndase esto como una abreviación de "$\mathcal{A}$ es ordenada e introducimos $cod_\tau(\mathcal{A})$ en M". Téngase muy en cuenta que el proceso de obtener $cod_\tau(\mathcal{A})$ a partir de $\mathcal{A}$ es muy eficiente (esto debe quedar claro después de la Sección 3)* y, por lo tanto, podemos ignorarlo en la contabilización del tiempo o el espacio en el que incurra M para decidir si $cod_\tau(\mathcal{A})$ es la codificación de una instancia positiva de un problema o no. Esto también nos permite permanecer tranquilos con el nivel de informalidad general que deseamos, donde, por ejemplo, $\mathcal{A}$ y $cod_\tau(\mathcal{A})$ se considerarán como esencialmente lo mismo.

2.2. Clases de complejidad

L (respectivamente **NL**; respectivamente **PESPACIO**) es la clase de problemas computacionales resolubles por máquinas de Turing determinísticas (resp. nodeterminísticas; resp. determinísticas) que operan en espacio $\log(n)$ (resp. $\log(n)$; resp. polinomial en n), donde n es la longitud de los datos de entrada. **P** (resp. **NP**) es la clase de problemas computacionales resolubles por máquinas de Turing determinísticas (resp. nodeterminísticas) que operan en tiempo polinomial en la longitud de los datos de entrada. Si K es un problema sobre τ, su complemento se define como $\overline{K} := \mathrm{EST}(\tau) - K$. Si **C** es una clase de complejidad, su complemento se define como $\mathbf{co}\mathbf{C} := \{\overline{K} : K \in \mathbf{C}\}$.

*Nos referimos a que la codificación es descriptible en PO.

Puesto que nodeterminismo es una generalización de determinismo, resulta inmediato que $\mathbf{L} \subseteq \mathbf{NL}$ y $\mathbf{P} \subseteq \mathbf{NP}$. También es inmediato que si **C** es cualquiera de las clases determinísticas, entonces **C** = **coC**. Algo menos trivial resulta demostrar que $\mathbf{NL} \subseteq \mathbf{P}$ y $\mathbf{NP} \subseteq \mathbf{PESPACIO}$, y con estas se tiene la siguiente cadena de contenciones

L $\subseteq$ **NL** $\subseteq$ **P** $\subseteq$ **NP** $\subseteq$ **PESPACIO**

Demostrar que alguna de estas cuatro inclusiones es estricta son de los problemas, aún abiertos, más importantes de la Complejidad Computacional. El más notorio es contestar la pregunta de si **P** es igual o no a **NP**.[**] Ahora bien, resulta que *alguna* de esas cuatro contenciones *debe* ser propia, ya que se ha logrado demostrar que **L** $\neq$ **PESPACIO** y $\mathbf{NL} \neq \mathbf{PESPACIO}$.

Otro problema abierto de igual importancia y que ha suscitado similar atención, es si **NP** es igual o no a su complemento. Observe que si se demostrase que **NP** $\neq$ **coNP**, esto implicaría que **P** $\neq$ **NP**.

A continuación presentamos una lista de problemas que resultarán de gran utilidad más adelante.

Ejemplo 3 Sea $\tau_2 := \{E, C, D\}$, donde E es un símbolo de relación binario, C y D son dos símbolos constantes. Una τ_2–estructura $\mathcal{A}$ puede visualizarse como un grafo más dos vértices específicos, es decir, $\mathcal{A} = \langle A, E^{\mathcal{A}}, s, t\rangle$ donde $E^{\mathcal{A}}$ es un conjunto de arcos, s y t dos vértices que son las interpretaciones en $\mathcal{A}$ de las constantes C y D. Considere también el vocabulario $\tau_3 := \{R, C, D\}$, donde R es un símbolo de relación ternario, C y D como antes. Una τ_3–estructura $\mathcal{A} = \langle A, R^{\mathcal{A}}, s, t\rangle$ puede visualizarse como un *sistema de caminos*; esto es, un conjunto de vértices A, una relación $R^{\mathcal{A}} \subseteq A \times A \times A$, dos vértices específicos $s, t \in A$, y donde un vértice es *accesible* si es s, o si dados x e y vértices accesibles y si se tiene $(x, y, z) \in R$ (siendo posible $x = y$), entonces z es accesible.[***] Considere los siguientes problemas:

DTC := $\{\mathcal{A} = \langle A, E^{\mathcal{A}}, s, t\rangle \in \text{EST}(\tau_2) : \mathcal{A}$ es un grafo dirigido y existe un camino de s a t de vértices con grado exterior = 1$\}$

TC := $\{\mathcal{A} = \langle A, E^{\mathcal{A}}, s, t\rangle \in \text{EST}(\tau_2) : \mathcal{A}$ es un grafo dirigido y existe un camino de s a $t\}$

PS := $\{\mathcal{A} = \langle A, R^{\mathcal{A}}, s, t\rangle \in \text{EST}(\tau_3) : \mathcal{A}$ es un sistema de caminos donde t es accesible desde $s\}$

HP := $\{\mathcal{A} = \langle A, E, s, t\rangle \in \text{EST}(\tau_2) : \mathcal{A}$ es un grafo dirigido y existe un camino hamiltoniano de s a $t\}$

WHEX := $\{\mathcal{A} = \langle A, E, s, t\rangle \in \text{EST}(\tau_2) : \mathcal{A}$ es un grafo y el Jugador 1 tiene una estrategia ganadora en el juego de Whex sobre $\mathcal{A}\}$

El grado exterior de un vértice es el número de arcos que salen de él. Un camino hamiltoniano es un camino que pasa por todos los vértices del grafo sin repeticiones. El juego de Whex sobre $\mathcal{A} = \langle A, E, s, t\rangle$ consiste en dos jugadores (Jugador 1 y Jugador 2) quienes se alternan en colorear los vértices del grafo $\mathcal{A}$ salvo s y t. Jugador 1 utiliza el color azul, mientras que Jugador 2 utiliza el color rojo. Las jugadas están sujetas a las siguientes restricciones:

> Jugador 1 es el primero en jugar y debe hacerlo coloreando azul un vértice adyacente a s. En sus sucesivas jugadas Jugador 1 debe colorear azul un vértice que no haya sido coloreado antes, ni azul ni rojo, adyacente al último vértice coloreado azul. Jugador 2 debe responder coloreando rojo un vértice que no haya sido coloreado antes, ni azul ni rojo, adyacente al último vértice coloreado azul (i.e., la jugada previa de Jugador 1).

Es decir, Jugador 1 intenta construir un camino, paso a paso, partiendo desde s y Jugador 2 intenta bloquear este camino a cada paso. El juego finaliza cuando todos los vértices (excepto s y t) han sido coloreados. Jugador 1 gana si, y sólo si, existe un camino de s a t en $\mathcal{A}$ de vértices azules. □

No es difícil diseñar un algoritmo nodeterminístico y que emplee espacio logarítmico para decidir membresía en TC (ejercicio). Por lo tanto, TC $\in \mathbf{NL}$, y esto es lo mejor que se puede decir actualmente. De igual manera, la mejor cota inferior computacional que se conoce para cada uno de los otros problemas mencionados es, respectivamente, así:

DTC $\in$ **L**, PS $\in$ **P**, HP $\in \mathbf{NP}$ y WHEX $\in$ **PESPACIO**.

Un concepto importante en la Complejidad Computacional es el de reducción de un problema a otro. Este sirve para comparar la complejidad de un problema respecto de otro.

[**] El 24 de Mayo del año 2000 este problema fue presentado por Michael Atiyah y John Tate, como uno de los siete problemas matemáticos que conforman los "Problemas del Milenio" y por cuyas soluciones el empresario estadounidense Landon Clay ofrece un millón de dólares por problema. Este anunció fue hecho en el Collège de France durante los actos organizados para celebrar el centenario de la lista de 23 problemas propuesta por Hilbert en 1900. La escogencia de estos problemas fue hecha por un panel de científicos conformado por Andrew Wiles, Alain Connes, Edward Witten y Arthur Jaffe (ver `http://www.claymath.org/index.htm`).

[***] Otra manera de visualizar una τ_3–estructura es como un sistema de deducción donde s es un axioma, t es un teorema y $R^{\mathcal{A}}$ es una regla que asocia a un par de teoremas (elementos "demostrables") una conclusión (que es entonces, a su vez, demostrable).

Definición 4 Sean τ y σ dos vocabularios y k un entero positivo. Una (τ,σ)–*traductora de aridad* k es una máquina de Turing M determinística con cinta de entrada, cinta de salida y varias cintas de trabajo, que acepta como entrada sólo codificaciones en $\{0,1\}^*$ de τ–estructuras ordenadas de cardinalidad n y, durante sus cómputos, escribe en su cinta de salida la codificación en $\{0,1\}^*$ de alguna σ–estructura ordenada de cardinalidad n^k, unívocamente determinada por los datos de entrada. Si $\mathcal{A} \in \text{EST}(\tau)$, denotamos por $M(cod_\tau(\mathcal{A}))$ la salida de M al recibir por entrada $cod_\tau(\mathcal{A})$ y decimos que M traduce la codificación en $\{0,1\}^*$ de la τ–estructura $\mathcal{A}$ en la palabra $M(cod_\tau(\mathcal{A}))$ en $\{0,1\}^*$, correspondiente a la codificación de alguna σ–estructura.

En pocas palabras, una (τ,σ)–traductora es una máquina que computa una función de $cod_\tau(\text{EST}(\tau))$ en $cod_\sigma(\text{EST}(\sigma))$. Si el espacio máximo (número de celdas) requerido en las cintas de trabajo para realizar la traducción es $O(\log n)$, donde n es la longitud de la entrada, hablamos entonces de una $\log$–(τ,σ)–traductora. (Similarmente, si el tiempo necesario para efectuar la traducción es polinomial en los datos de entrada, hablamos entonces de una poli–(τ,σ)–traductora. Sin embargo, éste tipo de traductores no nos interesarán mucho.) □

Definición 5 Un problema $K \subseteq \text{EST}(\tau)$ es $\log$–*reducible* a un problema $Q \subseteq \text{EST}(\sigma)$, y escribimos $K \leq_{log} Q$, si existe una $\log$–(τ,σ)–traductora M de aridad k tal que, para cualquier $\mathcal{A} \in \text{EST}(\tau)$,

$cod_\tau(\mathcal{A}) \in cod_\tau(K)$ si y sólo si $M(cod_\tau(\mathcal{A})) \in cod_\sigma(Q)$. □

Intuitivamente, si $K \leq_{\log} Q$ entonces Q es tanto o más difícil de decidir computacionalmente que K. La relación $\leq_{log}$ es reflexiva y transitiva.

Definición 6 Sea **C** una clase de complejidad. Un problema $K \subseteq \text{EST}(\tau)$ es *completo* para **C** via log–reducciones, si $K \in \mathbf{C}$ y si para todo $Q \in \mathbf{C}$ se tiene $Q \leq_{\log} K$. Esto se abreviará como "K es **C**–log–completo" o, simplemente, "K es **C**–completo" cuando no haya duda del tipo de reducción en consideración (más adelante veremos otras). □

Debido a la transitividad de $\leq_{\log}$ una vez que se conoce un problema K como **C**–completo, entonces dado otro problema Q en **C** para ver que Q es a su vez **C**–completo es suficiente probar que $K \leq_{\log} Q$.

El que tal vez es el primer problema **NP**–completo fue dado por Stephen Cook en [4] y trata sobre satisfabilidad de fórmulas en la lógica proposicional. Daremos un breve repaso de esta lógica y definiremos el problema de Cook a continuación.

2.3. Lógica proposicional y satisfacción

$\mathbf{Var} = \{x_1, x_2, x_3, \ldots\}$ es un conjunto infinito numerable de variables. Estas variables se combinan con los *operadores booleanos* $\wedge$, $\vee$, $\neg$, $\longrightarrow$ y $\longleftrightarrow$, para obtener las *fórmulas proposicionales*.

Definición 7 Una fórmula proposicional (o simplemente una proposición) es

(i) una variable $x \in \mathbf{Var}$;

(ii) $\neg(\varphi)$ (negación), donde φ es una proposición;

(iii) $(\varphi_1 \wedge \varphi_2)$ (conjunción), $(\varphi_1 \vee \varphi_2)$ (disyunción), $(\varphi_1 \longrightarrow \varphi_2)$ (implicación), $(\varphi_1 \longleftrightarrow \varphi_2)$ (equivalencia), donde φ_1 y φ_2 son proposiciones.

(Usualmente se omiten los paréntesis para mejorar la legibilidad de las fórmulas, pero teniendo cuidado de no incurrir en ambiguedades.)

La interpretación de éstas fórmulas es como sigue. Una *asignación de verdad* es una función $\mathbf{T} : W \to \{0,1\}$ donde W es un subconjunto finito de **Var**. Dada una proposición φ y una asignación de verdad **T**, decimos que **T** *satisface* φ, denotado $\mathbf{T} \models \varphi$, si sucede:

(i) cuando $\varphi := x \in W$, $\mathbf{T} \models x \iff \mathbf{T}(x) = 1$;

(ii) cuando $\varphi := \neg\psi$, $\mathbf{T} \models \varphi \iff \mathbf{T} \not\models \psi$ (i.e., **T** no satisface ψ, o equivalentemente $\mathbf{T}(\psi) = 0$);

(iii) cuando $\varphi := \psi \wedge \theta$, $\mathbf{T} \models \varphi \iff \mathbf{T} \models \psi$ y $\mathbf{T} \models \theta$;

(iv) cuando $\varphi := \psi \vee \theta$, $\mathbf{T} \models \varphi \iff \mathbf{T} \models \psi$ o $\mathbf{T} \models \theta$;

(v) cuando $\varphi := \psi \longrightarrow \theta$, $\mathbf{T} \models \varphi \iff$ $\mathbf{T} \models \psi$ implica $\mathbf{T} \models \theta$;

(vi) cuando $\varphi := \psi \longleftrightarrow \theta$, $\mathbf{T} \models \varphi \iff$ $\mathbf{T} \models \psi$ si y sólo si $\mathbf{T} \models \theta$. □

Definición 8 Una proposición φ es *satisfacible* si existe una asignación de verdad $\mathbf{T}$ tal que $\mathbf{T} \models \varphi$. □

Por ejemplo, la proposición

$$\varphi \quad := \quad (\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (x_2 \vee \neg x_3) \quad (1)$$

es satisfacible. Basta considerar $\mathbf{T} : \{x_1, x_2, x_3\} \to \{0,1\}$ con $\mathbf{T}(x_1) = \mathbf{T}(x_3) = 0$ y $\mathbf{T}(x_2)$ cualquier valor.

Decimos que una proposición φ está en *forma normal conjuntiva* (fnc) si es una conjunción de disyunciones de *literales* (i.e., variables o sus negaciones); es decir φ tiene la forma $C_1 \wedge C_2 \wedge \ldots \wedge C_n$, donde cada $C_i := l_{i1} \vee \ldots \vee l_{im}$ con $l_{ij} = x_{ij}$ o $\neg x_{ij}$. Las C_i se llaman *cláusulas*. Observe que φ en (1) está en fnc. Existe también la *forma normal disyuntiva* (fnd) y es un hecho que toda fórmula proposicional es equivalente a una fórmula en fnc y también a una fórmula en fnd (e.g. ver[6]).

El problema de Cook sobre la lógica proposicional es el siguiente. Dada una proposición φ, en forma normal conjuntiva, ¿existe una asignación de verdad que hace φ cierta?, en otras palabras, ¿es φ satisfacible? Sea SAT el

conjunto de todas las proposiciones, en forma normal conjuntiva, que son satisfacibles. Un algoritmo determinístico para decidir membresía en SAT requiere (hasta donde se sabe) tiempo exponencial. Sin embargo con nodeterminismo se puede tener tiempo polinomial.

Sea $\tau_{sat} = \{P, N\}$ con P y N dos símbolos relacionales binarios. Una τ_{sat}–estructura $\mathcal{A} = \langle A, P^{\mathcal{A}}, N^{\mathcal{A}}\rangle$ con $|A| = n$, representa una proposición en forma normal conjuntiva de n cláusulas, cada una con a lo sumo $2n$ literales, de manera que

> $(i,j) \in P^{\mathcal{A}}$ si y sólo si variable j aparece positiva en cláusula i
> $(i,j) \in N^{\mathcal{A}}$ si y sólo si variable j aparece negativa en cláusula i.

Como ejemplo codifiquemos φ en (1) como una estructura finita. Esta es

$$\begin{aligned} \mathcal{A}_{\varphi} &= \langle\{1,2,3\}, P^{\mathcal{A}_{\varphi}}, N^{\mathcal{A}_{\varphi}}\rangle, \text{ con} \\ P^{\mathcal{A}_{\varphi}} &= \{(2,2),(2,3),(3,2)\} \text{ y} \\ N^{\mathcal{A}_{\varphi}} &= \{(1,1),(1,3),(2,1),(3,3)\}. \end{aligned}$$

SAT, como clase de τ_{sat}–estructuras, es el problema

> SAT $:= \{\mathcal{A} \in \text{EST}(\tau_{sat}) : \mathcal{A}$ es una proposición (fnd) satisfacible $\}$.

Teorema 9 (S. Cook[4]) SAT *es* **NP**–*completo.* □

Observe que, por ejemplo, si K es un problema en **NP** y $Q \leq_{\log} K$ entonces $Q \in$ **NP**; es decir, **NP** es *cerrado bajo* $\log$*–reducciones*. Esto es cierto también para todas las otras clases de complejidad que hemos definido. Por lo tanto, hallar ejemplos de problemas completos via $\leq_{\log}$, en cada una de las clases de complejidad, es un paso importante para demostrar que estas coinciden o difieren. Cada uno de los problemas en el Ejemplo 3 son completos en las clases donde se encuentran. Existen textos que presentan largas listas de problemas completos via $\leq_{\log}$; en particular, la obra de Garey y Johnson[9] es considerada una enciclopedia de problemas **NP**–completos.

3. Lógica y Teoría de Modelos Finitos.

Nuestro próximo objetivo es definir la lógica de primer orden, lo cual constituye un primer paso hacia una descripción puramente sintáctica de las clases de complejidad computacional.

3.1. La Lógica de Primer Orden (PO).

3.1.1. Sintáxis

Sea **Var** el conjunto infinito numerable de variables presentado antes. Sean $\neg$, $\vee$, $\wedge$, $\longrightarrow$, $\longleftrightarrow$ los operadores booleanos, $\exists$ y $\forall$ los cuantificadores, $=$ el símbolo de igualdad,) y (los paréntesis y τ un vocabulario. La *lógica de Primer Orden sobre* τ, PO(τ), está constituida por sucesiones finitas de símbolos tomados entre los descritos en el párrafo anterior, que se denominan *fórmulas* y que se componen utilizando las reglas que siguen a continuación, un número finito de veces.

(i) Si $R \in \tau$ es un símbolo relacional de aridad s, $t_1, \ldots, t_s, t$ y t' son variables o constantes $C \in \tau$, entonces
$$R(t_1, \ldots, t_s) \text{ y } (t = t')$$
son fórmulas. (Estas son las fórmulas *atómicas*. Las variables o constantes usualmente se denominan *términos*.)

(ii) Si ϕ y ψ son fórmulas entonces
$$(\phi \wedge \psi),\ (\phi \vee \psi),\ \neg(\phi),\ (\phi \longrightarrow \psi),\ (\phi \longleftrightarrow \psi)$$
también son fórmulas.

(iii) Si ϕ es una fórmula y $x \in \mathbf{Var}$, entonces
$$\exists x(\phi) \text{ y } \forall x(\phi)$$
son fórmulas.

(Al igual que en la lógica proposicional, omitimos paréntesis siempre que sea posible.)

Las variables que aparecen en una fórmula fuera del alcance de algún cuantificador se llaman libres. Una *sentencia* es una fórmula sin variables libres.

El conjunto de todas las fórmulas de primer orden sobre cualquier vocabulario finito τ será indicado por PO. En símbolos,

$$\text{PO} = \bigcup_{\tau}\{\text{PO}(\tau) : \tau \text{ es un vocabulario finito }\}.$$

3.1.2. Semántica

Sean τ un vocabulario, $\mathcal{A}$ una τ-estructura de cardinalidad n y A su universo. Sea ϕ una sentencia en PO(τ). A continuación se define la relación de satisfación de sentencias en estructuras, $\mathcal{A} \models \phi$, la cual extiende la semántica que dimos para la lógica proposicional.

(i) Si $\phi := R(C_1, \ldots, C_k)$, donde $R \in \tau$ es un símbolo relacional de aridad K y $C_1, \ldots, C_k$ son símbolos constantes en τ, entonces
$\mathcal{A} \models R(C_1, \ldots, C_k) \iff R^{\mathcal{A}}(C_1^{\mathcal{A}}, \ldots, C_k^{\mathcal{A}})$ es cierto en $\mathcal{A}$
Si $\phi := C = C'$, donde C y C' son símbolos constantes en τ, entonces
$\mathcal{A} \models C = C' \iff C^{\mathcal{A}} = C'^{\mathcal{A}}$ es cierto en $\mathcal{A}$

(ii) $\mathcal{A} \models \neg\phi \iff \mathcal{A} \not\models \phi$ (i.e., no es cierto que $\mathcal{A} \models \phi$)

(iii) $\mathcal{A} \models \phi \wedge \psi \iff \mathcal{A} \models \phi$ y $\mathcal{A} \models \psi$

(iv) $\mathcal{A} \models \exists x\phi \iff$ para algún $a \in A$, $\mathcal{A} \models \phi(\frac{x}{a})$, donde $\phi(\frac{x}{a})$ es la fórmula obtenida a partir de ϕ sustituyendo todas las apariciones de x por a. (Observe que implícitamente hemos extendido el vocabulario original con nuevas constantes, tantas como elementos en $\mathcal{A}$, y así tratamos cada elemento de $\mathcal{A}$ como símbolo constante cuando aparece en alguna fórmula.)

La semántica para sentencias que involucran los operadores $\vee$, $\longrightarrow$, $\longleftrightarrow$ y el cuantificador $\forall$ se determina usando las reglas (ii) a (iv) y considerando: $\phi \vee \psi$ como una abreviación de $\neg(\neg\phi \wedge \neg\psi)$; $\phi \longrightarrow \psi$ como una abreviación de $\neg\phi \vee \psi$; $\phi \longleftrightarrow \psi$ como una abreviación de $(\phi \longrightarrow \psi) \wedge (\psi \longrightarrow \phi)$; $\forall x\phi$ como una abreviación de $\neg\exists x\neg\phi$.

Para extender la semántica a cualquier fórmula $\phi(x_1, \ldots, x_n)$ (cuyas variables libres están en $\{x_1, \ldots, x_n\}$) decimos,

$$\mathcal{A} \models \phi(x_1, \ldots, x_n) \iff \mathcal{A} \models \forall x_1 \ldots \forall x_n \phi(x_1, \ldots, x_n)$$

Ejemplo 10 Sean $\tau = \{R(\cdot,\cdot)\}$ y $\phi \in$ PO(τ) la siguiente sentencia:

$$\begin{aligned}\phi \quad := \quad & \forall x(\neg R(x,x)) \wedge \forall x\forall y\forall z((R(x,y) \wedge R(y,z)) \\ & \longrightarrow R(x,z)) \wedge \forall x\forall y(x = y \vee R(x,y) \vee R(y,x)).\end{aligned}$$

Dada $\mathcal{A} \in$ EST(τ), (e.g., $\mathcal{A} = \langle\{0, \ldots, n-1\}, R^{\mathcal{A}}\rangle$)
$\mathcal{A} \models \phi$ si, y sólo si, $\mathcal{A}$ está linealmente ordenado por $R^{\mathcal{A}}$
□

El problema definido por una sentencia ϕ en PO(τ) es
$$\text{MOD}(\phi) = \{\mathcal{A} \in \text{EST}(\tau) : \mathcal{A} \models \phi\}$$
Si S es un conjunto de τ-estructuras, es decir $S \subseteq$ EST(τ), decimos que S es *definible* en PO(τ) si existe una sentencia $\phi \in$ PO(τ) tal que $S =$ MOD(ϕ).

El siguiente teorema es parte del fólclore de la Teoría de Modelos Finitos y se demuestra por inducción en fórmulas (ver[5,14]).

Teorema 11 *Los problemas definibles por sentencias de primer orden están contenidos en* **L**. □

El enunciado del teorema anterior se expresará simbólicamente como PO $\subseteq$ **L**. Más adelante definiremos otras lógicas diferentes de PO que *capturarán* las clases de complejidad computacional por encima de **L**. Formalizaremos ese concepto de captura ahora.

Definición 12 Si $\mathcal{L}$ es una lógica y $\mathbf{C}$ es una clase de complejidad, entonces $\mathcal{L} \subseteq \mathbf{C}$ es una abreviación de lo siguiente: dada una sentencia ϕ en $\mathcal{L}$, el problema definido por ϕ, MOD(ϕ) (o, siendo más formales, la codificación como sucesiones de 0 y 1 de MOD(ϕ)), pertenece a $\mathbf{C}$. Similarmente, $\mathbf{C} \subseteq \mathcal{L}$ abreviará el que para cualquier problema K en $\mathbf{C}$ existe una sentencia ϕ en $\mathcal{L}$ tal que $K =$ MOD(ϕ). Si $\mathcal{L} \subseteq \mathbf{C}$ y $\mathbf{C} \subseteq \mathcal{L}$ escribimos entonces $\mathcal{L} = \mathbf{C}$, y decimos que la lógica $\mathcal{L}$ *captura* la clase de complejidad $\mathbf{C}$. □

La lógica de primer orden no es lo suficientemente expresiva como para capturar la clase **L**. Para ver esto considere el problema

$$\text{PAR} = \{\mathcal{A} \in \text{EST}(\tau_\epsilon) : |A| \text{ es par } \}$$

el cual es decidible en **L** (τ_ϵ es el vocabulario vacío). Sin embargo, PAR no es definible en PO, aún con respecto a conjuntos ordenados, como se sigue de la siguiente proposición. (Nos interesa incluir una relación de orden porque recuerde que al considerar las estructuras finitas como datos a ser leídos por una máquina de Turing, estas deben ser ordenadas.)

Proposición 13 *Si $\mathcal{A}$ y $\mathcal{B}$ son órdenes lineales de cardinalidad $\geq 2^n$, entonces $\mathcal{A}$ y $\mathcal{B}$ satisfacen las mismas sentencias de primer orden con no más de n cuantificadores.* □

En particular, si ϕ es una PO-sentencia que define órdenes lineales de cardinalidad par, n es el número de cuantificadores en ϕ, $\mathcal{A}$ es un orden lineal de cardinalidad 2^n, $\mathcal{B}$ es un orden lineal de cardinalidad 2^{n+1}, entonces se debe tener $\mathcal{A} \models \phi$ y $\mathcal{B} \not\models \phi$, pero por Proposición 13, tanto $\mathcal{A}$ como $\mathcal{B}$ satisfacen ϕ, lo cual es una contradicción. Por lo tanto, PAR no es definible en PO, aún respecto de conjuntos ordenados.

Así:

$$\text{PO} \subsetneq \mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PESPACIO}$$

La demostración de Proposición 13 utiliza una herramienta popular en la teoría de modelos finitos: los juegos de Ehrenfeucht–Fraissé, los cuales no presentaremos por ser irrelevantes al objetivo central de este trabajo. Estos se pueden estudiar en el libro[5]. Pasemos ahora a revisar el concepto de reducibilidad desde la perspectiva de la Lógica.

3.2. Reducibilidad en términos lógicos

En lo sucesivo $\mathcal{L}$ denota una lógica cualquiera (por los momentos sólo conocemos PO, pero sólo unas pocas páginas nos separan de otras lógicas). Comenzamos con el análogo lógico de una (τ, σ)–traductora.

Definición 14 Sea τ un vocabulario cualquiera y sea $\sigma = \{R_1, \ldots, R_r, C_1, \ldots, C_s\}$ un vocabulario donde cada R_i es un símbolo relacional de aridad n_i y cada C_j es un símbolo constante. Sea $\overline{z}$ una sucesión finita de variables y k un entero positivo. Una (τ, σ)–*traslación de aridad k y parámetro $\overline{z}$ de* EST(τ) *en* EST(σ) *descriptible en* $\mathcal{L}(\tau)$, es una ley definida por un conjunto Σ de fórmulas en $\mathcal{L}(\tau)$ que asigna a cada τ–estructura $\mathcal{A}$ y cada interpretación $\overline{a}$ de $\overline{z}$ en $\mathcal{A}$ una única (salvo isomorfismos) σ–estructura $\mathcal{A}_\Sigma$. El conjunto Σ tiene la forma

$$\Sigma := \{\phi_1(\overline{x}_1), \ldots, \phi_r(\overline{x}_r), \psi_1(\overline{y}_1), \ldots, \psi_s(\overline{y}_s)\},$$

donde cada ϕ_i es una $\mathcal{L}(\tau)$–fórmula con un vector $\overline{x}_i = (x_{i_1}, \ldots, x_{i_{kn_i}})$ de kn_i variables distintas y posiblemente otras variables tomadas de $\overline{z}$, y cada ψ_j es una $\mathcal{L}(\tau)$–fórmula con un vector $\overline{y}_j = (y_{j_1}, \ldots, y_{j_k})$ de k variables distintas y posiblemente otras variables tomadas de $\overline{z}$. La σ–estructura $\mathcal{A}_\Sigma$ se construye a partir de la τ–estructura $\mathcal{A}$, $\overline{a}$ y Σ de la siguiente manera:

- el universo de $\mathcal{A}_\Sigma$ es A^k;
- las relaciones (según σ) son, para cada $i = 1, \ldots, r$:

$$R_i^{\mathcal{A}_\Sigma} := \{\overline{u} \in A^{kn_i} : \langle \mathcal{A}, \overline{a}, \overline{u}\rangle \models \phi_i(\overline{x}_i)\}$$

- las constantes son, para cada $j = 1, \ldots, s$ y para todo $\overline{u} \in A^k$:

$$C_j^{\mathcal{A}_\Sigma} := \overline{u} \iff \langle \mathcal{A}, \overline{a}, \overline{u}\rangle \models \psi_j(\overline{y}_j) \wedge \forall v_1, \ldots, \forall v_k(\neg\psi_j(v_1, \ldots, v_k) \vee (v_1 = y_{j1} \wedge \ldots \wedge v_k = y_{jk})).$$

(La fórmula anterior asegura que $C_j^{\mathcal{A}_\Sigma}$ es único y, por lo tanto, bien definido.) □

Definición 15 Sean $\mathcal{L}$ una lógica, τ y σ dos vocabularios, K un problema sobre τ y Q un problema sobre σ. Decimos que K *es $\mathcal{L}$–reducible a* Q (lo que denotamos por $K \leq_{\mathcal{L}} Q$) si existe un entero $k > 0$ y un conjunto de fórmulas $\Sigma \subseteq \mathcal{L}(\tau)$ que definen una (τ, σ)–traslación de aridad k y parámetro $\overline{z}$ de EST(τ) en EST(σ), tal que, para toda τ–estructura $\mathcal{A}$ y cualquier interpretación $\overline{a}$ de $\overline{z}$ en $\mathcal{A}$,

$$\langle \mathcal{A}, \overline{a}\rangle \in K \text{ si y sólo si } \mathcal{A}_\Sigma \in Q. \quad \square$$

En particular, si $\mathcal{L}$ en la Definición 15 es PO tenemos entonces reducciones de primer orden, las cuales son más débiles que las reducciones computables en espacio logarítmico de acuerdo con lo demostrado en 3.1.

3.3. Satisfacción cuantificada

Retornemos a la lógica proposicional permitiéndonos ahora cuantificar variables. Una proposición φ con n variables $x_1, \ldots, x_n$ cuantificadas, por ejemplo, en la forma $\exists x_1 \forall x_2 \exists x_3 \ldots Q x_n \varphi$ ($Q \in \{\exists, \forall\}$) es satisfacible si "para alguno de los dos valores de verdad que puede tomar x_1 (i.e. 0 o 1), se tiene que para ambos valores de verdad que puede tomar x_2, se tiene que para alguno de los dos valores de verdad que puede tomar x_3, ..., etc., se concluye que φ es verdad". Por ejemplo la fórmula φ en (1) de la sección 2.3 cuantificada de la siguiente manera:

$$\exists x_1 \forall x_2 \exists x_3 \varphi := \exists x_1 \forall x_2 \exists x_3 [(\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (x_2 \vee \neg x_3)]$$

es satisfacible: vimos ya que existe un valor para x_1 y uno para x_3 tales que cualquiera sea el valor de x_2, φ es satisfacible.

Un pariente de SAT y de aparente mayor complejidad es QSAT. Una instancia del problema QSAT es una fórmula proposicional en fnc con todas sus variables cuantificadas por $\exists$ o $\forall$. Una instancia positiva es una instancia que es satisfacible.

Para representar QSAT como una clase de estructuras finitas, el vocabulario adecuado es $\tau_{qsat} = \{P, N, U\}$, donde P y N son los mismos predicados binarios definidos en la sección 2.3 y U es un predicado unario con el siguiente significado: Si $\mathcal{A}$ es una τ_{qsat}–estructura entonces

> $i \in U^{\mathcal{A}}$ si y sólo si la variable i está universalmente cuantificada.

Luego

$$\text{QSAT} = \{\mathcal{A} \in \text{EST}(\tau_{qsat}) : \mathcal{A} \text{ es una proposición cuantificada en fnc que es satisfacible}\}.$$

QSAT fue el primer problema encontrado **PESPACIO**–log–completo por L. Stockmeyer y A. Meyer (ver[9]). A partir de la completitud de QSAT se han demostrado que otros problemas son **PESPACIO**–completos. En particular, muchos problemas sobre juegos de información perfecta entre dos jugadores que toman turnos alternativamente, puesto que QSAT se puede ver como uno de tales juegos: Dada

$$\Phi := Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \phi$$

donde ϕ es una fórmula proposicional en fnc con las variables $x_1, \ldots, x_n$, los cuantificadores $Q_i \in \{\exists, \forall\}$ $(1 \leq i \leq n)$ que se alternan entre $\exists$ y $\forall$, comenzando con $Q_1 = \exists$ (lo cual no es una pérdida de generalidad ya que siempre podemos añadir cláusulas de la forma $x \vee \neg x$ sin que se altere el valor de verdad de ϕ). Se tienen dos jugadores, $\exists$ (el jugador existencial) y $\forall$ (el jugador universal), que toman turnos y asignan un valor de 0 (falso) o 1 (verdadero) a cada variable, comenzando con x_1 y siendo $\exists$ el primero en jugar. Por lo tanto, $\exists$ asigna valores de verdad a las variables existencialmente cuantificadas, mientras que $\forall$ hace lo mismo con las variables universalmente cuantificadas en Φ. Llamemos a este juego Qsat. Decimos que $\exists$ gana el juego de Qsat sobre Φ si y sólo si después del n–ésimo movimiento ϕ es verdadera. Entonces

> $\mathcal{A} \in$ QSAT si y sólo si $\exists$ tiene una estrategia ganadora en el juego de Qsat sobre $\mathcal{A}$.

Con esta perspectiva de QSAT como un juego podemos demostrar que WHEX es **PESPACIO**–completo respecto de $\leq_{\log}$.

Teorema 16 QSAT $\leq_{\log}$ WHEX.

Demostración: Debemos transformar fórmulas proposicionales cuantificadas Φ en grafos G_Φ, de manera que:

$(*)$**:** Φ es satisfacible si, y sólo si, Jugador 1 gana el juego de Whex sobre G_Φ y ciertos vértices s y t.

Sea $\Phi := \exists x_1 \forall x_2 \ldots Q_n x_n (C_1 \wedge C_2 \wedge \ldots \wedge C_m)$, donde cada cláusula C_i es una disyunción de literales x o $\neg x$. Definimos el grafo $G_\Phi = (V, E)$ de la siguiente manera:

$$
\begin{aligned}
V &= \{s, t, y\} \cup \{x_i, \overline{x}_i, u_i, \overline{u}_i, v_i : 1 \leq i \leq n\} \\
&\cup \{w_{2i} : 1 \leq i \leq \lceil n/2 \rceil\} \cup \{c_i, z_i : 1 \leq i \leq m\}
\end{aligned}
$$

$$
\begin{aligned}
E &= \{(s, x_1), (s, \overline{x}_1), (v_n, c_1), (v_n, z_1), (z_m, t), (y, t)\} \\
&\cup \{(x_i, u_i), (\overline{x}_i, \overline{u}_i), (u_i, t), (\overline{u}_i, t), (x_i, v_i), (\overline{x}_i, v_i) : \\
&\quad 1 \leq i \leq n\} \\
&\cup \{(v_i, x_{i+1}), (v_i, \overline{x}_{i+1}) : 1 \leq i \leq n-1\} \\
&\cup \{(v_{2i}, w_{2i}), (w_{2i}, t) : 1 \leq i \leq \lceil n/2 \rceil\} \\
&\cup \{(z_i, z_{i+1}), (z_i, c_{i+1}) : 1 \leq i \leq m-1\} \\
&\cup \{(c_i, y) : 1 \leq i \leq m\} \\
&\cup \{(c_i, u_j) : \text{ el literal } \neg x_j \text{ está en cláusula } C_i\} \\
&\cup \{(c_i, \overline{u}_j) : \text{ el literal } x_j \text{ está en cláusula } C_i\}
\end{aligned}
$$

(La Figura 1 es el grafo G_Φ para

$\Phi := \exists x_1 \forall x_2 \exists x_3 [(\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (x_2 \vee \neg x_3)]$).

Los movimientos de Jugador 1 y Jugador 2 sobre el grafo G_Φ corresponden a movimientos de $\exists$ y $\forall$ sobre Φ. El literal $\overline{x}$ denota la negación de x; por lo tanto el que Jugador 1 coloree x_{2i-1} o $\overline{x}_{2i-1}$ corresponde a $\exists$ asignar el valor de 0 o 1 a x_{2i-1}. De manera similar las coloraciones de Jugador 2 corresponden a las asignaciones de $\forall$.

No es difícil ver que la construcción de G_Φ a partir de Φ se puede realizar determinísticamente y usando espacio a lo sumo logarítmico en el número de cláusulas y literales en Φ. La verificación de la afirmación $(*)$ se deja como ejercicio, pero si le resulta difícil hacerlo vea los detalles en 3. □

Un poco más difícil (y de mayor interés) es probar que la reducción de QSAT a WHEX es expresable en la lógica de primer orden. Se puede intentar el método directo de expresar la reducción descrita arriba por fórmulas de primer orden. Esto involucra, entre otras cosas, codificar los elementos de V como tuplas compuestas por símbolos constantes del vocabulario, lo cual determinará la aridad de la reducción. Si bien esta solución parece simple no siempre es realizable, lo cual justifica investigar otras posibles maneras de refinar reducciones logarítmicas a reducciones describibles en PO. Una de estas alternativas la obtendremos como corolario de los resultados que presentamos en la sección 4. Allí indicaremos como obtener la completitud de WHEX respecto a reducciones de primer orden sin cuantificadores y con una forma muy particular.

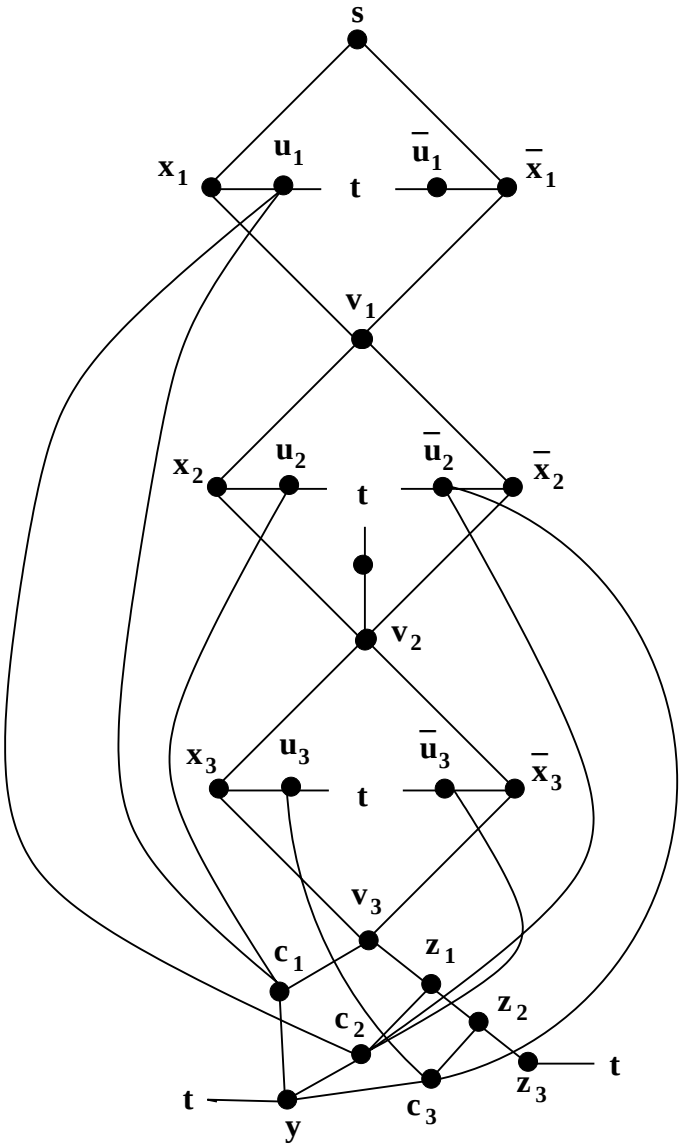

Figura 1: G_Φ para
$\Phi := \exists x_1 \forall x_2 \exists x_3 [(\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (x_2 \vee \neg x_3)]$.

3.4. Un paradigma para capturar clases de complejidades

Dada una clase de complejidad computacional $\mathbf{C}$ y una lógica $\mathcal{L}$, para demostrar que $\mathcal{L} = \mathbf{C}$, es decir que $\mathcal{L}$ captura $\mathbf{C}$ (ver Definición 12), procederemos de acuerdo con el siguiente esquema:

1. Para demostrar que $\mathcal{L} \subseteq \mathbf{C}$ indicar, para cada sentencia θ de $\mathcal{L}$, como construir un algoritmo o máquina de Turing M_θ, con las restricciones adecuadas para procesar sólo problemas en $\mathbf{C}$, que acepta únicamente la codificación de MOD(θ). Esta contención es, por lo general, la más fácil de demostrar.

2. Para demostrar que $\mathbf{C} \subseteq \mathcal{L}$ hacer lo siguiente:

   a) Hallar un problema Ω cuya codificación sea un

lenguaje en $\mathbf{C}$ y tal que sea completo en $\mathbf{C}$ via reducciones de primer orden.

b) Demuestre que Ω es expresable en $\mathcal{L}$.

c) Demuestre que $\mathcal{L}$ es cerrado bajo reducciones de primer orden. □

Puesto que PO $\subsetneq$ **L**, debemos buscar lenguajes más expresivos que la lógica de primer orden para capturar las clases de complejidad por encima de (e incluyendo) **L**. En la próxima sección estudiaremos una manera de incrementar el poder expresivo de PO que, a la vez, nos dará un método para demostrar que un problema es completo en **C** via reducciones de primer orden. Este método consiste en considerar nuestro problema escogido Ω en **C** como un operador que actúa sobre fórmulas de PO para formar otras fórmulas que, esencialmente, describen problemas reducibles a Ω via $\leq_{PO}$. Al demostrar que nuestro lenguaje (PO + Ω como operador) captura **C** tendremos como bono extra que Ω es PO–completo para **C**. Así, los resultados de la próxima sección no sólo explican una manera de extender el poder expresivo de PO, sino también forman nuestro paradigma para demostrar que un problema es PO–completo, que junto con el paradigma de captura constituyen nuestras herramientas para describir clases de complejidad espacial o temporal por lógicas.

4. Cuantificadores Generalizados o de Lindström.

En un trabajo de Per Lindström, publicado en 1966 (ver[16]), se describe una manera de asociar a cualquier clase de estructuras (finitas o infinitas), cerrada bajo isomorfismos, un operador o *cuantificador generalizado* que agregado a un lenguaje de primer orden produce una extensión lógica donde la clase de estructuras en cuestión es definible. Esta idea de Lindström fue empleada por Neil Immerman en los 80 (del siglo XX) para definir extensiones de PO que capturan las clases **L**, **NL** y **P** (ver[12]), y, posteriormente, otros investigadores continuaron el proyecto iniciado por Immerman lográndose completar la descripción de todas las clases de complejidad computacionales por extensiones de PO con cuantificadores generalizados. En esta sección presentamos estas construcciones lógicas y sus aplicaciones.

4.1. La Lógica Ω^*[PO].

Definición 17 Sea τ un vocabulario cualquiera y sea $\sigma = \{R_1, \ldots, R_r, C_1, \ldots, C_s\}$ otro vocabulario donde cada R_i es un símbolo relacional de aridad n_i y cada C_j es un símbolo constante. Sea Ω un problema sobre σ. *La extensión de* PO(τ) *con el cuantificador* Ω, denotada $\Omega^*[\text{PO}(\tau)]$, es el menor conjunto de fórmulas tal que:

(i) PO(τ) $\subseteq \Omega^*[\text{PO}(\tau)]$;

(ii) si $\psi, \phi \in \Omega^*[\text{PO}(\tau)]$ entonces $\neg(\phi)$, $(\psi \wedge \phi)$, $(\psi \vee \phi)$, $\exists z(\psi)$ y $\forall z(\psi)$ también son fórmulas de $\Omega^*[\text{PO}(\tau)]$;

(iii) si $\Sigma := \{\phi_1(\overline{x}_1), \ldots, \phi_r(\overline{x}_r), \psi_1(\overline{y}_1), \ldots, \psi_s(\overline{y}_s)\} \subseteq \Omega^*[\text{PO}(\tau)]$ define una (τ, σ)–traslación de aridad k y parámetro $\overline{z}$ de EST(τ) en EST(σ) (recuerde la definición 14), entonces la siguiente es una nueva fórmula en $\Omega^*[\text{PO}(\tau)]$:

$$\Phi(\overline{z}) := \Omega[\overline{x}_1, \ldots, \overline{x}_r, \overline{y}_1, \ldots, \overline{y}_s : \phi_1(\overline{x}_1), \ldots, \phi_r(\overline{x}_r), \psi_1(\overline{y}_1), \ldots, \psi_s(\overline{y}_s)]$$

donde las variables en cada tupla $\overline{x}_i$ y $\overline{y}_j$ están acotadas en Φ por el cuantificador Ω y, en consecuencia, las variables libres de Φ (contenidas en el parámetro $\overline{z}$) son las variables libres en algún ϕ_i y algún ψ_j distintas de las que aparecen en $\overline{x}_i$ y $\overline{y}_j$ respectivamente.

Semántica: La interpretación de las fórmulas obtenidas según (i) y (ii) se define de la manera usual. La interpretación de Φ obtenida según (iii) se define de la siguiente manera: si $\mathcal{A} \in$ EST(τ) y $\overline{a}$ es una interpretación de $\overline{z}$ en A (obviamente $|\overline{a}| = |\overline{z}|$), entonces $\langle \mathcal{A}, \overline{a} \rangle \models \Phi(\overline{z})$ si, y sólo si, la (τ, σ)–traslación de $\mathcal{A}$ descrita por Σ, es decir, la estructura $\mathcal{A}_\Sigma \in$ EST(σ), es tal que $\mathcal{A}_\Sigma \in \Omega$.

La extensión de la lógica de primer orden PO con el cuantificador Ω es el lenguaje

$$\Omega^*[\text{PO}] := \bigcup_\tau \{\Omega^*[\text{PO}(\tau)] : \tau \text{ es un vocabulario}\}$$

Fragmentos de Ω^*[PO]: Sea n un entero positivo. Definimos:

Ω^n[**PO**] es el lenguaje Ω^*[PO] excepto que a lo sumo n aplicaciones del cuantificador Ω pueden estar anidadas.

posΩ^*[**PO**] denota el lenguaje Ω^*[PO] donde ninguna aplicación de Ω está en el alcance de $\neg$ (i.e., todas las apariciones de Ω son *positivas*).

posΩ^n[**PO**](τ) denota el lenguaje Ω^*[PO] donde a lo sumo n aplicaciones de Ω pueden estar anidadas y todas positivas.

Observación 18 Hemos utilizado el mismo símbolo Ω para denotar un problema y, a la vez, el cuantificador asociado a ese problema. Esto no debería causar confusión ya que será siempre claro el sentido en que consideramos Ω a partir del contexto. Por otra parte, este abuso de notación resulta en una sana simplificación de la simbología al evitarnos múltiples subíndices y nombres. □

Ejemplo 19 Los cuantificadores $\exists$ y $\forall$ pueden verse como casos particulares de la construcción de Lindström. Sea $\sigma_1 = \{U\}$ donde U es una relación unaria y considere las siguientes clases de σ_1–estructuras:

$$\exists = \{\langle A, B\rangle : B \subseteq A \text{ y } B \neq \emptyset\} \text{ y}$$
$$\forall = \{\langle A, B\rangle : B = A\}$$

Si $\phi(x) \in$ PO(τ), $\mathcal{A}$ es una τ-estructura cualquiera y

$$\phi^{\mathcal{A}} := \{a \in A : \mathcal{A} \models \phi(a)\}$$

entonces

$$\mathcal{A} \models \exists x\phi(x) \iff \langle A, \phi^{\mathcal{A}}\rangle \in \exists \text{ y}$$
$$\mathcal{A} \models \forall x\phi(x) \iff \langle A, \phi^{\mathcal{A}}\rangle \in \forall$$

Por supuesto que $\exists^*$[PO] y $\forall^*$[PO] son exactamente PO. □

Ejemplo 20 En el Ejemplo 3 se definieron las clases de τ_2–estructuras DTC, TC, HP y WHEX, y la clase de τ_3–estructuras PS. Cada una de estas clases da origen a una extensión de PO donde es posible definirlas. Se tienen los lenguajes DTC*[PO], TC*[PO], PS*[PO], HP*[PO] y WHEX*[PO], y si, por ejemplo, $\mathcal{A} = \langle A, E^{\mathcal{A}}, s, t\rangle$ es una τ_2–estructura cualquiera (aquí $s = C^{\mathcal{A}}$ y $t = D^{\mathcal{A}}$), entonces

$$\mathcal{A} \in \text{TC} \iff \mathcal{A} \models \text{TC}[x, y, u, v : E(x, y), u = C, v = D].$$

Fórmulas análogas definen DTC, PS, HP y WHEX. □

En general, si Ω es un problema sobre $\sigma = \{R_1, \ldots, R_r, C_1, \ldots, C_s\}$, donde cada R_i es un símbolo relacional y cada C_j un símbolo constante, entonces Ω es definible en posΩ^1[PO] por la fórmula

$$\Omega[\overline{x}_1, \ldots, \overline{x}_r, y_1, \ldots, y_s : R_1(\overline{x}_1), \ldots, R_r(\overline{x}_r), y_1 = C_1, \ldots, y_s = C_s].$$

Observación 21 La fórmula

$$\Omega[\overline{x}_1, \ldots, \overline{x}_r, y_1, \ldots, y_s : R_1(\overline{x}_1), \ldots, R_r(\overline{x}_r), y_1 = C_1, \ldots, y_s = C_s]$$

se abreviará por

$$\Omega[\overline{x}_1, \ldots, \overline{x}_r : R_1(\overline{x}_1), \ldots, R_r(\overline{x}_r)](C_1, \ldots, C_s). \quad \square$$

Es fácil ver que posΩ^1[PO] es la mínima extensión de PO donde el problema Ω es definible:

Proposición 22 *Supóngase que $\mathcal{L}$ es una lógica cerrada bajo conectores y cuantificadores de primer orden, cerrada bajo sustituciones de símbolos relacionales por fórmulas del lenguaje y donde Ω es definible. Entonces* posΩ^1[PO] $\subseteq \mathcal{L}$. □

Una lógica $\mathcal{L}$ que satisface las propiedades de clausura descritas en la proposición anterior se denomina *regular*. PO y casi todas las extensiones de PO que estudiaremos son lógicas regulares.

Ejemplo 23 El problema DTC es definible en posTC^1[PO] por la sentencia

$$\theta := \text{TC}[x, y : (E(x, y) \wedge \forall z(E(x, z) \longrightarrow z = y))](C, D)$$

(El lector debe verificar que si $\mathcal{A}$ es una τ_2–estructura, entonces $\mathcal{A} \models \theta$ si, y sólo si, $\langle A, E^{\mathcal{A}}\rangle$ es un grafo donde hay un camino entre $C^{\mathcal{A}}$ y $D^{\mathcal{A}}$ cuyos vértices tienen todos grado exterior 1.)

Por lo tanto, DTC*[PO] $\subseteq$ TC*[PO] (y posDTC*[PO] $\subseteq$ posTC*[PO]). □

Algo un poco más interesante es ver que el complemento del problema DTC es definible en posDTC*[PO]:

Ejemplo 24 DTC*[PO] = posDTC*[PO]. Considere la siguiente negación de una fórmula en DTC*[PO]:

$$\Phi := \neg\text{DTC}[\overline{x}, \overline{y} : \phi(\overline{x}, \overline{y})](\overline{a}, \overline{b})$$

donde $\overline{a}$ y $\overline{b}$ son k–uplas formadas con los símbolos constantes C y D. Observe que Φ es cierta si ningún camino que comience en $\overline{a}$ y alcance $\overline{b}$ es determinístico, o si no es posible llegar a $\overline{b}$ desde $\overline{a}$. Veremos que estas condiciones se pueden expresar en posDTC*[PO]. Comencemos por definir

$$\theta(\overline{u}, \overline{z}, \overline{b}) := \neg(\overline{u} = \overline{b}) \wedge \text{DTC}[\overline{x}, \overline{y} : \phi(\overline{x}, \overline{y})](\overline{u}, \overline{z}) \wedge \text{DTC}[\overline{x}, \overline{y} : (\phi(\overline{x}, \overline{y}) \wedge \neg(\overline{y} = \overline{b}))](\overline{u}, \overline{z})$$

La fórmula $\theta(\overline{u}, \overline{z}, \overline{b})$ dice que el camino determinístico descrito por ϕ comienza en $\overline{u}$ y alcanza $\overline{z}$ sin pasar por $\overline{b}$. Ahora bien, existen tres posibilidades para un camino determinístico descrito por ϕ que comience en $\overline{a}$ y no pase por $\overline{b}$: el camino alcanza un vértice sin arcos saliendo de él, o un vértice tiene más de un arco saliendo de él, o se alcanza un ciclo. Teniendo todo esto en cuenta podemos ver que Φ es equivalente a

$$\exists\overline{z}\,\big(\theta(\overline{a}, \overline{z}, \overline{b}) \wedge [\forall\overline{u}(\neg\phi(\overline{z}, \overline{u}) \vee \exists\overline{v}(\neg(\overline{v} = \overline{u}) \wedge \phi(\overline{z}, \overline{v}))) \vee \exists\overline{u}(\neg(\overline{u} = \overline{z}) \wedge \theta(\overline{z}, \overline{u}, \overline{b}) \wedge \theta(\overline{u}, \overline{z}, \overline{b}))]\big)$$

y esta es una fórmula en posDTC*[PO]. □

Ejemplo 25 Sea $\tau = \{suc(\cdot, \cdot), 0, max\}$ donde suc es una relación binaria, 0 y max constantes, y considere la siguiente sentencia en posDTC^1[PO(τ)]:

$$\Phi := \text{DTC}[(x_1, x_2), (y_1, y_2) : (suc(x_1, y_1) \wedge ((x_2 = 0 \wedge y_2 = max) \vee (x_2 = max \wedge y_2 = 0)))]((0, 0), (max, max)).$$

Si $\mathcal{A}$ es una τ-estructura donde $[suc(\cdot, \cdot)]^{\mathcal{A}}$ se interpreta como la relación de orden parcial *sucesor* ($suc(a, b)$ es cierto en $\mathcal{A}$ si y sólo si b es el sucesor de a), $0^{\mathcal{A}}$ es el menor elemento en el orden y $max^{\mathcal{A}}$ el último, y considerando el sucesor de $max^{\mathcal{A}}$ como $0^{\mathcal{A}}$, entonces

$\mathcal{A} \models \Phi$ si y sólo si

A es un conjunto ordenado de cardinalidad par.

Por lo tanto, si nos restringimos a estructuras ordenadas con un primer y último elemento en el orden, entonces el problema PAR es definible en DTC[PO]. □

Proposición 26 PO $\subsetneq$ DTC[PO]*, sobre estructuras ordenadas* □

El Ejemplo 25 nos indica que, en principio, debemos restringirnos a problemas sobre estructuras ordenadas para describir mediante alguna lógica una clase de complejidad por encima o igual a **L**.

Definición 27 PO$_s$ es la lógica de primer orden PO que además de los símbolos lógicos necesarios para construir fórmulas (e.g. $\wedge$, $\neg$, $\exists$ y $=$) contiene los símbolos extras $suc(\cdot,\cdot)$, 0 y max, los cuales se interpretarán siempre como la relación sucesor sobre los naturales, el primer elemento y el último elemento en cualquier estructura. Además asumiremos que estos símbolos no son parte de los distintos vocabularios finitos con los que se definen problemas en PO. □

Observe que si τ es un vocabulario finito y $\mathcal{A}$ es una τ–estructura, entonces cuando digamos que $\mathcal{A}$ satisface una fórmula en PO$_s(\tau)$, o en alguna extensión de PO$_s(\tau)$, se estará asumiendo dos cosas sobre $\mathcal{A}$: 1) su universo es (isomorfo a) un segmento inicial de los naturales ordenados por la relación sucesor sobre los naturales, y 2) $\mathcal{A}$ tiene al menos dos elementos distintos. Estas consideraciones nos evitan los casos patológicos de problemas que no son clases de estructuras cerradas bajo isomorfismos, debido a una interpretación libre del orden posible en cada estructura.

Definición 28 Sea τ un vocabulario y Ω un problema sobre τ. Ω^*[PO$_s$] es el lenguaje Ω^*[PO] junto con la relación predefinida sucesor (suc) y las constantes predefinidas 0 y max que, como indicamos, se interpretarán, respectivamente, como orden parcial, el primero y el último elemento en el orden. De manera análoga se definen los fragmentos Ω^n[PO$_s$], posΩ^*[PO$_s$] y posΩ^n[PO$_s$]. □

Tenemos entonces que posPAR*[PO$_s$] $\subseteq$ posDTC*[PO$_s$], y como PAR no es definible en PO, aún respecto a estructuras ordenadas, lo cual es equivalente a no ser definible en PO$_s$, concluimos:

Corolario 29 PO$_s \subsetneq$ posDTC^1[PO$_s$]. □

Si $\Omega_1, \ldots, \Omega_m$ son problemas sobre los vocabularios σ_1, $\ldots$, σ_m, respectivamente, entonces no es difícil definir la extensión de PO (o PO$_s$) por el conjunto de cuantificadores generalizados $\Gamma = \{\Omega_1, \ldots, \Omega_m\}$ de manera similar a la Definición 17. Se obtiene así el lenguaje Γ^*[PO] (o Γ^*[PO$_s$]) donde se pueden tener fórmulas con cualquier alternación de los cuantificadores en Γ, además de los cuantificadores de primer orden $\exists$ y $\forall$.

En lo sucesivo trabajaremos con $\mathcal{L}$ una lógica regular; en particular (y para fijar ideas) piense que $\mathcal{L}$ es PO, PO$_s$, o una extensión de éstas con un conjunto Γ de uno o más cuantificadores generalizados. En este momento al lector le será útil repasar la Definición 15 de $\mathcal{L}$–reducción. El siguiente lema y su corolario son bastante obvios y sus demostraciones se dejan como ejercicio.

Lema 30 *Sea $\mathcal{L}$ una lógica regular, τ y σ dos vocabularios, donde $\sigma = \{R_1, \ldots, R_r, C_1, \ldots, C_c\}$ con cada R_i de aridad n_i. Sean Ω_1 un problema sobre τ y Ω_2 un problema sobre σ. Entonces $\Omega_1 \leq_{\mathcal{L}} \Omega_2$, via una (τ,σ)–traslación de aridad K, si, y sólo si, $\Omega_1 =$* MOD(Φ) *para una sentencia Φ en* pos$\Omega_2^1[\mathcal{L}(\tau)]$ *de la forma $\Omega_2[\overline{x}_1, \ldots, \overline{x}_r, \overline{y}_1, \ldots, \overline{y}_c : \phi_1, \ldots, \phi_r, \psi_1, \ldots, \psi_c]$, donde $\{\phi_1, \ldots, \phi_r, \psi_1, \ldots, \psi_c\} \subset \mathcal{L}(\tau)$ constituye una (τ,σ)–traslación de aridad k.* □

Corolario 31 *Sean $\mathcal{L}_1$ y $\mathcal{L}_2$ dos lógicas regulares tales que $\mathcal{L}_1 \subseteq \mathcal{L}_2$, y sean Ω_1 y Ω_2 dos problemas. Si $\Omega_1 \leq_{\mathcal{L}_1} \Omega_2$ entonces $\Omega_1 \leq_{\mathcal{L}_2} \Omega_2$.* □

Lema 32 *Sean Ω_1 y Ω_2 dos problemas sobre los vocabularios τ y σ respectivamente, donde $\sigma = \{R_1, \ldots, R_r, C_1, \ldots, C_c\}$ con cada R_i de aridad n_i. Entonces $\Omega_1 \leq_{\log} \Omega_2$ si y sólo si $\Omega_1 \leq_{DTC^1[PO_s]} \Omega_2$.*

Demostración: (Se anexa al final de este trabajo. Ver sección 7). □

El lema anterior es la pieza fundamental para pasar de la definición de las distintas clases de complejidad espacial y temporal en términos de máquina de Turing, a una caracterización de estas clases en términos sintácticos como lenguajes formales. El siguiente teorema indica una manera de establecer ese puente entre la complejidad computacional y la complejidad descriptiva de problemas.

Teorema 33 *Sea $\mathbf{C}$ una clase de complejidad computacional por encima o igual a* **L** *y cerrada bajo log–reducciones. Sea Ω un problema sobre algún vocabulario τ y tal que su codificación como lenguaje en $\{0,1\}^*$, $cod_\tau(\Omega)$, está en $\mathbf{C}$ y es completo via log–reducciones. Entonces*

$$\text{pos}\Omega^1[\text{PO}_s] \subseteq \mathbf{C} \subseteq \text{pos}\Omega^1[\text{DTC}^1[\text{PO}_s]]$$

Más aún posΩ^1[DTC^1[PO$_s$]] $= \mathbf{C}$.

Demostración: Sea Φ una sentencia en posΩ^1[PO$_s(\sigma)$] para algún vocabulario $\sigma = \{R_1, \ldots, R_r, C_1, \ldots, C_c\}$. Debemos demostrar que cod_σ(MOD(Φ)) $\in \mathbf{C}$. El caso de interés es cuando Φ es de la forma $\Omega[\overline{x}_1, \ldots, \overline{x}_r, \overline{y}_1, \ldots, \overline{y}_c : \phi_1, \ldots, \phi_r, \psi_1, \ldots, \psi_c]$, donde $\{\phi_1, \ldots, \phi_r, \psi_1, \ldots, \psi_c\} \subset$ PO$_s(\tau)$ es una (τ,σ)–traslación de aridad k. En ese caso, por el Lema 30, MOD(Φ) $\leq_{PO_s} \Omega$ y por el Corolario 31 y el Lema 32 se tiene que cod_σ(MOD(Φ)) es log-reducible a $cod_\tau(\Omega)$; como $\mathbf{C}$ es cerrado bajo log-reducciones, concluimos que cod_σ(MOD(Φ)) $\in \mathbf{C}$. El lector debe demostrar los otros casos para Φ.

Sea $S \in \mathbf{C}$ un conjunto de palabras sobre $\{0,1\}$ que representa algún problema computacional. Por hipótesis $S \leq_{\log} cod_\tau(\Omega)$. A su vez, existe algún vocabulario σ y un problema $K \subseteq$ EST(σ) tal que $S = cod_\sigma(K)$. Por el Lema 32, $K \leq_{DTC^1[PO_s]} \Omega$ y por el Lema 30 $K =$ MOD(θ) para alguna sentencia θ en posΩ^1[DTC^1[PO$_s$](σ)].

Hemos demostrado que posΩ^1[PO$_s$] $\subseteq$ $\mathbf{C}$ $\subseteq$ posΩ^1[DTC^1[PO$_s$]]. Para ver que posΩ^1[DTC^1[PO$_s$]] $=$ $\mathbf{C}$, considere $\Phi \in$ posΩ^1[DTC^1[PO$_s$](σ)] y proceda como en la primera parte de esta demostración para concluir que $cod_\sigma(\text{MOD}(\Phi)) \in \mathbf{C}$. □

El teorema anterior sugiere la siguiente metodología para "aproximarnos", con un lenguaje de primer orden más cuantificadores generalizados, a una descripción sintáctica de una clase de complejidad $\mathbf{C}$, cerrada bajo log–reducciones y por encima o igual a **L**:

(1) Escoja su problema favorito K en $\mathbf{C}$ que sea completo via log–reducciones (existen largas listas de estos problemas para cada una de las clases de interés).

(2) Codifique K como un conjunto Ω_K de estructuras finitas sobre algún vocabulario (finito) τ.

(3) Defina la extensión posΩ^*_K[PO$_s$] de acuerdo con el esquema de Lindström.

Cumplidos estos tres pasos se tendrá

$$\text{pos}\Omega^1_K[\text{PO}_s] \subseteq \mathbf{C} \subseteq \text{pos}\Omega^1_K[\text{DTC}^1[\text{PO}_s]].$$

El siguiente paso es demostrar

(4) DTC es expresable en posΩ^1_K[PO$_s$], o, equivalentemente, DTC $\leq_{PO_s}$ Ω_K.

Es el caso, por ejemplo, cuando Ω_K = TC (Ejemplo 23); también cuando Ω_K es PS, HP o WHEX. Luego, si se cumplen los pasos (1) a (4), obtenemos

$$\text{pos}\Omega^1_K[\text{PO}_s] \subseteq \mathbf{C} \subseteq \text{pos}\Omega^2_K[\text{PO}_s]$$

y lograremos capturar $\mathbf{C}$ sintácticamente con posΩ^1_K[PO$_s$] si podemos demostrar

(5) posΩ^1_K[PO$_s$] $=$ posΩ^2_K[PO$_s$],

es decir, que sólo basta una aplicación del cuantificador Ω_K para expresar todo lo que se pueda expresar con una doble aplicación anidada de Ω_K.

Más aún, cualquiera sea el problema Ω, según la definición de Ω^*[PO$_s$] si posΩ^1[PO$_s$] $=$ posΩ^2[PO$_s$] entonces, para todo $k \geq 1$,

$$\begin{aligned}\text{pos}\Omega^{k+1}[\text{PO}_s] &= \text{pos}\Omega^k[\text{pos}\Omega^1[\text{PO}_s]] \\ &= \text{pos}\Omega^k[\text{pos}\Omega^2[\text{PO}_s]] = \text{pos}\Omega^{k+2}[\text{PO}_s],\end{aligned}$$

por lo que posΩ^1[PO$_s$] $=$ posΩ^2[PO$_s$] implica que posΩ^1[PO$_s$] $=$ posΩ^*[PO$_s$] (y, en general, Ω^1[PO] $=$ Ω^2[PO] implica Ω^1[PO] $=$ Ω^*[PO]). Al fenómeno Ω^1[PO] $=$ Ω^*[PO] le daremos un nombre:

Definición 34 Sea Ω un problema sobre $\tau = \{R_1, \ldots, R_r, C_1, \ldots, C_s\}$. La lógica Ω^*[PO] tiene una *forma normal de primer orden* si, para cualquier vocabulario σ, toda sentencia Φ en Ω^2[PO(σ)] es equivalente a una sentencia de la forma:

$$\Omega[\overline{x}_1, \ldots, \overline{x}_r, \overline{y}_1, \ldots, \overline{y}_s : \phi(\overline{x}_1), \ldots, \phi(\overline{x}_r), \psi(\overline{y}_1), \ldots, \psi(\overline{y}_s)]$$

donde cada ϕ_i y cada ψ_j es una fórmula en PO(σ). Si, además, cada ϕ_i y cada ψ_j es una fórmula *proyectiva*, entonces tenemos una *forma normal proyectiva*. Similarmente se define una forma normal (de primer orden o proyectiva) para posΩ^*[PO], Ω^*[PO$_s$] y posΩ^*[PO$_s$]. □

¿Qué es una fórmula proyectiva? La definimos a continuación.

Definición 35 Sea τ cualquier vocabulario. Una fórmula ϕ en PO(τ) (o en PO$_s(\tau)$) es una *proyección de primer orden (ppo)* si tiene la forma

$$\alpha_0 \vee (\alpha_1 \wedge \beta_1) \vee \ldots \vee (\alpha_m \wedge \beta_m)$$

para algún $m \geq 0$, donde

(i) cada α_i es una fórmula que no contiene símbolos de τ, y sólo utiliza $=$ u otro predicado numérico predefinido, si los hay (e.g. en el caso de PO$_s$ serían suc, 0 y max);

(ii) si $i \neq j$ entonces α_i y α_j son mutuamente excluyentes;

(iii) cada β_i es una fórmula atómica o su negación construida con símbolos de τ únicamente.

Si ninguna de las fórmulas α_i tiene cuantificadores entonces tenemos una *proyección libre de cuantificadores (plc)*. □

Observe que posΩ^1[PO$_s$] $=$ $\mathbf{C}$ es equivalente a que Ω sea completo para $\mathbf{C}$ via reducciones descriptibles en PO$_s$. Si además posΩ^*[PO$_s$] tiene una forma normal proyectiva, o una forma normal proyectiva sin cuantificadores, entonces Ω es completo para $\mathbf{C}$ por reducciones mucho más débiles, como son las proyecciones y las proyecciones libres de cuantificadores. Resulta difícil concebir reducciones más simples que las plc y aún tener problemas completos respecto a tales reducciones. Debemos hacer notar que las clases de complejidad de interés son cerradas bajo reducciones de primer orden y proyecciones libres de cuantificadores (ver[14]).

Resumimos todas nuestras observaciones anteriores en el siguiente teorema.

Teorema 36 *Sea $\mathbf{C}$ una clase de complejidad computacional tal que $\mathbf{C} \supseteq \mathbf{L}$ y es cerrada bajo log–reducciones, Sea Ω un problema sobre un vocabulario τ tal que $cod_\tau(\Omega) \in \mathbf{C}$ y es completo via log–reducciones. Entonces*

$$\text{pos}\Omega^1[\text{DTC}^1[\text{PO}_s]] = \mathbf{C}.$$

Si además DTC $\leq_{PO_s}$ Ω *o* DTC^1[PO$_s$] $\subseteq$ posΩ^k[PO$_s$], *para algún* $k \geq 1$, *entonces se tiene*

$$\text{pos}\Omega^1[\text{PO}_s] \subseteq \mathbf{C} \subseteq \text{pos}\Omega^{k+1}[\text{PO}_s].$$

Si además posΩ^1[PO$_s$] = posΩ^2[PO$_s$] *(i.e. hay una forma normal de primer orden), entonces*

$$\text{pos}\Omega^1[\text{PO}_s] = \text{pos}\Omega^*[\text{PO}_s] = \mathbf{C}$$

y Ω *es completo via* PO$_s$*–reducciones. Si la forma normal es proyectiva, entonces* Ω *es completo via proyecciones de primer orden.* □

Veamos que la lógica posTC*[PO$_s$] tiene una forma normal proyectiva.

Teorema 37 (N. Immerman[12]) *Sea* σ *un vocabulario. Cualquier sentencia* $\Phi \in$ posTC*[PO$_s(\sigma)$] *es equivalente a una sentencia de la forma* TC$[\overline{x}, \overline{y}, \overline{u}, \overline{v} : \psi(\overline{x}, \overline{y}), \overline{u} = \overline{0}, \overline{v} = \overline{max}]$, *donde* $\psi(\overline{x}, \overline{y})$ *es una* σ*–fórmula de primer orden proyectiva,* $\overline{x}$ *y* $\overline{y}$ *son* k*–tuplas de variables, para algún* $k \geq 1$, *y donde* $\overline{0}$ *y* $\overline{max}$ *son* k*–tuplas de las constantes pre-definidas* 0 *y* max *respectivamente.*

Demostración: Por inducción en la complejidad de Φ.
Caso 1: El caso más simple es cuando Φ es una sentencia en PO$_s(\sigma)$. Considere, entonces, dos nuevas variables x y y que no aparezcan en Φ. Entonces,

$$\models \Phi \iff \text{TC}[x, y : (\Phi \wedge x = x \wedge y = y)](0, max).$$

La afirmación anterior es cierta puesto que, dada una σ–estructura $\mathcal{A}$ que satisface Φ, obtenemos un grafo completo (i.e., el conjunto de sus arcos es todo A^2) y, en consecuencia, hay un camino desde $0^{\mathcal{A}}$ a $max^{\mathcal{A}}$. Si $\mathcal{A}$ no satisface Φ obtenemos un grafo sin arcos y, por lo tanto, no puede haber algún camino.
Caso 2: $\Phi :=$ TC$[\overline{x}, \overline{y} : \psi(\overline{x}, \overline{y})](\overline{C}, \overline{D})$. Aqui deseamos reemplazar las k–tuplas de constantes $\overline{C}$ y $\overline{D}$ por $\overline{0}$ y $\overline{max}$ respectivamente. Para lograr esto construimos una copia del grafo que define Φ donde cada vértice se etiqueta $(\overline{x}, 0, max)$, para evitar repeticiones, y se agregan dos nuevos arcos: $((\overline{0}, 0, 0), (\overline{C}, 0, max))$ y $((\overline{D}, 0, max), (\overline{max}, max, max))$. La fórmula proyectiva deseada es

$$\begin{aligned}\theta(\overline{x}, x_1, x_2, \overline{y}, y_1, y_2) :=\\ &(\overline{x} = \overline{0} \wedge x_1 = x_2 = y_1 = 0 \wedge \overline{y} = \overline{C} \wedge y_2 = max)\\ &\vee(\overline{x} \neq \overline{y} \wedge \psi(\overline{x}, \overline{y}) \wedge x_1 = y_1 = 0 \wedge x_2 = y_2 = max)\\ &\vee(\overline{x} = \overline{D} \wedge x_1 = 0 \wedge x_2 = y_1 = y_2 = max \wedge \overline{y} = \overline{max})\end{aligned}$$

Luego

$$\begin{aligned}&\models \Phi \iff\\ &\quad \text{TC}[(\overline{x}, x_1, x_2), (\overline{y}, y_1, y_2) : \theta]((\overline{0}, 0, 0), (\overline{max}, max, max))\end{aligned}$$

Caso 3: $\Phi := \forall z$TC$[\overline{x}, \overline{y} : \psi(\overline{x}, \overline{y}, z)](\overline{0}, \overline{max})$. La intuición aqui es que, para cada z, se tiene un grafo G_z definido por $\psi(\cdot, \cdot, z)$, y, sobre modelos finitos, un $\forall$ se reproduce con una conjunción finita (sobre todos los elementos del modelo). Luego la solución será conectar todos estos G_z posibles en serie. La fórmula que describe esta situación es la siguiente:

$$\begin{aligned}\theta(\overline{x}, x_1, \overline{y}, y_1) :=\\ &(\psi(\overline{x}, \overline{y}, x_1) \wedge x_1 = y_1)\\ &\vee(\overline{x} = \overline{max} \wedge \overline{y} = \overline{0} \wedge y_1 = x_1 + 1 \wedge x_1 \neq max)\end{aligned}$$

(recuerde que $y_1 = x_1 + 1$ es una abreviación de $suc(x_1, y_1)$). Así,

$$\models \Phi \iff \text{TC}[(\overline{x}, x_1), (\overline{y}, y_1) : \theta]((\overline{0}, 0), (\overline{max}, max))$$

La Figura 2 ilustra el grafo que define θ.

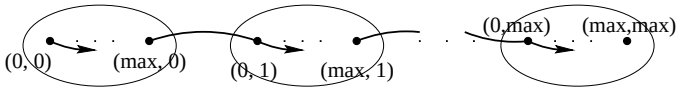

Figura 2: La solución para
$\Phi := \forall z$TC$[\overline{x}, \overline{y} : \psi(\overline{x}, \overline{y}, z)](\overline{0}, \overline{max})$.

Caso 4: $\Phi := \exists z$TC$[\overline{x}, \overline{y} : \psi(\overline{x}, \overline{y}, z)](\overline{0}, \overline{max})$. La idea es similar al caso anterior, sólo que esta vez $\exists$ se reproduce en modelos finitos como una disyunción, y esto sugiere conectar los G_z en paralelo. Luego

$$\begin{aligned}&\models \Phi \iff\\ &\quad \text{TC}[(\overline{x}, x_1), (\overline{y}, y_1) :\\ &\quad\{(\overline{x} = \overline{y} = \overline{0} \wedge x_1 = 0) \vee (\psi(\overline{x}, \overline{y}, x_1) \wedge x_1 = y_1)\\ &\quad\vee (\overline{x} = \overline{y} = \overline{max} \wedge y_1 = max)\}](\overline{0}, 0), (\overline{max}, max))\end{aligned}$$

La Figura 3 ilustra la construcción que resuelve este caso.

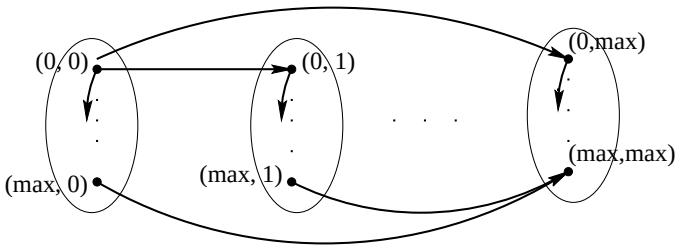

Figura 3: La solución para
$\Phi := \exists z$TC$[\overline{x}, \overline{y} : \psi(\overline{x}, \overline{y}, z)](\overline{0}, \overline{max})$.

Caso 5:
$\Phi :=$ TC$[\overline{x}, \overline{y} :$ TC$[\overline{x}', \overline{y}' : \psi(\overline{x}, \overline{y}, \overline{x}', \overline{y}')](\overline{0}, \overline{max})](\overline{0}, \overline{max})$.
Ahora, por cada par $(\overline{x}, \overline{y})$ que fijemos, tenemos un grafo $G_{\overline{x},\overline{y}}$ determinado por $\psi(\overline{x}, \overline{y}, \cdot, \cdot)$. Una estructura $\mathcal{A}$ de tamaño n producirá n^2 de estos grafos, los cuales arreglamos como una matriz $n \times n$ con $\overline{x}$ indicando las filas y $\overline{y}$ las columnas. Conectamos el nuevo vértice $(\overline{0}, \overline{0}, \overline{0})$ a cada uno de los primeros vértices en los grafos de la primera fila; estos son los de la forma $(\overline{0}, \overline{y}, \overline{0})$. Conectamos

el máximo de cada grafo en la i-ésima columna (excepto la última) con el primer elemento de cada grafo en la i-ésima fila. El máximo del grafo en la última columna y en cualquier fila, se conecta con un arco al nuevo vértice $(\overline{max}, \overline{max}, \overline{max})$. La fórmula (proyectiva) que describe esta construcción es la siguiente:

$$\begin{aligned}\theta(\overline{x},\overline{y},\overline{z},\overline{x}',\overline{y}',\overline{z}') :=& \\ & (\overline{x}=\overline{y}=\overline{0}\wedge\overline{z}=\overline{0}\wedge\overline{x}'=\overline{0}\wedge\overline{z}'=\overline{0}) \\ \vee\ & (\overline{x}\neq\overline{y}\wedge\overline{z}\neq\overline{max}\wedge\overline{x}=\overline{x}'\wedge\overline{y}=\overline{y}'\wedge\psi(\overline{x},\overline{y},\overline{z},\overline{z}') \\ \vee\ & (\overline{y}\neq\overline{max}\wedge\overline{z}=\overline{max}\wedge\overline{x}'=\overline{y}\wedge\overline{z}'=\overline{0}) \\ \vee\ & (\overline{y}=\overline{max}\wedge\overline{z}=\overline{max}\wedge\overline{x}'=\overline{y}'=\overline{max}\wedge\overline{z}'=\overline{max})\end{aligned}$$

Se tiene que

$$\begin{aligned}&\models\Phi\ \longleftrightarrow \\ &\quad \mathsf{TC}[(\overline{x},\overline{y},\overline{z}),(\overline{x}',\overline{y}',\overline{z}'):\theta](\overline{0},\overline{0},\overline{0}),(\overline{max},\overline{max},\overline{max})).\end{aligned}$$

Cualquier otro caso se deduce fácilmente por analogía con alguno de los cinco casos expuestos aqui. Por ejemplo, la conjunción de dos fórmulas se resuelve similar al caso $\forall$. □

El lector ya debe haber sospechado que buena parte de los argumentos en la demostración del Teorema 37 se pueden emplear para demostrar que posDTC*[PO$_s$] tiene una forma normal proyectiva. Sólo se debe tener cuidado de mantener siempre un camino determinístico. Así, por ejemplo, el caso existencial no se puede resolver conectando los grafos en paralelo como hicimos antes. Tenemos así el siguiente teorema debido a N. Immerman y publicado en[12].

Teorema 38 *Sobre estructuras finitas (ordenadas):*

(i) **L** $=$ posDTC^1[PO$_s$] $=$ posDTC*[PO$_s$] $=$ DTC*[PO$_s$].

(ii) **NL** $=$ posTC^1[PO$_s$] $=$ posTC*[PO$_s$].

Más aún, la forma normal de cada una de las lógicas es proyectiva y, en consecuencia, los problemas DTC *y* TC *son completos por reducciones de primer orden proyectivas.*

Demostración: Aplique el Teorema 36 a DTC y TC. Utilice los ejemplos 24, 23, el Teorema 37 y demuestre de manera análoga una forma normal para la lógica posDTC^1[PO$_s$]. □

Aplicamos la metodología expuesta en el Teorema 36 a los problemas PS, HP y WHEX, para así concluir lo siguiente.

Teorema 39 *Sobre estructuras finitas (ordenadas):*

(i) **P** $=$ posPS^1[PO$_s$] $=$ posPS*[PO$_s$] $=$ PS*[PO$_s$].

(i) **NP** $=$ posHP^1[PO$_s$] $=$ posHP*[PO$_s$].

(ii) **PESPACIO** $=$ posWHEX^1[PO$_s$] $=$ posWHEX*[PO$_s$] $=$ WHEX*[PO$_s$].

Más aún, la forma normal de cada una de las lógicas es proyectiva y, en consecuencia, los problemas PS, HP *y* WHEX *son completos por reducciones de primer orden proyectivas.* □

Los resultados sobre los cuantificadores generalizados PS y HP son de[21,20]. WHEX se definió y estudió en 3. A continuación indicaré al lector como demostrar el teorema anterior.

El problema TC es expresable en posPS^1[PO] por la siguiente razón: Si $\mathcal{A} = \langle A, E^{\mathcal{A}}, s, t\rangle$ es una τ_2–estructura, entonces un camino de s a t puede verse como el conjunto de triplas $R^{\mathcal{A}} = \{(s,s,s),(t,t,t)\} \cup \{(s,x,y) : E^{\mathcal{A}}(x,y)\}$. Para expresar TC en posWHEX^1[PO] observe que, dado un grafo G, se utilizan los arcos de G para construir una escalera con peldaños en diagonal, uniendo los dos extremos superiores a C y los dos extremos inferiores a D. La expresabilidad de TC en posHP*[PO$_s$] es un poco más difícil y se encuentra en[20].

Para demostrar que las lógicas posPS*[PO$_s$] y posWHEX*[PO$_s$] tienen una forma normal proyectiva, se procede de manera similar a la demostración del Teorema 37. Por ejemplo, para eliminar el cuantificador existencial en $\exists v\mathsf{PS}[\overline{x},\overline{y},\overline{z} : \psi(\overline{x},\overline{y},\overline{z},v)](\overline{0},\overline{max})$, debemos escribir una fórmula Φ tal que, para cada estructura $\mathcal{A}$ de cardinalidad n, $\Phi^{\mathcal{A}}$ describa un sistema de caminos formado por copias disjuntas de los sistemas de caminos descritos por $\psi^{\mathcal{A}}(\overline{x},\overline{y},\overline{z},0)$, $\psi^{\mathcal{A}}(\overline{x},\overline{y},\overline{z},1)$, ..., $\psi^{\mathcal{A}}(\overline{x},\overline{y},\overline{z},n-1)$, unidos a un nodo inicial $\mathbf{0}$ y a un nodo final $\mathbf{max}$ y con las reglas adicionales

$$\begin{gathered}(\mathbf{0},\mathbf{0},\overline{0}_0),\ (\mathbf{0},\mathbf{0},\overline{0}_1),\ \ldots,\ (\mathbf{0},\mathbf{0},\overline{0}_{n-1}) \\ (\overline{max}_0,\overline{max}_0,\mathbf{max}),\ (\overline{max}_1,\overline{max}_1,\mathbf{max}),\ \ldots, \\ (\overline{max}_{n-1},\overline{max}_{n-1},\mathbf{max})\end{gathered}$$

donde $\overline{0}_i$ (respectivamente, $\overline{max}_i$) es el nodo inicial (resp., final) del sistema de caminos descrito por $\psi^{\mathcal{A}}(\overline{x},\overline{y},\overline{z},i)$, para cada $i = 0,1,\ldots,n-1$.

Para WHEX la solución es incorporar el siguiente artificio a las construcciones hechas en la demostración del Teorema 37. Dado un grafo $G = (v,E)$ con nodos (inicial y final) s y t en V, construimos un nuevo grafo $X(G)$ a partir de una copia de G con dos nuevos nodos iniciales, s_1 y s_2, en lugar de s, dos nodos finales, t_1 y t_2, en lugar de t y un nuevo vértice w. Se trazan arcos entre s_1 (resp. s_2) y cada vértice que forma un arco con s en el G original; se hace lo mismo con t_1 (resp. t_2) y cada vértice que forma un arco con t en G. Se trazan arcos entre w y s_1, s_2, t_1 y t_2, respectivamente. El nuevo grafo $X(G)$ se ilustra en la Figura 4. Luego consideramos el juego de Whex sobre $X(G)$ como el juego usual de Whex sobre un grafo, salvo que la primera jugada de Jugador 1 debe ser colorear uno de los nodos iniciales s_1 o s_2, el juego termina coloreándose uno o los dos nodos finales y gana Jugador 1 si alcanza a colorear uno de los nodos finales. El lector debe verificar que Jugador 1 gana este nuevo juego de Whex sobre $X(G)$ si, y sólo si, Jugador 1 gana el juego usual de Whex sobre (G,s,t).

Ahora, supongamos que deseamos eliminar el cuantificador existencial en
$\exists z\text{WHEX}[\overline{x},\overline{y} : \psi(\overline{x},\overline{y},z)](\overline{0},\overline{max})$. Entonces procedemos como se hizo para TC, salvo que cada copia del grafo G_i, dada por $\psi(\overline{x},\overline{y},i)$, se sustituye por $X(G_i)$, uniéndose los dos nodos iniciales con el nodo inicial principal $\overline{0}$ y los dos nodos finales con el nodo final principal $\overline{max}$.

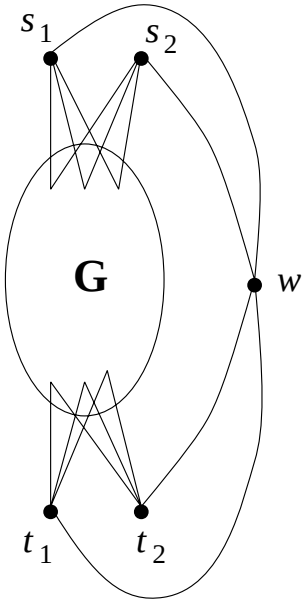

Figura 4: El grafo $X(G)$.

La forma normal para posHP*[PO$_s$] se puede estudiar en[20]. Intente el lector demostrar ese hecho. El caso difícil es la eliminación del cuantificador existencial. En principio se colocan los grafos en paralelo, como se hizo para TC, pero debe usted tener en cuenta que se desea un camino que recorra todos los puntos de todos estos grafos, en caso de que halla un camino hamiltoniano en alguno de ellos. □

5. Segundo Orden y Operadores Inductivos.

5.1. La Lógica de Segundo Orden (SO).

5.1.1. Sintáxis

Para obtener fórmulas de segundo orden, extendemos el conjunto **Var** de variables (de primer orden) con un conjunto $\mathbf{Var}^2$ de *variables de segundo orden* $X_1, X_2, X_3, \ldots$, las cuales simbolizan conjuntos o relaciones de cualquier aridad.

Sea τ un vocabulario. La *lógica de Segundo Orden sobre* τ, SO(τ), es el menor conjunto de fórmulas sobre el vocabulario τ que contiene a PO(τ), es cerrado bajo las operaciones booleanas, cerrado bajo cuantificación de primer orden y cerrado bajo *cuantificación de segundo orden*:

$$\exists X\phi \text{ y } \forall X\phi$$

donde X es variable de segundo orden de aridad n.

5.1.2. Semántica

Extendemos la definición de satisfacción (sección 3.1.2) a fórmulas de segundo orden de la siguiente manera. Para una τ-estructura $\mathcal{A}$, la interpretación de una variable de segundo orden $X \in \mathbf{Var}^2$ de aridad $k \geq 1$ es algún subconjunto en A^k. Luego, dada ϕ una fórmula en SO(τ), se define la relación $\mathcal{A} \models \phi$ inductivamente por (i)–(iv) de la sección 3.1.2 más lo siguiente:
Si X es una variable de segundo orden de aridad k,

(v) $\mathcal{A} \models \exists X\phi \iff$ para algún $T \subseteq A^k$, $\mathcal{A} \models \phi(\frac{X}{T})$, donde $\phi(\frac{X}{T})$ es la fórmula obtenida a partir de ϕ sustituyendo todas las apariciones de X por T,

(vi) $\mathcal{A} \models \forall X\phi \iff$ para todo $T \subseteq A^k$, $\mathcal{A} \models \phi(\frac{X}{T})$.

El conjunto de todas las fórmulas de segundo orden sobre cualquier vocabulario finito τ será indicado por SO. En símbolos,

$$\text{SO} = \bigcup_{\tau}\{\text{SO}(\tau) : \tau \text{ es un vocabulario finito }\}.$$

$\exists$SO (respectivamente $\forall$SO) es el fragmento de fórmulas de SO donde todos los cuantificadores de segundo orden son existenciales (respectivamente universales) y preceden a todos los demás símbolos. Es claro que, para cualquier vocabulario τ,

$$\text{PO}(\tau) \subseteq \exists\text{SO}(\tau) \subseteq \text{SO}(\tau)$$

Ejemplo 40 Sea $\tau_2 = \{E(\cdot,\cdot)\}$ y $\Phi \in \exists\text{SO}(\tau_2)$ definida por:

$$\begin{aligned}\Phi \;:=\; & \exists R(\text{“}R \text{ es un orden lineal”}) \wedge \\ & (\forall x\forall y((R(x,y) \wedge \forall z(\neg R(x,z) \vee \neg R(z,y))) \\ & \longrightarrow E(x,y))),\end{aligned}$$

donde R es una variable relacional binaria y “R es un orden lineal” es la sentencia en Ejemplo 10. Sea $\mathcal{A} \in$ EST(τ_2). $\mathcal{A}$ puede interpretarse como un grafo y:
$\mathcal{A} \models \Phi$ si, y sólo si, $\mathcal{A}$ tiene un camino hamiltoniano □

Ejemplo 41 El complemento de PS también se puede describir por una sentencia de $\exists$SO. Recuerde que las estructuras en PS se definen mediante un vocabulario que consiste de una relación ternaria R y dos constantes C y D. Luego la sentencia

$$\begin{aligned}\Phi \;:=\; & \exists H\forall x\forall y\forall z((\neg H(C) \vee \neg H(D)) \\ & \wedge ((H(x) \wedge H(y) \wedge R(x,y,z)) \longrightarrow H(z))),\end{aligned}$$

expresa la existencia de un conjunto H que contiene a todos los elementos obtenidos de acuerdo con la regla R (los "accesibles"), pero que no contiene simultáneamente C y D; por lo tanto, D no es accesible a partir de C. En consecuencia, una estructura $\mathcal{A}$ no está en PS si y sólo si $\mathcal{A} \models \Phi$. □

5.1.3. El Teorema de Fagin

El siguiente teorema, debido a Ronald Fagin y publicado en 1974 ([7]), es la génesis de la Teoría Descriptiva de la Complejidad Computacional y nos dice que no sólo el problema de determinar si un grafo es hamiltoniano es definible en ∃SO, sino también cualquier otro problema en **NP**. La demostración que presentamos ahora no es la que hizo Fagin y está ceñida a nuestro paradigma para establecer el puente entre clases de complejidad computacional y lógicas. Sin embargo, al hacerlo así, pareciera que en principio necesitamos restringirnos a estructuras ordenadas para poder capturar **NP** con ∃SO. Esto no es así. La lógica ∃SO es bastante fuerte en términos de expresabilidad, y allí podemos definir una relación de orden (véase el Ejemplo 40); por lo tanto el Teorema de Fagin es válido sobre todas las estructuras finitas.

Teorema 42 (R. Fagin[7]) $\exists\text{SO} = \textbf{NP}$.

Demostración: $(\subseteq)$: Sea $\Phi = (\exists R_1)\ldots(\exists R_k)\,\psi(R_1,\ldots,R_k)$, donde R_i tiene aridad a_i para $i = 1,\ldots,k$ y $\psi \in$ PO. Se necesitan n^{a_i} bits para describir una relación de aridad a_i en un universo de tamaño n. En consecuencia, es posible construir una máquina de Turing nodeterminística que corra en tiempo a lo sumo polinomial en la longitud de sus entradas, de manera que produzca $R_1,\ldots,R_k$ nodeterminísticamente en tiempo polinomial, y luego verifique que la estructura expandida $\langle \mathcal{A}, R_1^{\mathcal{A}},\ldots,R_k^{\mathcal{A}}\rangle$ satisface ψ. Esto último se puede hacer en **L** (Teorema 11) y, en consecuencia, en **NP**.

$(\supseteq)$: Hemos indicado que el problema HP es completo para **NP** via reducciones describibles por fórmulas de primer orden, libres de cuantificadores y que incluyen la relación predefinida de sucesor. En el Ejemplo 40 vimos que HP es expresable en ∃SO. Por lo tanto, **NP** $\subseteq$ ∃SO (en principio sobre estructuras ordenadas pero realmente sobre todas las estructuras finitas por lo dicho antes del enunciado de este teorema). □

Corolario 43 (1) $\exists\text{SO} \neq \forall\text{SO}$ *si, y sólo si,* **NP** $\neq$ **coNP**. (2) $\exists\text{SO} = \forall\text{SO}$ *si, y sólo si, hamiltonicidad es expresable en* ∀SO. □

5.2. Operadores Inductivos.

Otra manera de incrementar el poder expresivo de PO es añadiendo a su sintáxis algún mecanismo de recursión.

Sea τ un vocabulario y R un símbolo relacional de aridad k con $R \notin \tau$. Sea ϕ una fórmula de primer orden con variables libres $x_1, \ldots, x_k$ y en la que R aparece, i.e., $\phi := \phi(x_1,\ldots,x_k,R) \in \text{PO}(\tau \cup \{R\})$.[****] Dada una τ–estructura $\mathcal{A}$, ϕ define un operador sobre $\mathcal{A}$ que transforma una relación de aridad k en una relación de aridad k de la siguiente manera:

$$\phi_{\mathcal{A}}(R) = \{(a_1,\ldots,a_k) \in A^k : \mathcal{A} \models \phi(a_1,\ldots,a_k,R^{\mathcal{A}})\}.$$

Considere la siguiente sucesión de conjuntos en A^k:

$$\phi^0_{\mathcal{A}} \;=\; \{(a_1,\ldots,a_k) \in A^k : \mathcal{A} \models \phi(a_1,\ldots,a_k,\emptyset)\}$$

y, para $i > 0$,

$$\phi^i_{\mathcal{A}} \;=\; \phi_{\mathcal{A}}(\phi^{i-1}_{\mathcal{A}}).$$

Es decir,

$$\begin{aligned}
\phi^1_{\mathcal{A}} &= \{(a_1,\ldots,a_k) \in A^k : \mathcal{A} \models \phi(a_1,\ldots,a_k,\phi^0_{\mathcal{A}})\},\\
&\vdots\\
\phi^{r+1}_{\mathcal{A}} &= \{(a_1,\ldots,a_k) \in A^k : \mathcal{A} \models \phi(a_1,\ldots,a_k,\phi^r_{\mathcal{A}})\},\\
&\vdots
\end{aligned}$$

Proposición 44 *Sea $\phi(\overline{x},R)$ una fórmula en forma normal disyuntiva. Si R aparece positiva en $\phi(\overline{x},R)$ (i.e., R aparece fuera del alcance de $\neg$), entonces $\phi^i_{\mathcal{A}} \subseteq \phi^{i+1}_{\mathcal{A}}$.*

Demostración: Hacer como ejercicio. (Ayuda: Demuestre primero el siguiente lema, por inducción en fórmulas. Lema: Si R aparece positiva en $\phi(\overline{x},R)$ y $P \subseteq Q \subseteq A^k$, entonces $\mathcal{A} \models \phi(\overline{a},P) \;\longrightarrow\; \phi(\overline{a},Q)$.) □

Por lo tanto, si R aparece positiva en $\phi(\overline{x},R)$, la sucesión $\{\phi^i_{\mathcal{A}}\}_{i\geq 0}$ alcanza un punto fijo (y en $\leq |A|^k$ pasos). Denotamos por $\phi^M_{\mathcal{A}}$ el *menor punto fijo* para $\phi(\overline{x},R)$ en $\mathcal{A}$, cuando R es positiva en ϕ; es decir, $\phi^M_{\mathcal{A}} = \phi^i_{\mathcal{A}}$, donde i es el menor entero positivo tal que $\phi^i_{\mathcal{A}} = \phi^{i+1}_{\mathcal{A}}$.

Si R no es positiva en ϕ, la sucesión $\{\phi^i_{\mathcal{A}}\}_{i\geq 0}$ no es necesariamente creciente y puede no alcanzar un punto fijo. (Pero si existe i tal que $\phi^i_{\mathcal{A}} = \phi^{i+1}_{\mathcal{A}}$, entonces $i \leq 2^{|A|^k}$).

Definimos el *punto fijo parcial* de $\phi(\overline{x},R)$ en $\mathcal{A}$, como el conjunto

$$\phi^P_{\mathcal{A}} = \begin{cases} \phi^i_{\mathcal{A}}\,, & \text{si } i \text{ es el menor entero tal que } \phi^i_{\mathcal{A}} = \phi^{i+1}_{\mathcal{A}} \\ \emptyset\,, & \text{si no existe } i \text{ tal que } \phi^i_{\mathcal{A}} = \phi^{i+1}_{\mathcal{A}} \end{cases}$$

Ejemplo 45 Considere el vocabulario adecuado para describir grafos $\tau = \{E\}$, donde E es un símbolo relacional binario, y sea R otro símbolo relacional binario. Sea

$$\phi(x,y,R) := (x = y) \vee \exists z(E(x,z) \wedge R(z,y)).$$

[****] Sin embargo, nuestra intención más adelante es tratar R como variable libre y, por lo tanto, para ser estrictos, $\phi(x_1,\ldots,x_k,R)$ es una fórmula de segundo orden.

R es positivo en ϕ y en consecuencia

$$\phi^0_{\mathcal{A}} \subseteq \phi^1_{\mathcal{A}} \subseteq \cdots \subseteq \phi^i_{\mathcal{A}} \subseteq \cdots \subseteq \phi^M_{\mathcal{A}} \subseteq A^2$$

donde

$$\begin{aligned}\phi^m_{\mathcal{A}} \quad = \quad & \{(v,w) : \text{existe un camino de } v \text{ a } w \\ & \text{en el grafo } \mathcal{A} = \langle A, E^{\mathcal{A}}\rangle \text{ de longitud} \leq m\},\end{aligned}$$

por lo que $\phi^M_{\mathcal{A}}$ es la clausura transitiva de $E^{\mathcal{A}}$. (El lector debe convencerse de esto escribiendo los conjuntos $\phi^m_{\mathcal{A}}$ para $m = 0, 1, 2, \ldots$.) □

Ejemplo 46 Sea τ y R como en el ejemplo anterior y considere

$$\psi(x, y, R) := [(x = y) \wedge \forall x \forall y \neg R(x, y)] \vee \exists z(E(x, z) \wedge R(z, y)).$$

R no es positivo en ψ y la $(m + 1)$-ésima iteración define la relación

$$\begin{aligned}\phi^m_{\mathcal{A}} \quad = \quad & \{(v,w) : \text{existe un camino de } v \text{ a } w \\ & \text{en el grafo } \mathcal{A} = \langle A, E^{\mathcal{A}}\rangle \text{ de longitud} = m\}.\end{aligned}$$

Esta puede o no alcanzar un punto fijo, dependiendo de $E^{\mathcal{A}}$. (Por ejemplo, se alcanza si $\mathcal{A}$ es un grafo completo, es decir, $E^{\mathcal{A}} = A^2$. El lector debe construir otros ejemplos donde no se alcanza y donde si se alcanza el punto fijo.) □

5.2.1. Las lógicas Menor Punto Fijo y Punto Fijo Parcial.

La lógica *menor punto fijo*, LFP[PO], es un conjunto de fórmulas tal que, para cualquier vocabulario τ:

(i) PO(τ) $\subseteq$ LFP[PO] (i.e., contiene a todas las τ-fórmulas de primer orden);

(ii) es cerrado bajo operaciones booleanas y cuantificación de primer orden: Si $\phi, \psi \in$ LFP[PO] entonces $\neg(\phi)$, $(\phi \vee \psi)$, $(\phi \wedge \psi)$, $\exists x(\phi)$, $\forall x(\phi)$ están en LFP[PO];

(iii) si R es un símbolo relacional de aridad k, $R \notin \tau$ y $\phi(x_1, \ldots, x_k, R)$ es una $(\tau \cup \{R\})$–fórmula en LFP[PO] donde (la variable)[*****] R aparece positiva, entonces

$$\mathsf{LFP}[\overline{x}, R : \phi](t_1, \ldots, t_k) \in \text{LFP[PO]}.$$

Análogamente, se define la lógica *punto fijo parcial*, PFP[PO], sintácticamente como un conjunto de fórmulas tal que, para cualquier vocabulario τ, se tiene (i) y (ii), con PFP en lugar de LFP, y

(iii′) Si R es un símbolo relacional de aridad k, $R \notin \tau$ y $\phi(x_1, \ldots, x_k, R)$ es una $(\tau \cup \{R\})$–fórmula en PFP[PO] (R puede aparecer negada en ϕ), entonces

$$\mathsf{PFP}[\overline{x}, R : \phi](t_1, \ldots, t_k) \in \text{PFP[PO]}.$$

[*****]Ver nota en la página 110.

Semántica: El significado de las fórmulas en LFP[PO] y PFP[PO] se define para los casos (i) y (ii) igual a como se hizo, inductivamente, para definir la semántica de PO y SO. Para (iii) y (iii'), sea $\mathcal{A}$ una τ–estructura y $a_1, \ldots, a_k \in A$, entonces

$\mathcal{A} \models \mathsf{LFP}[\overline{x}, R : \phi](a_1, \ldots, a_k)$ si y sólo si $(a_1, \ldots, a_k) \in \phi^M_{\mathcal{A}}$

y

$\mathcal{A} \models \mathsf{PFP}[\overline{x}, R : \phi](a_1, \ldots, a_k)$ si y sólo si $(a_1, \ldots, a_k) \in \phi^P_{\mathcal{A}}$.

Es obvio de las definiciones que PO $\subseteq$ LFP[PO] $\subseteq$ PFP[PO].

Ejemplo 47 La siguiente fórmula en LFP[PO],

$$\begin{aligned}&\Phi(u, v) := \\ &\quad \mathsf{LFP}[x, y, R : ((x = y) \vee \exists z(E(x, z) \wedge R(z, y))](u, v)\end{aligned}$$

expresa el que u y v están conectados por un camino (vease el Ejemplo 45 donde ya calculamos el punto fijo para la fórmula). □

El ejemplo anterior junto con el hecho de que TC no es expresable en PO, demuestran que PO $\subsetneq$ LFP[PO].

Otra vez aquí se presenta la misma deficiencia expresiva que encontramos en las extensiones de PO por cuantificadores generalizados: Las lógicas LFP[PO] y PFP[PO] verifican una ley de 0–1 (ver[5]) y, en consecuencia, no se puede definir con ellas el problema PAR (y por lo tanto no podrán capturar clases por encima o igual a **L**). La solución es, como en el caso de los cuantificadores generalizados, trabajar con estructuras ordenadas, o, equivalentemente, tomar LFP y PFP sobre PO$_s$ en vez de PO; es decir, considerar las lógicas LFP[PO$_s$] y PFP[PO$_s$].

Ejemplo 48 Sean τ_ϵ el vocabulario vacío y sea X un símbolo relacional unario. Considere la siguiente $(\tau_\epsilon \cup \{X\})$-fórmula en LFP[PO$_s$]:

$$\phi(x, X) := suc(0, x) \vee \exists y \exists z[X(y) \wedge suc(y, z) \wedge suc(z, x)]$$

Si $\mathcal{A} = \langle A \rangle$ es una τ_ϵ–estructura, entonces

$\mathcal{A} \models \exists w \neg \mathsf{LFP}[x, X : \phi(x, X)](w)$ si y sólo si

A es un conjunto ordenado de cardinalidad par. □

Proposición 49 PO$_s \subsetneq$ LFP[PO$_s$] $\subseteq$ PFP[PO$_s$]. □

Ejemplo 50 Veamos que PS, o el problema de determinar si un vértice t es accesible desde un vértice s en un sistema de caminos, es definible en LFP[PO]. Recuerde que el vocabulario para PS es $\tau_3 = \{R, C, D\}$ donde R es un símbolo relacional ternario, C y D constantes. Sea X una variable relacional unaria y $\mathcal{A} \in$ EST(τ_3). El lector puede verificar fácilmente que $\mathcal{A}$ está en PS si, y sólo si, $\mathcal{A}$ satisface la sentencia

$$\begin{aligned}&\exists w(\mathsf{LFP}[z, X : \\ &\quad (z = C \vee \exists x \exists y(X(x) \wedge X(y) \wedge R(x, y, z)))](w) \wedge w = D).\end{aligned}$$

(La idea es que en el conjunto X vamos guardando los vértices accesibles hasta alcanzar D.) □

En el siguiente ejemplo presentamos un nuevo problema computacional, que será de utilidad para realizar la descripción lógica de la clase **PESPACIO** por medio del operador PFP.

Ejemplo 51 En este ejemplo visualizamos las estructuras sobre el vocabulario $\tau_3 = \{R, C, D\}$ como sistemas de deducción (ver nota en la página 97). Sea $\mathcal{A} = \langle A, R^{\mathcal{A}}, C^{\mathcal{A}}, D^{\mathcal{A}}\rangle$ un sistema de deducción. Los elementos de $\mathcal{A}$ son *afirmaciones*. Decimos que una afirmación z en $\mathcal{A}$ es *parcialmente demostrable* (*p.d.*) si es $C^{\mathcal{A}}$, o se obtiene a partir de alguna afirmación x p.d. y otra afirmación y, no necesariamente p.d., por medio de una aplicación de la regla R a x y y (i.e. $(x, y, z) \in R^{\mathcal{A}}$). Dadas dos afirmaciones w y u, si para algún natural n se cumple que:

$$\begin{aligned} \forall b_1 \exists c_1 \quad : \quad & (b_1 = w \text{ o } b_1 \text{ no es p.d.}) \text{ y } (u, b_1, c_1) \in R \quad (2) \\ & \text{y, para } 1 \leq i < n, \\ & \forall b_{i+1} \exists c_{i+1} : (b_{i+1} = w \text{ o } b_{i+1} \text{ no es p.d.}) \text{ y} \quad (3) \\ & (c_i, b_{i+1}, c_{i+1}) \in R \text{ y } \ c_n = w, \end{aligned}$$

entonces decimos que w es demostrable a partir de u, por medio de una secuencia de afirmaciones parcialmente demostrables $c_1, c_2, \ldots, c_n$, con $c_n = w$. Definimos el problema Sistema Parcial de Deducción (PPS) como la siguiente clase de τ_3-estructuras:

$$\begin{aligned} \text{PPS} := \{\mathcal{A} \in \text{EST}(\tau_3) : & \mathcal{A} \text{ es un sistema de} \\ & \text{deducción, donde } D^{\mathcal{A}} \text{ es demostrable a partir de} \\ & C^{\mathcal{A}} \text{ por una sucesión de afirmaciones p.d.}\} \end{aligned}$$

No es difícil ver que PPS está en **PESPACIO**: en el paso i-ésimo de una demostración necesitamos tener almacenado en la cinta de trabajo la última afirmación c_i parcialmente demostrable, generar nodeterminísticamente una afirmación b_{i+1}, y guardar un c_{i+1} para el cual $(c_i, b_i, c_{i+1}) \in R^{\mathcal{A}}$. □

Para demostrar que PPS es log–completo para **PESPACIO** debemos uniformizar un poco el juego de Whex. Básicamente consideramos que el Jugador 2 es siempre quien inicia el juego. La correspondiente clase de estructuras la denotamos por WHEX′ y es fácil ver que ésta tiene iguales propiedades que WHEX (e.g. es un problema completo para **PESPACIO** via plc, etc.).

Teorema 52 WHEX′ $\leq_{\log}$ PPS.

Demostración: Dado un grafo $G = \langle V, E, u, w\rangle$, se define el sistema de deducción $\mathcal{A}_G = \langle A, R, s, t\rangle$, con $A = V$, $s = u$, $w = t$ y

$$\begin{aligned} R \quad = \quad & \{(a, b, c) : E(a, c) \wedge E(a, b) \wedge b \neq c\} \\ & \cup \{(a, w, w) : E(a, w)\} \end{aligned}$$

Supongamos que $G \in$WHEX. Entonces, cualquiera sea el vértice b_1 con que Jugador 2 comienza el juego de Whex sobre G, el Jugador 1 puede responder con c_1 tal que $E(u, b_1)$ y $E(u, c_1)$ se verifiquen (y de tal manera que c_1 esté sobre un camino a w). Sucesivamente, en el i–ésimo movimiento, cualquiera sea el vértice b_i que Jugador 2 selecciona, Jugador 1 puede responder con un c_i tal que $E(c_{i-1}, b_i)$ y $E(c_{i-1}, c_i)$ se verifican (y c_i está sobre un camino a w), y así sucesivamente, hasta que Jugador 1 alcanza un vértice c_n tal que $E(c_n, w)$ es cierto. Luego la sucesión $c_1, c_2, \ldots, c_n, w$, constituyen un conjunto de afirmaciones parcialmente demostrables en $\mathcal{A}_G$, que satisfacen condiciones (2) y (3). En consecuencia, $\mathcal{A}_G \in$PPS.

Recíprocamente, si suponemos que $\mathcal{A}_G \in$PPS, entonces existe una demostración de t a partir de s por una sucesión de afirmaciones p.d., que verifican condiciones (2) y (3). Estas condiciones describen una estrategia ganadora para el Jugador 1 en la versión del juego de Whex sobre G donde el Jugador 2 realiza el primer movimiento. □

Argumentamos a continuación porque PPS es completo via proyecciones libres de cuantificadores. Primero consideramos PPS como un cuantificador generalizado que agregamos a PO_s para obtener la lógica $\text{PPS}^*[\text{PO}_s]$, y veamos que el problema DTC es expresable en $\text{posPPS}^*[\text{PO}_s]$: dado un grafo G, con vértices distinguidos s y t, un camino $\langle s, c_1, c_2, \ldots, c_n, t\rangle$ de s a t puede visualizarse como una demostración de t a partir de s, por la siguiente sucesión de aplicaciones de la regla R:

$$(s, t, c_1), (c_1, t, c_2), \ldots, (c_i, t, c_{i+1}), \ldots, (c_n, t, t).$$

Seguidamente, para demostrar que $\text{posPPS}^*[\text{PO}_s] = \text{posPPS}^1[\text{PO}_s]$ procedemos de manera similar a la demostración de Stewart de la forma normal para $\text{PS}^*[\text{PO}_s]$ (Teorema 4.2 de[21]). Por ejemplo, para eliminar el cuantificador existencial en la sentencia $\exists w\text{PPS}[x, y, z : \psi(x, y, z, w)](C, D)$, consideramos, dada una estructura $\mathcal{A}$ de cardinalidad n, la unión disjunta de n sistemas de deducción descritos por la fórmula $\psi^{\mathcal{A}}(x, y, z, i)$, para $i = 0, 1, \ldots, n-1$, con una afirmación inicial y una afirmación final común a todos los sistemas. Tenemos así un resultado análogo al Teorema 39.

Teorema 53 $\text{PPS}^*[\text{PO}_s] = \text{posPPS}^*[\text{PO}_s] = \text{posPPS}^1[\text{PO}_s]$ = **PESPACIO** *y la forma normal para* $\text{posPPS}^*[\text{PO}_s]$ *es proyectiva. En consecuencia, el problema* PPS *es completo para* **PESPACIO** *via plc (junto con la relación sucesor).* □

Finalmente, es fácil ver que PPS es definible en PFP[PO] (sin necesidad de orden) por la sentencia

$$\begin{aligned} \exists w(\text{PFP}[z, X : & \\ & z = C \vee \exists x \forall y (X(x) \wedge (\neg X(y) \vee y = D) \\ & \wedge R(x, y, z))](w) \wedge w = D) \end{aligned}$$

Esencialmente, el conjunto X guarda las afirmaciones parcialmente demostrables, comenzando con C y hasta que se alcance D.

Hemos completado todos los ingredientes necesarios para demostrar el siguiente importante resultado.

Teorema 54 *Sea τ un vocabulario y K un problema sobre τ*

(Immerman,Vardi 1982) *K es definible en* LFP[PO$_s$](τ) *si, y sólo si, $K \in$* **P***.*

(Abiteboul-Vianu 1989) *K es definible en* PFP[PO$_s$](τ) *si, y sólo si, $K \in$* **PESPACIO***.*

En otras palabras: sobre estructuras ordenadas
LFP[PO] = **P** *y* PFP[PO] = **PESPACIO***.*

Demostración: Veamos primero que si $K = \text{MOD}(\Phi)$, donde Φ es una sentencia en LFP[PO$_s(\tau)$], entonces $K \in$ **P**. Ya hemos visto que si $\Phi \in \text{PO}_s(\tau)$ entonces el problema $cod_\tau(\text{MOD}(\Phi))$ es decidible en **L**. Inductivamente, sea $\phi(x_1, \ldots, x_k, R) \in \text{PO}_s(\tau \cup \{R\})$ con R una relación de aridad k, que no está en τ y aparece positiva en ϕ, y considere $\Phi := \text{LFP}[\overline{x}, R : \phi](t_1, \ldots, t_k)$. Para cualquier τ–estructura $\mathcal{A}$, la correspondiente sucesión inductiva

$$\phi^i_{\mathcal{A}} = \{(a_1, \ldots, a_k) \in A^k : \mathcal{A} \models \phi(\overline{a}, \phi^{i-1}_{\mathcal{A}})\}$$

está acotada por A^k y cada conjunto de la sucesión sólo añade nuevas k–tuplas al conjunto construido en el paso anterior, por lo que sólo pueden haber, a lo sumo, $|\mathcal{A}|^k$ pasos hasta alcanzar el punto fijo. Por lo tanto, verificar si $\langle \mathcal{A}, \overline{a} \rangle \models \Phi$, o, equivalentemente, si $\overline{a} \in \phi^M_{\mathcal{A}}$ puede hacerse determinísticamente y en tiempo polinomial.

Ahora probemos lo mismo para el caso de PFP. Sea $\phi(x_1, \ldots, x_k, R)$ como antes, salvo que no exigiremos que R aparezca positiva en ϕ, y sea $\Phi := \text{PFP}[\overline{x}, R : \phi](t_1, \ldots, t_k)$. Dada una τ–estructura $\mathcal{A}$ con $n = |\mathcal{A}|$, la correspondiente sucesión inductiva $\phi^i_{\mathcal{A}}$, o bien verifica que $\phi^{2^{n^k}-1}_{\mathcal{A}} = \phi^{2^{n^k}}_{\mathcal{A}}$ (y en ese caso el punto fijo parcial $\phi^P_{\mathcal{A}}$ es $\phi^{2^{n^k}-1}_{\mathcal{A}}$), o bien que $\phi^P_{\mathcal{A}} = \emptyset$. Sea M una máquina determinística de Turing que al leer $cod_\tau(\mathcal{A})$ comienza por establecer un contador hasta el número $2^{n^k} - 1$ (es decir, escribe la palabra $1 \cdots 1$ de longitud n^k en una cinta de trabajo). Luego M procede a construir los sucesivos $\phi^i_{\mathcal{A}}$, utilizando el contador para asegurarse que se construyen a lo sumo $2^{n^k} - 1$ de estos conjuntos. Para construir $\phi^i_{\mathcal{A}}$ se coloca en una segunda cinta de trabajo una palabra de longitud n^k que es el código de una relación S de aridad k y, en una tercera cinta, se construye

$$\phi^i_{\mathcal{A}} = \{(a_1, \ldots, a_k) \in A^k : \mathcal{A} \models \phi(\overline{a}, S)\}.$$

Luego M analiza si $\phi^i_{\mathcal{A}} = S$. Si la respuesta es afirmativa entonces este conjunto es el punto fijo parcial, M entonces verifica si $\overline{t}^{\mathcal{A}}$ está allí y, en caso afirmativo, acepta, de lo contrario rechaza. Si $\phi^i_{\mathcal{A}}$ no es igual a S, se borra S de la cinta 2, se coloca $\phi^i_{\mathcal{A}}$ en su lugar y se repite la rutina. Note que esta rutina corre en $\leq n^k$ pasos y se hacen a lo sumo 2^{n^k} invocaciones de la misma. Si el contador se hace negativo, ya que se producen 2^{n^k} invocaciones de la rutina sin construirse el punto fijo, entonces este es vacío y M rechaza. El espacio utilizado es a lo sumo n^k.

Veamos ahora que todos los problemas decidibles en **P** (respectivamente **PESPACIO**) son definibles en LFP[PO$_s$] (respectivamente PFP[PO$_s$]).

PS es completo para **P** via proyecciones de primer orden con la relación de orden predefinida suc y las constantes 0 y max. En el Ejemplo 50 se demostró que PS es definible en LFP[PO] y, por lo tanto, en LFP[PO$_s$]. Concluimos entonces que **P** $\subseteq$ LFP[PO$_s$], al ser LFP[PO$_s$] obviamente cerrada bajo $\leq_{PO_s}$.

Análogamente, PPS es completo para **PESPACIO** via proyecciones de primer orden con la relación suc y las constantes 0 y max. Además PPS es definible en PFP[PO] y, por lo tanto, en PFP[PO$_s$]. Concluimos entonces que **PESPACIO** $\subseteq$ PFP[PO$_s$]. □

El estudio que hemos hecho del problema PPS, junto con lo que sabemos del problema PS, nos permite dar una caracterización del problema **P** versus **PESPACIO** en términos del comportamiento de sistemas de deducción mecánicos: **P** = **PESPACIO** si y sólo si PPS es primer orden reducible a PS. Informalmente, esto nos dice que para demostrar que la clase **PESPACIO** es igual a la clase **P**, es suficiente demostrar que cualquier sistema de deducción que emplea ayuda auxiliar (en la forma de lemas provistos por el usuario) para realizar sus demostraciones, puede ser simulado por un sistema de deducción donde todas las demostraciones son secuencias de afirmaciones mecánicamente obtenidas sin ayuda externa.

Otras consecuencias más obvias de la teoría que hemos desarrollado, aunque no menos interesante, se resumen en el siguiente corolario.

Corolario 55 *Sobre estructuras finitas (ordenadas), se tiene:*

(i) LFP[PO$_s$] = PFP[PO$_s$] *si y sólo si* **P** = **PESPACIO***;*

(ii) LFP[PO$_s$] = $\exists$SO *si y sólo si* **P** = **NP***.* □

5.3. Otras Lógicas

Existe una manera de restringir el poder expresivo de SO para sólo capturar los problemas decidibles en tiempo polinomial determinísticamente y en espacio logaritmico nodeterminísticamente. Estos resultados se deben a Erich Grädel[10]. El punto de partida es el siguiente hecho: toda fórmula de segundo orden es equivalente a una fórmula en forma prenexa normal conjuntiva, similar al caso de fórmulas de primer orden, donde todos los cuantificadores de segundo orden aparecen primero, seguidos de los cuantificadores de primer orden y luego la fórmula libre de cuantificadores en forma normal conjuntiva (ver[5]). En particular, consideremos sólamente las fórmulas de segundo orden que presentan la siguiente forma prenexa:

$$\Phi := Q_1 R_1 \ldots Q_r R_r \forall x_1 \ldots \forall x_s \theta$$

es decir, la cuantificación de primer orden es sólo universal y θ es una fórmula libre de cuantificadores en forma normal conjuntiva. Una tal fórmula Φ se dice que es de *tipo Horn* si y sólo si cada cláusula de θ tiene a lo sumo una ocurrencia positiva de alguna de las variables de segundo orden cuantificadas R_i. Por otra parte Φ es de *tipo Krom* si y sólo si cada cláusula de θ tiene a lo sumo dos ocurrencias de una misma variable de segundo orden cuantificada R_i. Sean SO–Horn y SO–Krom los conjuntos de fórmulas de segundo orden que son de tipo Horn y Krom respectivamente.

Teorema 56 (E. Grädel[10]) *Sobre estructuras finitas ordenadas,*

(i) SO–Horn = **P**.

(ii) SO-Krom = **NL**. □

Ahora bien, para hacer una demostración de este teorema empleando nuestro paradigma de captura (sección 3.4) resulta más fácil trabajar con los complementos respectivos de las clases **P** y **NL** que con las propias clases[******]. Por ejemplo, considere el complemento del problema PS que denotamos por $\overline{\text{PS}}$. Este es completo para **coP** y es fácilmente expresable en SO-Horn: lo hemos hecho en el Ejemplo 41. También es fácil ver que SO-Horn es cerrada bajo reducciones de primer orden. En consecuencia, **coP** $\subseteq$ SO-Horn, y como **P** $=$ **coP** obtenemos que **P** $\subseteq$ SO-Horn. La contención contraria se deja como ejercicio.

Para demostrar que SO-Krom $=$ **NL** proceda usted de manera análoga trabajando con el complemento del problema TC y asumiendo la igualdad **NL** $=$ **coNL**, la cual es cierta pero no es nada trivial de demostrar; de hecho esta igualdad era una conjetura de larga edad hasta que fue resuelta en 1988 por Neil Immerman[13] e independientemente por Róbert Szelepcsényi[22], y ambos investigadores recibieron el premio Gödel en su edición de 1995 por eso.

6. Notas Finales

He presentado aqui una manera uniforme de visualizar los resultados más notables sobre el puente existente entre diversas clases de complejidad computacional y diferentes lógicas, y creo que la metodología expuesta, además de cumplir un fin didáctico al simplificar la exposición de estos laboriosos resultados, puede contribuir a descubrir las razones formales por las cuales estas capturas ocurren. La idea del paradigma propuesto para establecer el puente entre una lógica y la clase computacional que describe (sección 3.4) no es originalmente mia, más bien es una observación extraída del folclore pues se encuentra con frecuencia empleada por otros autores para establecer casos particulares de esta relación entre la lógica y la computación. Sin embargo, creo es esta la primera vez que se hace un uso sistemático de la misma para explicar todas las descripciones lógicas de todas las clases de complejidad posibles.

El método de los cuantificadores generalizados para producir extensiones de la lógica de primer orden de mayor capacidad expresiva, se originó en el artículo de Lindström[16]; pero su uso en la computación (de la manera que explico en la sección 4) la inició Immerman en[12]. Este método es útil para demostrar que un problema, conocido completo para una clase via log–reducciones, es completo via reducciones describibles en primer orden, y su formulación explícita como pasos (1) a (5) en la página 106 se hace por primera vez en este trabajo. En particular, el teorema clave (Teorema 33) es un resultado hasta ahora sólo publicado en mi tesis doctoral[2]. El lema fundamental en que se basa (Lema 32), que establece el puente entre la lógica y la computación, aparece enunciado de una manera diferente en la obra de Ebbinghaus y Flum[5], y aunque allí no se le atribuye a algún autor, su origen se puede trazar a los varios trabajos de Immerman, en particular[12], por encontrarse implícito en estos. Sin embargo la demostración que doy de este lema (sección 7 a continuación) es diferente a la de[5] y sigue más bien el razonamiento expuesto en la demostración de una proposición similar que aparece en el artículo[19] de Iain Stewart.

Como hemos indicado ya, el Teorema de Fagin (Teorema 5.1.3) dio origen a esta teoría de la descripción de la complejidad algorítmica mediante lenguajes formales, pero además, junto con otros resultados del mismo Fagin, todos publicados por primera vez en su tesis doctoral, fijó la atención de varios lógicos en el estudio más general de las estructuras finitas, iniciándose así toda una nueva área de investigación en las matemáticas denominada actualmente como la *Teoría de Modelos Finitos*, con un gran número de practicantes, físicamente esparcidos por el mundo pero electrónicamente reunidos en el portal `http://www-mgi.informatik.rwth-aachen.de/FMT/` donde el lector puede hallar una larga lista de referencias sobre este tema, una lista de problemas abiertos (aunque no muy actualizada) y otras cosas. Para quien desee iniciarse en el estudio de los temas pertinentes a la Teoría de Modelos Finitos recomiendo el artículo expositorio de Ronald Fagin[8], o la monografía publicada por el centro DIMACS [15], o los libros de texto[5,14].

Concluiré con una anécdota personal: En abril de 1998 conocí al matemático checo Pavel Pudlák en un congreso efectuado en la Universidad de Campinas, Brasil. Allí, en una de nuestras amistosas conversaciones sobre la teoría descriptiva de la Complejidad Computacional, Pavel me reveló que en su tesis de maestría presentada en la Universidad de Charles, Checoslovaquia, en 1975, demostró un teorema análogo al Teorema de Fagin, pero al conocer que "su" resultado ya había sido demostrado y recientemente publicado por Ronald Fagin decidió "engavetar" su tesis de maestría y el único reporte internacional que existe sobre este trabajo es un pequeño resumen (ver[18]), que

[******]Debemos indicar que esto mismo hace Grädel en su demostración

Pavel tuvo la amabilidad de enviarme por correo, donde sólo se enuncian los teoremas conducentes a una demostración por el autor de "una conexión entre los lenguajes nodeterminísticamente reconocibles en tiempo polinomial y clases *proyectivamente definibles* de estructuras finitas".

Esta anécdota significa para mí una comprobación más de que ciertas perlas de la matemática no pueden atribuirse a una persona sino a un momento del desarrollo de la historia en que se llega a acumular una gran cantidad de información, gracias a un gran esfuerzo colectivo, que conduce de manera natural al descubrimiento de tal gema. Respecto al Teorema de Fagin, Fagin mismo nos revela en[7] que este teorema había sido en parte demostrado por James Bennet, alrededor del mismo año, en su tesis doctoral que jamás publicó, y que Jones and Selman publicaron en 1972 un resultado parecido aunque utilizando técnicas distintas. Definitivamente el Teorema de Fagin es una gema de los setentas y Fagin logró capturar, antes que otros, ese gran momento.

7. Apéndice: Demostración del Lema 32

Supóngase que $\Omega_1 \leq_{DTC^1[PO_s]} \Omega_2$ via la (τ,σ)–traslación de aridad k, $\Sigma = \{\phi_1, \ldots, \phi_r, \psi_1, \ldots, \psi_c\} \subseteq$ DTC^1[PO$_s(\tau)$], y tal que para cada $\mathcal{A} \in$ EST(τ) su correspondiente σ-estructura según Σ es $\mathcal{A}_\Sigma = \langle A^k, R_1^{\mathcal{A}_\Sigma}, \ldots, R_r^{\mathcal{A}_\Sigma}, C_1^{\mathcal{A}_\Sigma}, \ldots, C_c^{\mathcal{A}_\Sigma}\rangle$, con cada relación $R_i^{\mathcal{A}_\Sigma}$ y cada constante $C_j^{\mathcal{A}_\Sigma}$ definida por ϕ_i y ψ_j respectivamente, de acuerdo con la Definición 14. Como DTC es un problema en **L** y PO$_s \subseteq$ **L**, verificar que $\mathcal{A} \models \phi_i$ o $\mathcal{A} \models \psi_j$ con $\phi_i, \psi_j \in$ DTC^1[PO$_s(\tau)$] se puede hacer utilizando espacio logarítmico y, en consecuencia, la reducción de $cod_\tau(\Omega_1)$ en $cod_\sigma(\Omega_2)$ que a cada $cod_\tau(\mathcal{A})$, con $\mathcal{A} \in$ EST(τ), asocia la palabra $cod_\sigma(\mathcal{A}_\Sigma)$ es realizable por una (τ,σ)–traductora de aridad k que emplea a lo sumo espacio de trabajo logarítmico.

En la otra dirección, sea M una $\log$–(τ,σ)–traductora de aridad k tal que para toda palabra $w \in \{0,1\}^*$:

1. si $w = cod_\tau(\mathcal{A})$ para algún $\mathcal{A} \in$ EST(τ) de cardinalidad n, entonces $M(w) = cod_\sigma(\mathcal{B})$ para algún $\mathcal{B} \in$ EST(σ) de cardinalidad n^k; y además
2. $w \in cod_\tau(\Omega_1)$ si y sólo si $M(w) \in cod_\sigma(\Omega_2)$.

Debemos hallar un conjunto de fórmulas $\Sigma = \{\phi_1, \ldots, \phi_r, \psi_1, \ldots, \psi_c\}$ en DTC^1[PO$_s(\tau)$] que definen una (τ,σ)–traslación de aridad k de EST(τ) en EST(σ) tal que para cada $\mathcal{A} \in$ EST(τ), $M(cod_\tau(\mathcal{A})) = cod_\sigma(\mathcal{A}_\Sigma)$.

Recordemos de la sección 2.1 que una configuración instantánea CI de M se puede codificar por una tupla de longitud finita y, por lo tanto, esta puede definirse en PO por medio de un número finito de variables que toman valores entre 0 y $|A|-1$, donde $\mathcal{A}$ es la τ–estructura cuya codificación damos como entrada a M. Téngase en cuenta que no se incluye el contenido de la cinta de salida de M en sus configuraciones instantáneas.

Lo primero que debemos hacer es definir en DTC*[PO$_s$] el predicado $PROX(CI, CI')$ cuyo significado es que la configuración instantánea CI' es consecuencia inmediata de la configuración instantánea CI por la aplicación de las reglas de transición de M. En los párrafos que siguen espero dar suficientes detalles como para convencer al lector de que esto puede hacerse.

Para fijar ideas pongamos $CI := \langle q, i, u_1, h_1, \ldots, u_k, h_k\rangle$ y $CI' := \langle q', i', u_1', h_1', \ldots, u_k', h_k'\rangle$, donde q es el estado actual, i la posición del cabezal en la cinta de la entrada, u_j es la palabra escrita en la j-ésima cinta de trabajo y h_j es la posición del cabezal en la j-ésima cinta de trabajo; los términos en CI' tienen interpretaciones similares. Informalmente $PROX(CI, CI')$ es una disyunción sobre todas las posibles transiciones que M puede realizar. Estas son un número finito y no dependen de las entradas que reciba M. Cada transición es una regla de la forma $(q, b, \overline{w}) \to (q', D, \overline{w}', \overline{d})$ que dice:

> "si M está en estado q y el cabezal de la cinta de entrada lee el bit b y los k cabezales de las k cintas de trabajo leen los bits w_1,...,w_k respectivamente (hemos abreviado $w_1, \ldots, w_k$ como $\overline{w}$), entonces M pasa al estado q', mueve el cabezal de la cinta de entrada en la dirección D, los k cabezales de las k cintas de trabajo escriben los bits w_1',..., w_k', respectivamente ($(w_1', \ldots, w_k') = \overline{w}'$) y mueve los cabezales de las K cintas de trabajo en las direcciones $(d_1, \ldots, d_k) = \overline{d}$"

Los estados de M son un número finito y fijo; por lo tanto se pueden codificar con un número finito de términos. Lo mismo ocurre con las tres posibles direcciones en que se pueden mover los cabezales: izquierda, derecha y parado. Todos estos elementos son expresables en primer orden. Usaremos q, q', q_i y similares para denotar estados y para las direcciones de movimientos usaremos izq (izquierda), der (derecha) y $para$ (parado).

Pongamos entonces

$$PROX(CI,CI') := \bigvee_{(q,b,\overline{w})\to(q',D,\overline{w}',\overline{d})} CI \xrightarrow{M} CI'$$

donde $(q, b, \overline{w}) \to (q', D, \overline{w}', \overline{d})$ varia sobre todas las transiciones posibles de M y $CI \xrightarrow{M} CI'$ es una abreviación de la fórmula que afirma: "CI' se obtuvo a partir de CI por medio de una de las reglas de transición de M". Veamos cuales son los ingredientes para describir esta frase en el lenguaje formal DTC*[PO$_s$].

Debemos tener la siguiente cláusula:

$$\begin{aligned}(D = izq &\longrightarrow i' = i-1)\\ \vee(D = der &\longrightarrow i' = i+1)\\ \vee(D = para &\longrightarrow i' = i)\end{aligned}$$

la cual describe la posición del cabezal de la cinta de entrada en el instante CI' respecto de la posición que indicaba el instante CI y de acuerdo con el movimiento D indicado por la transición en uso.

Necesitamos una cláusula similar para describir la nueva posición del cabezal en la j-ésima cinta de trabajo, y esto para cada $j = 1, 2, \ldots, k$:

$$\begin{aligned}(d_j = izq \ &\longrightarrow \ h'_j = h_j - 1)\\ \vee(d_j = der \ &\longrightarrow \ h'_j = h_j + 1)\\ \vee(d_j = para \ &\longrightarrow \ h'_j = h_j)\end{aligned}$$

Ahora falta describir como debe ser la palabra u'_i ($1 \leq i \leq k$) en la configuración CI' respecto de la palabra u_i en la configuración CI. Esta es así: todos los bits en u'_i, salvo el bit en la posición h_i, deben ser iguales a los bits en las posiciones correspondientes de u_i; M sólo escribe w_i en la posición h_i de u_i y obtenemos u'_i. Para esto necesitamos un predicado $ON(\overline{w}, j)$ que declare lo siguiente: "el j-ésimo bit de la palabra $\overline{w}$ es 1". Asumamos por el momento que podemos describir este predicado en DTC*[PO$_s$], entonces la sentencia que nos falta es:

$$\begin{aligned}\bigwedge_{i=1}^{k}[\forall j(j = h_i \vee (ON(u_i, j) \ &\longleftrightarrow \ ON(u'_i, j)))\\ \wedge(ON(u_i, h_i) \ &\longleftrightarrow \ w_i = 1)\\ \wedge(ON(u'_i, h'_i) \ &\longleftrightarrow \ w'_i = 1)]\end{aligned}$$

Veamos ahora que $ON(\overline{w}, j)$ es expresable en DTC*[PO$_s$]. Voy a hacer un poco de trampa que justifico como se suelen justificar todas las trampas en matemáticas: "Sin pérdida de generalidad podemos asumir que ...".

Recuerde que $\overline{w}$ es una palabra sobre el alfabeto $\{0, 1\}$ y, como tal, podemos considerarla como la representación en binario de algún natural, que denotaremos por w. Entonces, visto como naturales, $ON(w, j)$ es cierto si, y sólo si, el cociente de dividir w por 2^j es impar.

Para expresar esta última condición equivalente a $ON(w, j)$, necesitamos primero expresar en DTC*[PO$_s$] el predicado $SUMA(x, y, z) :=$ "$x + y = z$", que define la suma de enteros positivos. Considere la siguiente fórmula sobre pares (x, y) y (x', y'),

$$\theta(x, y, x', y') := suc(x', x) \wedge suc(y, y').$$

Observe que $\theta(x, y, x', y')$ es cierta si, y sólo si, $x' = x - 1$ y $y' = y + 1$. Luego

$$SUMA(x, y, z) := \text{DTC}[(x, y), (x', y') : \theta](x, y, 0, z).$$

Considere ahora

$$\begin{aligned}&\beta(u, v, u', v') :=\\ &\quad \exists z(SUMA(u', u', z) \wedge (u = z \vee suc(z, u))) \wedge suc(v', v).\end{aligned}$$

Observe que hay un β–arco de (u, v) a (u', v') si $u' = u/2$ o $u' = (u - 1)/2$, y $v' = v - 1$. Sea $IMPAR(z) := \exists x \exists y(SUMA(x, x, y) \wedge suc(y, z))$. Se tiene entonces que

$$ON(w, j) := \exists z(IMPAR(z) \wedge \text{DTC}[u, v, u', v' : \beta](w, j, z, 0)).$$

Estos son los ingredientes para describir $PROX(CI, CI')$ en DTC*[PO$_s$].

Adicionalmente, necesitaremos los predicados $ESCRIB0(CI)$, $ESCRIB1(CI)$ y $NOESCRIB(CI)$ que dicen que el único movimiento de M a partir de la configuración instantánea CI es escribir un 0, escribir un 1 y no escribir en la cinta de salida, respectivamente. Estos predicados pueden expresarse en PO(τ) y se dejan como ejercicio.

Ahora bien, la fórmula $\phi_1(\overline{x}_1)$ que requerimos debe definir la relación R_1 en la estructura $\mathcal{A}_\Sigma$, cuya codificación $cod_\sigma(\mathcal{A}_\Sigma)$ M produce como salida, al recibir por entrada $cod_\tau(\mathcal{A})$. Es decir

$$R_1^{\mathcal{A}_\Sigma} = \{\overline{u} \in A^{kn_1} : \langle \mathcal{A}, \overline{u}\rangle \models \phi_1(\overline{x}_1)\}$$

Esta relación corresponde a una palabra de longitud n^{kn_1} que M genera, bit a bit, de la siguiente manera: M al leer $cod_\tau(\mathcal{A})$ transita un camino de configuraciones, que comienza en la inicial CI_0, hasta una cierta configuración instantánea J tal que, al alcanzarse, ocasiona la escritura de un 1 en un lugar de la cinta de salida, correspondiente a la posición de cierta tupla $\overline{u}$ en A^{kn_1} (en un orden lexicográfico) y que constituirá uno de los elementos de $R_1^{\mathcal{A}_\Sigma}$. Todo esto nos indica que la fórmula $\phi_1(\overline{x}_1)$ debe tener la siguiente forma

$$\begin{aligned}\phi_1(\overline{x}_1) := \exists J \ \text{DTC}[&(CI, z, w, \overline{y}), (CI', z', w', \overline{y}') :\\ &\theta_1]((CI_0, 0, 0, \overline{0}), (J, max, max, \overline{x}_1)),\end{aligned}$$

donde

$$\begin{aligned}\theta_1 := [&PROX(CI, CI') \wedge ESCRIB0(CI)\\ &\wedge z' = 0 \wedge w = 0 \wedge w' = max \wedge \overline{y}' = \overline{y}]\\ \vee[&PROX(CI, CI') \wedge ESCRIB1(CI)\\ &\wedge z' = max \wedge w = 0 \wedge w' = max \wedge \overline{y}' = \overline{y}]\\ \vee[&PROX(CI, CI') \wedge NOESCRIB(CI)\\ &\wedge z' = z \wedge w = 0 \wedge w' = 0 \wedge \overline{y}' = \overline{y}]\\ \vee[&CI = CI' \wedge z' = z \wedge w = max \wedge w' = 0\\ &\wedge \overline{y} \neq \overline{max} \wedge \overline{y}' = \overline{y} + 1]\end{aligned}$$

siendo

CI_0 una tupla de símbolos constantes que codifica la configuración instantánea inicial de M;
CI, CI' y J tuplas de variables que representan configuraciones instantáneas;
$\overline{y}$ y $\overline{y}'$ tuplas de kn_1 variables cada una, que se utilizan para contar el número de símbolos escritos en la cinta de salida. La expresión $\overline{y}' = \overline{y} + 1$ es una abreviación de la fórmula (de primer orden) que dice que el número representado por la kn_1–tupla $\overline{y}'$ es mayor en 1 que el representado por $\overline{y}$.

Sea $\mathcal{A} \in$ EST(τ) y $\overline{u} \in A^{kn_1}$ tal que

$$\begin{aligned}\langle \mathcal{A}, \overline{u}\rangle \models \exists J \ \text{DTC}[&(CI, z, w, \overline{y}), (CI', z', w', \overline{y}') :\\ &\theta_1]((CI_0, 0, 0, \overline{0}), (J, max, max, \overline{x}_1)).\end{aligned}$$

Entonces existe una configuración instantánea, representada por J, tal que M al tener por entrada $cod_\tau(\mathcal{A})$ alcanza esa configuración y, al hacerlo, escribe el $\overline{u}$–ésimo símbolo, el cual es un 1; por lo tanto, si $\overline{u} = (\overline{u}_1, \ldots, \overline{u}_{n_1})$, donde cada u_j es una k–tupla, entonces $R_1^{\mathcal{A}_\Sigma}(\overline{u}_1, \ldots, \overline{u}_{n_1})$ es cierto, donde $cod_\sigma(\mathcal{A}_\Sigma) = M(cod_\tau(\mathcal{A}))$. Recíprocamente, si $R_1^{\mathcal{A}_\Sigma}(\overline{u}_1, \ldots, \overline{u}_{n_1})$ es cierto, entonces el $\overline{u}$–ésimo símbolo escrito en la cinta de salida es un 1 y, por lo tanto, existe alguna configuración instantánea que es alcanzada por M al tener por entrada $cod_\tau(\mathcal{A})$ y, en ese momento, escribe un 1 como el $\overline{u}$–ésimo símbolo en la cinta de salida; en consecuencia

$$\langle \mathcal{A}, \overline{u}\rangle \models \exists J\ \mathsf{DTC}[(CI, z, w, \overline{y}), (CI', z', w', \overline{y}') : \theta_1]((CI_0, 0, 0, \overline{0}), (J, max, max, \overline{x}_1)).$$

Así, $\langle \mathcal{A}, \overline{u}\rangle \models \phi_1(\overline{x}_1)$ si y sólo si $R_1^{\mathcal{A}_\Sigma}(\overline{u}_1, \ldots, \overline{u}_{n_1})$ es cierto.

La fórmula ϕ_2 es muy similar a ϕ_1 salvo que debemos tomar en cuenta que los 0's y 1's que M escribe en su cinta de salida, correspondiente a la codificación de $R_2^{\mathcal{A}_\Sigma}$, deben aparecer después de la palabra que codifica a $R_1^{\mathcal{A}_\Sigma}$; en consecuencia debemos establecer los contadores de kn_1–tuplas y kn_2–tuplas necesarios.

Sea

$$\phi_2(\overline{x}_2) := \exists J\ \mathsf{DTC}[(CI, z, w, \overline{y}_1, \overline{y}_2), (CI', z', w', \overline{y}_1', \overline{y}_2') : \theta_2]((CI_0, 0, 0, \overline{0}, \overline{0}), (J, max, max, \overline{max}, \overline{x}_2)),$$

donde

$$\begin{aligned}\theta_2 := {} & [PROX(CI, CI') \wedge ESCRIB0(CI) \\ & \wedge z' = 0 \wedge w = 0 \wedge w' = max \\ & \wedge\ \overline{y}_1' = \overline{y}_1 \wedge \overline{y}_2' = \overline{y}_2] \\ & \vee [PROX(CI, CI') \wedge ESCRIB1(CI) \\ & \wedge z' = max \wedge w = 0 \wedge w' = max \\ & \wedge\ \overline{y}_1' = \overline{y}_1 \wedge \overline{y}_2' = \overline{y}_2] \\ & \vee [PROX(CI, CI') \wedge NOESCRIB(CI) \\ & \wedge z' = z \wedge w = 0 \wedge w' = 0 \\ & \wedge\ \overline{y}_1' = \overline{y}_1 \wedge \overline{y}_2' = \overline{y}_2] \\ & \vee [CI = CI' \wedge z' = z \wedge w = max \wedge w' = 0 \\ & \wedge \overline{y}_1 \neq \overline{max} \wedge \overline{y}_1' = \overline{y}_1 + 1 \\ & \wedge\ \overline{y}_2 = \overline{0} \wedge \overline{y}_2' = \overline{0}] \\ & \vee [CI = CI' \wedge z' = z \wedge w = max \wedge w' = 0 \\ & \wedge \overline{y}_1 = \overline{max} \wedge \overline{y}_1' = \overline{max} \\ & \wedge\ \overline{y}_2 \neq \overline{max} \wedge \overline{y}_2' = \overline{y}_2 + 1]\end{aligned}$$

y

CI_0 es una tupla de símbolos constantes que codifica la configuración instantánea inicial de M;
CI, CI' y J son tuplas de variables que representan configuraciones instantáneas;
$\overline{y}_1$ y $\overline{y}_1'$ son tuplas de kn_1 variables, $\overline{y}_2$ y $\overline{y}_2'$ son tuplas de kn_2 variables, todas estas se utilizan para contar el número de símbolos escritos en la cinta de salida.

Con un razonamiento similar al empleado en el caso de ϕ_1 se concluye que, si $\overline{u} = (\overline{u}_1, \ldots, \overline{u}_{n_2}) \in A^{kn_2}$, entonces:

$\langle \mathcal{A}, \overline{u}\rangle \models \phi_2(\overline{x}_2)$ si y sólo si $R_2^{\mathcal{A}_\Sigma}(\overline{u}_1, \ldots, \overline{u}_{n_2})$ es cierto.

Las fórmulas ϕ_3, ..., ϕ_r, ψ_1, ..., ψ_c se definen de manera análoga. □

Agradecimiento: a los dos árbitros de Acta Científica Venezolana por la minuciosa revisión que hicieron de la versión preliminar de este artículo, contribuyendo con sus comentarios a la mejoría de mi exposición.

REFERENCIAS

1. **Abiteboul, S.** y **Vianu, V.** Generic computation and its complexity. *Proc. 23rd. Ann. ACM Symp. on Theory of Computing*, IEEE Press: 209–219, 1991.

2. **Arratia-Quesada, A. A.** On the existence of normal forms for logics that capture complexity classes. Tésis PhD, University of Wisconsin–Madison, 1997.

3. **Arratia, A.** On the descriptive complexity of a simplified game of Hex. *Logic J. of the IGPL*: 10: 2, 105–122, 2002.

4. **Cook, S.** The complexity of theorem proving procedures. *Proc. 3rd Ann. ACM Symp. on Theory of Computing*: 151–158, 1971.

5. **Ebbinghaus, H.D.** y **Flum, J.** Finite Model Theory. Springer-Verlag, 1995.

6. **Ebbinghaus, H.D. Flum, J.** y **Thomas, W.** Mathematical Logic. Springer-Verlag, 1984.

7. **Fagin, R.** Generalized first–order spectra and polynomial–time recognizable sets. En R. M. Karp, editor, *Complexity of Computation, SIAM–AMS Proc.*: 7, 43–73, 1974.

8. **Fagin, R.** Finite model theory – a personal perspective, *Theoret. Comp. Sci.*: 116, 3–31, 1993.

9. **Garey, M. R.** y **Johnson, D. S.** Computers and Intractability. Freeman, San Francisco, 1979.

10. **Grädel, E.** Capturing complexity classes by fragments of Second Order logic. *Theoret. Comp. Sci.*: 101, 35–57, 1992.

11. **Hopcroft, J.** y **Ullman, J.** Introduction to Automata Theory, Languages and Computation. Addison–Wesley, 1979.

12. **Immerman, N.** Languages that capture complexity classes. *SIAM Journal on Computing*: 16, 760–778,1987.

13. **Immerman, N.** Nondeterministic space is closed under complement. *SIAM Journal on Computing*: 17, 935–938, 1988.

14. **Immerman, N.** Descriptive Complexity. Springer, 1998.

15. **Immerman, N.** y **Kolaitis, Ph.** (eds.). Descriptive Complexity and Finite Models, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 31, 1996.

16. **Lindström, P.** First order predicate logic with generalized quantifiers. *Theoria*: 32, 186–195, 1966.

17. **Papadimitriou, C.H.** Computational Complexity. Addison–Wesley, 1994.

18. **Pudlák, P.** The observational predicate calculus and complexity of computations (preliminary communication). *Commentationes Mathematicae Universitatis Carolinae*: 16, 2, 395–398, 1975.

19. **Stewart, I.A.** Comparing the expressibility of languages formed using NP–complete operators. *J. Logic Computat.*: 1 : 3, 305–330, 1991.

20. **Stewart, I.A.** Using the Hamiltonian path operator to capture NP. *J. Comp. Syst. Sci.*: 45, 127-151, 1992.

21. **Stewart, I.A.** Logical description of monotone NP problems. *J. Logic Computat.*: 4, 337–357, 1994.

22. **Szelepcsényi, R.** The method of forced enumeration for nondeterministic automata. *Acta Informatica*: 26, 279–284, 1988.

VOLUMEN 53 – N° 2 – 2002, CODEN ACVEAV 53 (2) 71-164
ISSN 0001 5504, Depósito Legal pp 195002 DF 456, Caracas–Venezuela

acta científica venezolana

Asociación Venezolana para el Avance de la Ciencia

ACTA CIENTIFICA VENEZOLANA es una revista multidisciplinaria que considera para su publicación trabajos originales en cualquier área de la ciencia. Un *Artículo* describe un estudio completo y definitivo. Una *Nota* un proyecto completo, pero más corto, que se refiere a hallazgos originales o importantes modificaciones de técnicas ya descritas. Un *Ensayo* trata aspectos relacionados con la ciencia pero no está basado en resultados originales. Una *Revisión* es un artículo solicitado por invitación de la comisión editora y comenta la literatura más reciente sobre un tema especializado. *Avances de Investigación* da cabida a comunicaciones sobre investigaciones en marcha que ameritan una rápida difusión. Las secciones *Editorial* y *Opinión* están abiertas a toda la comunidad científica. Instrucciones precisas para la presentación definitiva del texto, aparecen en el primer número de cada volumen.

Los manuscritos deben ser enviados por triplicado al Editor, quien los someterá a revisión crítica de al menos dos árbitros. La aceptación de los manuscritos está basada en el contenido científico y en la presentación de acuerdo a las normas editoriales de la revista. Se aceptarán trabajos escritos en castellano, portugués o inglés. Los manuscritos enviados para publicación deben ser concisos y correctos en su estilo y en el uso de abreviaturas. El hecho de someter un trabajo implica que el mismo no ha sido publicado ni está sometido a consideración de otra revista científica.

Los trabajos deben ir acompañados por un resumen, tanto en español como en inglés. El autor debe anexar un título en español y uno en inglés. Debe también indicar un título más breve, en el mismo idioma del artículo, para ser utilizado como encabezamiento de cada página, y una lista de palabras clave, tanto en el idioma original como en inglés. Las figuras y fotografías deben identificarse en el reverso a lápiz con el número que le corresponde y el nombre del primer autor y título del trabajo. Debe presentarse una lista de figuras junto con las leyendas de cada una, escritas a doble espacio en hojas separadas. Cada tabla debe también presentarse en hoja aparte. En general, recomendamos a los autores acompañar el texto del trabajo con una lista de todos los anexos: figuras, fotografías, tablas, etc. Las referencias deben estar ordenadas alfabéticamente y numeradas progresivamente.

Para asegurar mayor rapidez en la consideración de su manuscrito, se aconseja prepararlo de acuerdo a las "Instrucciones a los Autores" que se publican en el primer número de cada año, o pueden solicitarse por escrito al Editor. Invitamos a los autores a incluir, en caso de disponibilidad, una copia del texto aceptado y corregido en formato electrónico (PC ó Macintosh), para facilitar el rápido proceso de las pruebas.

La Comisión Editora es responsable de los comentarios y editoriales que aparezcan sin firma. Las opiniones expresadas no traducen necesariamente el criterio de la Asociación Venezolana para el Avance de la Ciencia, ni obligan a sus miembros. Los lectores están cordialmente invitados a expresar su opinión en la sección *Cartas al Editor*.

Tarifas de suscripción anual: Bs. 20000 para Bibliotecas Nacionales. Bs. 16000 para no miembros de AsoVAC. US$ 180 para Bibliotecas extranjeras y US$ 100 para suscriptores individuales fuera de Venezuela.

ACTA CIENTIFICA VENEZOLANA is a multidisciplinary journal, publishing research papers in any scientific topic. *Articles* should describe a complete and definitive study. *Notes* should describe a complete project, shorter, and usually referring to original findings or important modifications of previously described techniques. *Essays* discuss general scientific problems but are not based on original results. *Reviews* are published only by request of the Editors and discuss the most recent literature on a given subject. The section *Research Advances* will publish short communications of results which deserve immediate publication. The sections *Editorial* and *Opnion* are open to all the scientific community. Precise instructions to present manuscripts appear in the first number of each volume.

Manuscripts should be sent in original and two copies to the Editors, and will be critically reviewed by at least two referees. Acceptance of papers for publication will be based only on their scientific content and on the presentation of the material according to Acta's editorial norms, Manuscripts can be presented in Spanish, Portuguese or English. English and Spanish abstracts should be provided in all cases, with the paper's complete title translated into these languages. A shorter title in the paper's original language should be included for use as running head. A list of key words in the same language and in English should be included.

Manuscripts submitted for publication should be concise and appropriate in style and use of abbreviations. Submission of a paper implies it has not been published nor is being considered for publication by any other journal. Figures, photographs and tables should be clearly numbered and identified by the first author and short title written with a pencil in their reverse side. Figure legends should be typed on a separate sheet. A list of all material accompanying the manuscript (such as figures, photos, tables, etc.) should be included. References, at the end of the manuscript, should be ordered alphabetically, numbered progressively.

In order to ensure prompt attention to each manuscript, authors are advised to consult the "Instructions to Authors" which appear in the first issue of each year and can be obtained from the Editorial Board. We invite our contributors to submit, if possible, a copy of the text in electronic format (PC or Macintosh) for faster processing.

The Editorial Board is responsible for all commentaries and editorials which are unsigned. AsoVAC does not necessarily agree with any opinions expressed in Acta Cientifica Venezolana, nor these opinions represent those of any individual member. Readers are invited to make comments by sending letters to the Editor.

Yearly subscription rates: US$ 180 for foreign libraries and US$ 100 for individual subscriptions outside of Venezuela.

Esta Revista ha sido financiada parcialmente por:
Consejo Nacional de Investigaciones Científicas y Tecnológicas (CONICIT). Fundación Venezolana para el Avance de la Ciencia (FundaVAC). CDCH de la Universidad Central de Venezuela.CDCHT de la Universidad Centroccidental Lisandro Alvarado. CDCHT de la Universidad de Los Andes. CDCH de la Universidad de Carabobo;

Sistemas de Referencia:
Biological Abstracts; Biblioteca Regional de Medicina (BIREME); Bowker Serial Directories; Cambridge Scientific Abstracts; Chemical Abstracts Service; Current titles in Ocean, Coastal, Lake & Waterway Sciences; International Bibliography of Periodical Literature (IBZ); Index Medicus; Mathematical Reviews; Periodica CICH-UNAM; Revencyt; University Microfilm International; ZellerDietrich Bibliographische Verlage; Zentralblat fur Mathematik; Zoological Record; Current Awareness in Biological Sciences.

Versión Electrónica: http://acta.ivic.ve
Depósito Legal: pp195002cs1154
ISSN: 1317-8202