ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE CIENCIAS

COMPLEJIDAD COMPUTACIONAL DE PROBLEMAS DE DISTRIBUCIÓN JUSTA DE RECURSOS DESDE EL PUNTO DE VISTA DESCRIPTIVO

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE MATEMÁTICO

PROYECTO DE INVESTIGACIÓN

PAÚL ALEJANDRO RISCO ITURRALDE
paul.risco@yahoo.com

Director: DR. NERIO ALBERTO BORGES VILORIA
nerio.borges@epn.edu.ec

Codirector: DR. DIEGO FERNANDO RECALDE CALAHORRANO
diego.recalde@epn.edu.ec

QUITO, JULIO 2022

DECLARACIÓN

Yo PAÚL ALEJANDRO RISCO ITURRALDE, declaro bajo juramento que el trabajo aquí escrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual, correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normatividad institucional vigente.

Paúl Alejandro Risco Iturralde

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por PAÚL ALEJANDRO RISCO ITURRALDE, bajo nuestra supervisión.

Dr. Nerio Alberto Borges Viloria
Director del Proyecto

Dr. Diego Fernando Recalde Calahorrano
Codirector del Proyecto

AGRADECIMIENTOS

A todas las personas que hicieron posible mi llegada a esta etapa.

Agradezco al Dr. Nerio Borges, director de este trabajo, por su guía y paciencia. Me reuní con él casi todas las semanas que duró este proyecto. Agradezco también al Dr. Diego Recalde por aceptar ser el codirector del mismo.

Doy gracias a quienes publican material académico gratuito en Internet, así como a quienes distribuyen a través de la red programas informáticos sin costo. Cabe mencionar aquí al MSc. Andrés Merino por la plantilla que elaboró para los trabajos de titulación de la Facultad de Ciencias.

Finalmente agradezco a toda mi familia por su apoyo, en especial a mis padres.

DEDICATORIA

A mis padres, Rubén y Giovanna.
Les pido disculpas por todos los malos ratos que les hice pasar durante mi carrera universitaria.

Índice general

Resumen VIII

Abstract IX

1. Introducción 1

2. Preliminares 4
2.1. Lógica 4
2.1.1. Lógica proposicional 4
2.1.2. Lógica de primer orden 8
2.1.3. Lógica de segundo orden 14
2.2. Complejidad computacional 19
2.3. Teoría descriptiva de la complejidad computacional 22
2.3.1. Superfluidad 31

3. Distribución justa de recursos indivisibles 36
3.1. Preferencias binarias 41
3.2. Preferencias aditivas 43

4. Caracterización descriptiva de la complejidad computacional de algunos problemas de distribución justa 46
4.1. Problema 1: dos agentes con preferencias binarias monótonas idénticas capturadas por una fórmula proposicional 47
4.2. Problema 2: preferencias aditivas restringidas 53
4.2.1. Construcción de una sentencia que define el problema 54
4.2.2. Reducción 70
4.3. Problema 3: preferencias aditivas no restringidas 84

5. Conclusiones y trabajo futuro 89

A. Distribución justa: proposiciones auxiliares **93**

B. Caracterización descriptiva de la complejidad computacional de algunos problemas de distribución justa: proposiciones auxiliares **97**

B.1. Problema 1 97

B.2. Problema 2 100

C. Algoritmos **108**

C.1. Problema 1 108

C.2. Problema 2 112

Bibliografía **123**

Resumen

En la teoría descriptiva de la complejidad computacional, las proyecciones de primer orden son un tipo de reducción muy débil entre problemas de decisión. En este trabajo de fin de carrera se prueba NP-hardness, vía proyecciones de primer orden, de tres problemas de distribución justa analizados en [4]. En uno de los casos el resultado se obtiene a partir de una generalización del concepto de superfluidad encontrado en [2]. En todos los problemas la cuestión es la existencia de una asignación Pareto-eficiente y libre de envidia.

Palabras clave: complejidad computacional descriptiva, proyecciones de primer orden, NP-hardness, distribución justa de recursos indivisibles, asignación Pareto-eficiente y libre de envidia

Abstract

In the context of descriptive computational complexity, first-order projections are many-one reductions that possess very little computational power. In this undergraduate project, three fair division problems found in [4] are proved NP-hard via first-order projections. In order to get one of the results, a generalization of the concept of superfluity found in [2] is used. All the problems involve the existence of a Pareto-efficient and envy-free allocation.

Keywords: descriptive complexity, first-order projections, NP-hardness, fair division of indivisible goods, Pareto-efficient and envy-free allocation

Capítulo 1

Introducción

La complejidad computacional es una rama de la informática teórica que, a grosso modo, estudia la cantidad de recursos que utilizan los algoritmos. En esta, un problema computacional es definido como una pregunta a ser respondida que incluye la descripción de sus parámetros, o variables libres, y de los detalles que la respuesta, o solución, debe poseer. Así, una instancia de un problema corresponde a una asignación de valores para sus parámetros. Por otro lado, un algoritmo que resuelve un problema es un procedimiento general que asigna a cada instancia del problema una solución.

La complejidad computacional de un algoritmo es la cantidad de recursos requeridos para ejecutarlo. Si comparamos a través de este concepto algoritmos que resuelven el mismo problema, es natural considerar más eficiente a aquel que utiliza menos recursos. De esta manera, la complejidad del algoritmo más eficiente que resuelve un problema constituye una medida de la dificultad, o complejidad computacional, de este último [8].

La teoría descriptiva de la complejidad computacional estudia la dificultad de los problemas con herramientas de la lógica matemática. Así por ejemplo, la clase NP, conformada por aquellos problemas de decisión cuyas soluciones pueden ser verificadas en tiempo polinomial por una máquina determinista de Turing, ha sido caracterizada como el conjunto de problemas que pueden ser descritos en la lógica existencial de segundo orden. Se observa que esta caracterización no hace mención alguna de máquinas o tiempo [9]. Así, la teoría descriptiva permite convertir los problemas, métodos, y resultados de la complejidad computacional en nociones de la lógica, y viceversa, ampliando las posibilidades metodológicas para ambas [7] (sección 7.5).

En el área de distribución justa de recursos aparecen problemas computacionales. En general, se considera un conjunto de dos o más agentes y un conjunto de bienes; los bienes deben ser asignados a los agentes con criterios de eficiencia y/o justicia, usualmente matemáticamente formulados. Una vez declaradas las preferencias de los agentes con respecto a los paquetes de bienes, se puede exigir por ejemplo una asignación tal que ningún agente prefiera estrictamente el paquete asignado a otro agente; en este caso se considera que no existe resentimiento social o envidia [4].

Como en cualquier otro contexto en el que surgen problemas computacionales a ser resueltos en la práctica, la complejidad es un factor importante, pues si no existen algoritmos eficientes para abordarlos, en gran parte de los casos no hay avance. Previo a la presentación del plan de este proyecto, el autor no había encontrado en la literatura análisis de complejidad computacional de problemas de distribución justa desde el enfoque descriptivo. Así, se plantea el objetivo de estudiar tres problemas descritos en [4] y para los cuales no se conoce algoritmos eficientes, caracterizando su complejidad usando fragmentos adecuados de la lógica de segundo orden, a diferencia de analizar la complejidad de algoritmos que los resuelvan.

En todos los problemas que analizaremos, la cuestión es la existencia de asignaciones eficientes y libres de envidia. En el primero habrá solo dos agentes con las mismas preferencias, las cuales a su vez serán binarias, es decir, existe una colección de paquetes deseados y una de paquetes no deseados. En el resto de problemas, los agentes asignarán pesos a los bienes; uno consistirá en una restricción del otro.

Los dos primeros problemas que analizaremos están en la clase de complejidad NP. El último está en una clase que contiene a NP. Además, todos estos problemas son NP-hard vía reducciones polinomiales, es decir, dado cualquier problema de la clase de complejidad NP, y dado uno de los problemas que analizaremos, existe una aplicación que transforma, en tiempo polinomial, cada instancia del primero en una instancia equivalente del segundo. En este proyecto, nuestro interés es demostrar, con herramientas de la lógica matemática, que los tres problemas son NP-hard vía proyecciones de primer orden. Las proyecciones de primer orden son reducciones que poseen muy poco poder computacional: básicamente, cada bit de una instancia en la imagen de la aplicación depende a lo mucho de un bit de la instancia correspondiente en el dominio. Para probar NP-hardness en el caso del último problema, introducimos una modificación de la definición de superfluidad encontrada en [2].

El documento está organizado de la siguiente manera. En el capítulo 2 presentamos los conceptos de lógica y complejidad computacional requeridos para entender el trabajo. Este tiene tres secciones. En la tercera se desarrolla la modificación del concepto de superfluidad mencionada en el párrafo previo. En el capítulo 3 se da definiciones y se presenta proposiciones del área de distribución justa de recursos indivisibles. El capítulo 4 es la parte principal de este trabajo. Ahí se caracteriza, a través de técnicas sintácticas, la complejidad computacional de los tres problemas de distribución justa mencionados. El último capítulo contiene las conclusiones y las direcciones en las que este proyecto podría continuar.

Capítulo 2

Preliminares

En este capítulo presentamos los conceptos y resultados necesarios para entender el trabajo. Comenzamos con las nociones de lógica, y luego revisamos de manera breve complejidad computacional para finalmente presentar la teoría descriptiva.

2.1. Lógica

Dado un conjunto de símbolos de un lenguaje, en esta sección explicaremos qué son las fórmulas bien formadas de ese lenguaje y cómo interpretarlas, es decir, trataremos las partes sintáctica y semántica, esto para la lógica proposicional, la lógica de primer orden, y la lógica de segundo orden.

La lógica proposicional es la más básica. En base a ella introducimos la lógica de primer orden, y en base a esta última, la de segundo orden.

Fórmulas bien formadas de la lógica proposicional serán usadas en este trabajo para describir las preferencias de los agentes en ciertos problemas de distribución justa. Por otro lado, se utilizará fórmulas de primer orden y segundo orden para caracterizar la complejidad computacional de algunos problemas en esa área.

El contenido de esta sección está basado principalmente en el capítulo 1 y el apéndice A de [11], el capítulo 1 de [9], y la sección 2.2 de [4]. La exposición se ha hecho adaptando los conceptos a las necesidades del proyecto.

2.1.1. Lógica proposicional

Sea $V = \{x_0, \ldots, x_{m-1}\}$ un conjunto de m variables proposicionales, es decir, un conjunto de símbolos tal que sus miembros pueden tomar solo los valores de verdad

true o false. Notamos con $\top$ a la constante lógica que toma solo el valor true, y con $\bot$ a aquella que toma solo el valor false.

Ahora, consideremos el conjunto de símbolos $\{\neg, \vee, \wedge\}$. Estos son conocidos como *conectivas lógicas*. Tenemos las siguientes definiciones.

Definición 2.1 (Fórmula bien formada de L_V)**.** Todos los miembros de $V \cup \{\top, \bot\}$ son *fórmulas bien formadas del lenguaje proposicional* L_V. Además, si φ y ψ son fórmulas bien formadas de L_V, las siguientes también lo son:

- $(\neg\varphi)$, conocida como la *negación de* φ,
- $(\varphi \vee \psi)$, llamada la *disyunción de* φ *y* ψ ,y
- $(\varphi \wedge \psi)$, conocida como la *conjunción de* φ *y* ψ.

Solo las fórmulas mencionadas son fórmulas bien formadas de L_V.

Ejemplo 2.1. Si $V = \{x_0, x_1, x_2\}$, las siguientes son fórmulas bien formadas de L_V:

$$x_1,$$
$$((x_0 \vee x_2) \wedge x_1), \text{ y} \tag{2.1}$$
$$(x_2 \wedge \top). \tag{2.2}$$

Sin ningún riesgo, podemos escribir las fórmulas 2.1 y 2.2, respectivamente, como

$$(x_0 \vee x_2) \wedge x_1, \text{ y}$$
$$x_2 \wedge \top.$$

Definición 2.2 (Fórmula positiva)**.** Una fórmula bien formada de L_V es *positiva* si no contiene ocurrencias del símbolo $\neg$.

Definición 2.3 (Asignación de valores de verdad)**.** Una *asignación de valores de verdad* para los miembros de V es una función $M : V \to \{\text{true}, \text{false}\}$.

Dada una asignación de valores de verdad para los miembros de V, cada fórmula bien formada de L_V toma exactamente un valor de verdad de acuerdo a las tablas de verdad usuales para $\neg$, $\vee$, y $\wedge$.

Definición 2.4 (Satisfacción de fórmulas de L_V)**.** Si, bajo una asignación de valores de verdad M, la fórmula φ de L_V toma el valor de verdad true, escribiremos

$$M \models \varphi$$

y diremos que M *satisface* φ.

Ahora, podemos pensar en una asignación de valores de verdad M para las variables en V, como un subconjunto de V tal que,

$$x_i \in M \iff M(x_i) = \text{true}.$$

Con esto, a partir de las tablas de verdad de las conectivas lógicas, si M es una asignación de valores de verdad para los miembros de V, y φ y ψ son fórmulas en L_V, entonces:

$$M \models \top,$$
$$M \not\models \bot,$$

$$\begin{aligned} M \models x_i &\iff x_i \in M, \\ M \models (\neg\varphi) &\iff M \not\models \varphi, \\ M \models (\varphi \vee \psi) &\iff M \models \varphi \text{ o } M \models \psi, \text{ y} \\ M \models (\varphi \wedge \psi) &\iff M \models \varphi \text{ y } M \models \psi. \end{aligned}$$

Observación 2.1. Al igual que con las expresiones aritméticas, los paréntesis en una fórmula bien formada de L_V son usados para indicar el orden en el que se procede a operar para determinar el valor de verdad de la fórmula. Hacemos notar que las operaciones binarias que definen las conectivas $\vee$ y $\wedge$ son conmutativas y asociativas. En efecto, si φ_1, φ_2, y φ_3 son fórmulas bien formadas de L_V, se puede probar que

$$\begin{aligned} M \models (\varphi_1 \vee \varphi_2) &\iff M \models (\varphi_2 \vee \varphi_1), \text{ y} \\ M \models ((\varphi_1 \vee \varphi_2) \vee \varphi_3) &\iff M \models (\varphi_1 \vee (\varphi_2 \vee \varphi_3)). \end{aligned}$$

Lo mismo se tiene en el caso de la conectiva $\wedge$. Luego, podemos escribir una fórmula como

$$((x_0 \vee x_2) \vee (x_1 \vee x_2))$$

de esta manera:

$$x_0 \vee x_2 \vee x_1 \vee x_2.$$

En este trabajo seguiremos esta práctica.

Definición 2.5 (Fórmulas lógicamente equivalentes)**.** Decimos que las fórmulas φ y ψ de L_V son *lógicamente equivalentes* si, para toda asignación de valores de verdad

M,

$$M \models \varphi \iff M \models \psi.$$

En este caso usaremos la notación $\varphi \equiv \psi$.

Definición 2.6 (Literal)**.** Un *literal* es un miembro de V o la negación de un miembro de V.

Definición 2.7 (Fórmula proposicional en forma normal disyuntiva)**.** Una fórmula φ de L_V está en *forma normal disyuntiva*, o *FND*, si es una disyunción de una o más fórmulas de L_V, cada una de las cuales es una conjunción de uno o más literales.

Definición 2.8 (Fórmula proposicional en forma normal conjuntiva)**.** Una fórmula φ de L_V está en *forma normal conjuntiva*, o *FNC*, si es una conjunción de una o más fórmulas de L_V, cada una de las cuales es una disyunción de uno o más literales.

La terminología en las últimas dos definiciones considera a un literal como una conjunción degenerada y como una disyunción degenerada.

Ahora, las fórmulas bien formadas de L_V que están en FNC, pueden ser representadas por una colección de conjuntos, como lo hacemos en el siguiente ejemplo.

Ejemplo 2.2. La fórmula

$$(\neg x_0) \wedge (x_1 \vee x_2) \wedge (x_1 \vee x_2) \wedge (x_3 \vee x_4 \vee (\neg x_1))$$

está en FNC. Notar que dos de las disyunciones de literales son las mismas. Podemos representar esta fórmula por la colección de conjuntos

$$\{\{\neg x_0\}, \{x_1, x_2\}, \{x_3, x_4, \neg x_1\}\}.$$

Definición 2.9 (Cláusula de una fórmula proposicional en FNC)**.** Si φ es una fórmula de L_V que está en FNC, y es representada como una colección de conjuntos como en el ejemplo 2.2, cada uno de los conjuntos en cuestión recibe el nombre de *cláusula*.

Observación 2.2. Recíprocamente, las colecciones de cláusulas tienen asociadas fórmulas de L_V. Una colección de cláusulas no vacías naturalmente da lugar a una fórmula en FNC, pero se admite también las colecciones vacías y las cláusulas vacías. Ahora, una fórmula en FNC es satisfecha por una asignación M si y solo si M satisface cada una de las disyunciones involucradas. Además, M satisface una disyunción si y solo si satisface al menos un literal en ella. Generalizando, M satisface una colección de cláusulas si y solo si M satisface cada una de las cláusulas de la colección, y M satisface una cláusula si y solo si M satisface al menos un literal en la cláusula. Notar que una cláusula vacía no puede ser satisfecha. En la sección

4.1 se trabajará con tuplas de cláusulas, a diferencia de colecciones de cláusulas. Sin embargo, a partir de cualquiera de estas tuplas, se puede formar la colección de las cláusulas de la tupla, y a esta colección se puede asociar una fórmula ψ de L_V. Luego, $M \models \psi$ si y solo si M satisface todas las cláusulas de la colección, es decir, si y solo si M satisface todas las cláusulas de la tupla. Así, si alguna cláusula es vacía, $\psi \equiv \bot$.

Tenemos la siguiente proposición.

Proposición 2.1. *Sean* $V = \{x_0, x_1, \ldots, x_{n-1}\}$ *un conjunto de variables proposicionales, y*

$$\langle Y_0, Y_1, \ldots, Y_{n-1} \rangle$$

una tupla de dos o más cláusulas que pueden contener solo variables proposicionales. Entonces, al menos una cláusula en la tupla es vacía si y solo si la fórmula asociada a la tupla, ψ*, es lógicamente equivalente a* $\bot$*.*

Demostración. Primero, en este contexto y para cualquier i, una asignación de valores de verdad M satisface Y_i si y solo si existe $x_k \in Y_i$ tal que $x_k \in M$.

Ahora, supongamos existe una cláusula Y_j que es vacía. Por lo expuesto en el anterior párrafo, para todo $M \subseteq V$, M no satisface Y_j, con la consecuencia $M \not\models \psi$. De esta manera, $\psi \equiv \bot$.

Recíprocamente, si $\psi \equiv \bot$ y ninguna cláusula de la tupla es vacía, tomando $M = \{x_0, x_1, \ldots, x_{n-1}\}$, vemos que M satisface todas las cláusulas de la tupla. En otras palabras, $M \models \psi$, lo cual es contradictorio. Concluimos que, si $\psi \equiv \bot$, necesariamente ψ tiene al menos una cláusula vacía. □

2.1.2. Lógica de primer orden

Sean los siguientes conjuntos de símbolos:

- $VAR = \{y_i | \, i \in \mathbb{Z}^+\}$ el conjunto de *variables*,
- $\{c_i | \, i \in \mathbb{Z}^+\}$ el de *símbolos de constante*,
- $\left\{R_i^j | \, i, j \in \mathbb{Z}^+\right\}$ el de *símbolos de relación*,
- $\{(,), , \}$ el de *delimitadores*,
- $\{\exists, \forall\}$ el de *símbolos de cuantificación*,
- $\{\neg, \vee, \wedge, \rightarrow, \leftrightarrow, \oplus\}$ el de *conectivas lógicas*, y
- $\{\textbf{true}, \textbf{false}\}$ el de *constantes lógicas*.

El símbolo R_i^j es un símbolo de relación de aridad j. La unión del conjunto de variables y el de símbolos de constante es el conjunto de *términos*.

Definición 2.10 (Fórmula bien formada de primer orden con vocabulario σ)**.** Consideremos la tupla

$$\sigma = \langle R_1^{a_1}, \ldots, R_r^{a_r}, c_1, \ldots, c_s \rangle,$$

conformada por símbolos de relación y de constante. σ es llamado *vocabulario relacional* o simplemente *vocabulario*. Ahora, sea $i \in \{1, \ldots, r\}$, y sean $t_1, \ldots, t_{a_i}$ términos en $VAR \cup \{c_1, \ldots, c_s\}$. Luego,

$$R_i^{a_i}(t_1, \ldots, t_{a_i})$$

es una *fórmula atómica con vocabulario* σ. Las fórmulas atómicas con vocabulario σ, **true**, y **false**, son *fórmulas bien formadas (fbf) de primer orden con vocabulario* σ. Además, si α y β son fbf de primer orden con vocabulario σ, y si $y \in VAR$, las siguientes también son fbf de primer orden con vocabulario σ:

- la *negación de* α, $(\neg\alpha)$,
- la *disyunción de* α *y* β, $(\alpha \vee \beta)$,
- la *conjunción de* α *y* β, $(\alpha \wedge \beta)$,
- el *condicional con antecedente* α *y consecuente* β, $(\alpha \to \beta)$,
- el *bicondicional con antecedente* α *y consecuente* β, $(\alpha \leftrightarrow \beta)$,
- la *disyunción exclusiva de* α *y* β, $(\alpha \oplus \beta)$,
- $((\exists y)\alpha)$, y
- $((\forall y)\alpha)$.

Solo las consideradas son fbf de primer orden con vocabulario σ. Si α es una fbf de primer orden con algún vocabulario σ, la llamaremos *fórmula bien formada de primer orden*.

Ejemplo 2.3. Sea

$$\sigma = \langle R_1^2, R_2^1, c_1 \rangle.$$

Las siguientes expresiones son fbf de primer orden con vocabulario σ.

$$((\forall y_1)(R_1^2(c_1, y_1) \vee R_2^1(y_1))) \tag{2.3}$$

$$((\exists y_1)((\forall y_3)R_1^2(y_3, y_1))) \tag{2.4}$$

$$R_1^2(y_{66}, y_4) \tag{2.5}$$

$$(((\exists y_3)R_2^1(y_3)) \oplus R_1^2(y_{67}, y_3)) \tag{2.6}$$

Sin riesgo, podemos escribir las fórmulas 2.3, 2.4, y 2.6, respectivamente, así:

$$(\forall y_1)(R_1^2(c_1, y_1) \vee R_2^1(y_1))$$
$$(\exists y_1)(\forall y_3)R_1^2(y_3, y_1)$$
$$((\exists y_3)R_2^1(y_3)) \oplus R_1^2(y_{67}, y_3)$$

Observar que se ha eliminado paréntesis.

Para facilitar la lectura de fórmulas, se usará como símbolos de variables x, y, z_1, z_2, y otros. Asimismo, se usará símbolos de relación Q^2, V^3, entre otros. Además, se eliminará paréntesis sin dar cabida a ambigüedades.

Definición 2.11 (Variables libres y ligadas)**.** Sea α una fbf de primer orden. Entonces $((\forall x)\alpha)$ también es una fbf de primer orden. En este caso, decimos que α es el *alcance del cuantificador* $(\forall x)$. Asimismo, para la fórmula $((\exists x)\alpha)$, decimos que α es el *alcance del cuantificador* $(\exists x)$. Ahora, si β es una fbf de primer orden, y la variable x ocurre en β como parte de un cuantificador $(\forall x)$ o $(\exists x)$, o en el alcance de alguno de esos cuantificadores, decimos que esa ocurrencia de x es una ocurrencia *ligada*; en caso contrario, la llamamos ocurrencia *libre*. Ahora, si x ocurre de manera libre en β, la llamamos una *variable libre de* β; si ocurre de manera ligada, la llamamos una *variable ligada de* β.

Ejemplo 2.4. Consideremos la fórmula 2.3. Todas las ocurrencias de y_1 en esa fórmula son ligadas; sigue que y_1 es una variable ligada de esa fórmula. Por otro lado, en la fórmula 2.4 todas las ocurrencias de y_1 y y_3 son ligadas; entonces esas variables son variables ligadas. En cuanto a la fórmula 2.5, tanto la ocurrencia de y_{66} como de y_4 son libres; luego, esas variables son variables libres de esa fórmula. Finalmente, y_3 ocurre de manera ligada y libre en 2.6, mientras y_{67} ocurre de manera libre; así, y_{67} es un variable libre y y_3 es una variable ligada y libre de 2.6.

Si algunas de las variables libres de α están en el conjunto $\{y_{i_1}, \ldots, y_{i_k}\}$, podemos escribir α como $\alpha(y_{i_1}, \ldots, y_{i_k})$, de manera que, si $t_1, \ldots, t_k$ son términos adecuados, expresamos como $\alpha(t_1, \ldots, t_k)$ a la fórmula que resulta de reemplazar cada ocurrencia libre de y_{i_j} en α, por t_j, esto para todo j.

Definición 2.12 (Sentencia de primer orden)**.** Una fbf de primer orden que no tiene variables libres es llamada *sentencia de primer orden*.

Pasamos ahora a la parte semántica de la lógica de primer orden con la siguiente definición.

Definición 2.13 (Estructura con vocabulario σ o σ-estructura)**.** Sea σ el vocabulario

$$\sigma = \langle R_1^{a_1}, \ldots, R_r^{a_r}, c_1, \ldots, c_s \rangle.$$

Una *estructura con vocabulario* σ, o *σ-estructura*, es una tupla

$$\mathcal{A} = \langle |\mathcal{A}|, R_1^{\mathcal{A}}, \ldots, R_r^{\mathcal{A}}, c_1^{\mathcal{A}}, \ldots, c_s^{\mathcal{A}} \rangle,$$

con $|\mathcal{A}|$ un conjunto, llamado el *universo de* $\mathcal{A}$, con $R_i^{\mathcal{A}}$ una relación de aridad a_i sobre $|\mathcal{A}|$, y con $c_j^{\mathcal{A}}$ un elemento fijo de $|\mathcal{A}|$. Con el símbolo $\|\mathcal{A}\|$ notamos a la cardinalidad de $|\mathcal{A}|$. Llamaremos a $\|\mathcal{A}\|$ el *tamaño de* $\mathcal{A}$. Además, usamos la notación STRUC$[\sigma]$ para el conjunto de todas las estructuras con vocabulario σ.

Observación 2.3. Enfatizamos que con la expresión $|\mathcal{A}|$ en la definición anterior, no estamos refiriéndonos a cardinalidad de conjunto alguno; sin embargo, en el resto de contextos sí usaremos el símbolo $|$ con este propósito.

Ejemplo 2.5. Sea

$$\sigma = \langle P^2, N^2 \rangle.$$

un vocabulario. En este no hay símbolos de constante. Sea $\mathcal{A}$ una σ-estructura de tamaño n, con $|\mathcal{A}| = \{0, 1, \ldots, n-1\}$. Entonces, podemos asociar a cada miembro i de $|\mathcal{A}|$, la variable proposicional x_i y la cláusula Y_i (ver la subsección anterior). Así, podemos definir: $(k, l) \in P^{\mathcal{A}}$ si y solo si $x_l \in Y_k$, y $(j, m) \in N^{\mathcal{A}}$ si y solo si $\neg x_m \in Y_j$. Es decir, $(k, l) \in P^{\mathcal{A}}$ si y solo si x_l ocurre de manera positiva en Y_k, y $(j, m) \in N^{\mathcal{A}}$ si y solo si x_m ocurre de manera negativa en Y_j. Un caso concreto es $\mathcal{A}$ con

$$\begin{aligned} |\mathcal{A}| &= \{0, 1, 2\}, \\ P^{\mathcal{A}} &= \{(0, 1), (2, 2)\}, \text{ y} \\ N^{\mathcal{A}} &= \{(0, 0), (0, 2), (1, 1), (1, 2)\}. \end{aligned}$$

De esta manera, tenemos que el conjunto de variables proposicionales es $V = \{x_0, x_1, x_2\}$, y la tupla de cláusulas $\langle Y_0, Y_1, Y_2 \rangle$ es tal que

$$\begin{aligned} Y_0 &= \{\neg x_0, x_1, \neg x_2\}, \\ Y_1 &= \{\neg x_1, \neg x_2\}, \text{ y} \\ Y_2 &= \{x_2\}. \end{aligned}$$

La fórmula proposicional ψ de L_V asociada es

$$((\neg x_0) \vee x_1 \vee (\neg x_2)) \wedge ((\neg x_1) \vee (\neg x_2)) \wedge x_2.$$

Ejemplo 2.6. Consideremos el vocabulario

$$\tau_g = \langle E^2 \rangle.$$

Podemos pensar en τ_g como el vocabulario de los grafos. Si $\mathcal{A}$ es una τ_g-estructura de tamaño n, con $|\mathcal{A}| = \{0, 1, \ldots, n-1\}$, podemos pensar en cada uno de los miembros de $|\mathcal{A}|$ como un vértice de un grafo, y en la relación $E^{\mathcal{A}}$ como la relación binaria tal que (i, j) es uno de sus miembros si y solo si (i, j) es una arista del grafo.

Ahora, sean σ un vocabulario, $\mathcal{A} \in \text{STRUC}[\sigma]$, $W \subseteq VAR$, y C el conjunto de símbolos de constante en σ. Una *interpretación en* $\mathcal{A}$ es una función $i : W \cup C \to |\mathcal{A}|$. Esta definición será extendida en la subsección siguiente.

En este trabajo, dados un vocabulario σ y $\mathcal{A} \in \text{STRUC}[\sigma]$, si c_k es un símbolo de constante de σ, asumiremos que toda interpretación i en $\mathcal{A}$ es tal que $i(c_k) = c_k^{\mathcal{A}}$.

Definición 2.14 (Satisfacción de fbf de primer orden)**.** Sean σ un vocabulario, $\mathcal{A} \in \text{STRUC}[\sigma]$, e i una interpretación en $\mathcal{A}$ que tiene en su dominio a las variables libres relevantes. Definimos inductivamente el concepto de *satisfacción* de una fbf de primer orden con vocabulario σ:

$$\langle \mathcal{A}, i \rangle \models \textbf{true},$$
$$\langle \mathcal{A}, i \rangle \not\models \textbf{false},$$

$$\begin{aligned}
\langle \mathcal{A}, i \rangle \models R_k^{a_k}(t_1, \ldots, t_{a_k}) &\iff (i(t_1), \ldots, i(t_{a_k})) \in R_k^{\mathcal{A}}, \\
\langle \mathcal{A}, i \rangle \models (\neg\alpha) &\iff \langle \mathcal{A}, i \rangle \not\models \alpha, \\
\langle \mathcal{A}, i \rangle \models (\alpha \vee \beta) &\iff \langle \mathcal{A}, i \rangle \models \alpha \text{ o } \langle \mathcal{A}, i \rangle \models \beta, \\
\langle \mathcal{A}, i \rangle \models (\alpha \wedge \beta) &\iff \langle \mathcal{A}, i \rangle \models \alpha \text{ y } \langle \mathcal{A}, i \rangle \models \beta, \\
\langle \mathcal{A}, i \rangle \models (\alpha \to \beta) &\iff \text{si } \langle \mathcal{A}, i \rangle \models \alpha, \text{ entonces } \langle \mathcal{A}, i \rangle \models \beta, \\
\langle \mathcal{A}, i \rangle \models (\alpha \leftrightarrow \beta) &\iff \langle \mathcal{A}, i \rangle \models \alpha \text{ si y solo si } \langle \mathcal{A}, i \rangle \models \beta, \\
\langle \mathcal{A}, i \rangle \models (\alpha \oplus \beta) &\iff \text{o bien } \langle \mathcal{A}, i \rangle \models \alpha, \text{ o bien } \langle \mathcal{A}, i \rangle \models \beta, \text{ y} \\
\langle \mathcal{A}, i \rangle \models ((\exists y)\alpha) &\iff \text{existe } d \in |\mathcal{A}| \text{ tal que } \langle \mathcal{A}, i, d/y \rangle \models \alpha,
\end{aligned}$$

con la interpretación $(i, d/y)$ definida por

$$(i, d/y)(t) = \begin{cases} i(t) & \text{si } t \neq y, \\ d & \text{si } t = y. \end{cases}$$

Finalmente,

$$\langle \mathcal{A}, i \rangle \models ((\forall y)\alpha) \iff \text{ para todo } d \in |\mathcal{A}|, \langle \mathcal{A}, i, d/y \rangle \models \alpha.$$

Si $\langle \mathcal{A}, i \rangle \models \alpha$, diremos que $\langle \mathcal{A}, i \rangle$ *satisface* α.

Ahora, dada una fbf de primer orden con vocabulario σ,

$$\alpha\left(y_{j_1}, \ldots, y_{j_l}\right),$$

y dadas una σ-estructura $\mathcal{A}$, y una interpretación i en $\mathcal{A}$ tal que $i\left(y_{j_k}\right) = d_k$, si $\langle \mathcal{A}, i \rangle \models \alpha$, podemos expresar esto con la notación

$$\mathcal{A} \models \alpha(d_1, \ldots, d_l).$$

Por otro lado, si $\mathcal{B}$ es una σ-estructura, y β es una fbf de primer orden con vocabulario σ tal que, $\langle \mathcal{B}, i \rangle \models \beta$ para toda interpretación i en $\mathcal{B}$, podemos escribir $\mathcal{B} \models \beta$. Asimismo, si $\langle \mathcal{B}, i \rangle \not\models \beta$ para toda interpretación i en $\mathcal{B}$, podemos expresar esto con $\mathcal{B} \not\models \beta$. La sentencias de primer orden verifican esta propiedad, es decir, su satisfacción, o no satisfacción, no depende de interpretación alguna.

Ejemplo 2.7. Recordemos el ejemplo 2.6, y consideremos la fórmula

$$\alpha(x) = ((\exists y)E^2(y, x)) \wedge ((\exists y)E^2(x, y)).$$

Sea $\mathcal{A}$ una τ_g-estructura como en el ejemplo mencionado, con $n = 10$, y sea $i : \{x\} \to |\mathcal{A}|$ la interpretación tal que $i(x) = 5$. Luego,

$$\mathcal{A} \models \alpha(5) \tag{2.7}$$

si y solo si

$$\langle \mathcal{A}, i \rangle \models (\exists y)E^2(y, x) \text{ y} \tag{2.8}$$
$$\langle \mathcal{A}, i \rangle \models (\exists y)E^2(x, y). \tag{2.9}$$

Ahora, 2.8 es cierta si y solo si existe $d \in |\mathcal{A}|$ tal que $\langle \mathcal{A}, i, d/y \rangle \models E^2(y, x)$, es decir, tal que $(d, 5) \in E^{\mathcal{A}}$. Además, 2.9 es cierta si y solo si existe $e \in |\mathcal{A}|$ tal que $\langle \mathcal{A}, i, e/y \rangle \models E^2(x, y)$, es decir, tal que $(5, e) \in E^{\mathcal{A}}$. Así, se tiene que 2.7 es cierta si y solo si existen $d, e \in |\mathcal{A}|$ tales que $(d, 5)$ y $(5, e)$ son aristas del grafo asociado a $\mathcal{A}$.

Ejemplo 2.8. Consideremos el ejemplo 2.5, y la sentencia de primer orden

$$\beta = ((\forall x)(\forall y)\neg N^2(x, y)) \wedge ((\forall x)(\exists y)P^2(x, y)).$$

Sea $\mathcal{B} \in \text{STRUC}[\sigma]$, con $\|\mathcal{B}\| = n$ y tal que $|\mathcal{B}| = \{0, 1, \ldots, n-1\}$. Así, tenemos $V = \{x_0, x_1, \ldots, x_{n-1}\}$, y tenemos la tupla de cláusulas $\langle Y_0, Y_1, \ldots, Y_{n-1}\rangle$. Además, $\mathcal{B} \models \beta$ si y solo si

$$\mathcal{B} \models (\forall x)(\forall y)\neg N^2(x, y) \text{ y} \tag{2.10}$$

$$\mathcal{B} \models (\forall x)(\exists y)P^2(x, y). \tag{2.11}$$

Ahora, 2.10 es cierta si y solo si, para todo $i \in |\mathcal{B}|$,

$$\langle \mathcal{B}, i/x\rangle \models (\forall y)\neg N^2(x, y).$$

Y esto es cierto si y solo si, para todo $i \in |\mathcal{B}|$ y para todo $k \in |\mathcal{B}|$, $\langle \mathcal{B}, i/x, k/y\rangle \models \neg N^2(x, y)$, es decir, si y solo si, para cualesquier $i, k \in |\mathcal{B}|$, $\langle \mathcal{B}, i/x, k/y\rangle \not\models N^2(x, y)$. En conclusión, 2.10 es cierta si y solo si, para cualesquier $i, k \in |\mathcal{B}|$, $\neg x_k \notin Y_i$.

De manera análoga, 2.11 es cierta si y solo si, para todo $l \in |\mathcal{B}|$, existe $m \in |\mathcal{B}|$ tal que $x_m \in Y_l$.

De esta manera, si $\mathcal{B} \models \beta$, entonces la fórmula ψ de L_V asociada a $\mathcal{B}$ es positiva y está en FNC. Recíprocamente, si ψ está en FNC y es positiva entonces, tomando $M = V$, vemos que $M \models \psi$. Así, toda cláusula Y_j es no vacía, pues en caso contrario ψ no podría ser satisfecha (ver observación 2.2). Es más, ninguna cláusula puede contener la negación de algún elemento de V; de lo contrario, ψ no sería positiva. Luego, 2.10 y 2.11 son ciertas. En conclusión, si ψ está en FNC y es positiva, entonces $\mathcal{B} \models \beta$.

2.1.3. Lógica de segundo orden

La lógica de segundo orden permite cuantificar relaciones sobre el universo de las estructuras.

Sea $\text{FO}(\sigma)$ el conjunto de fbf de primer orden con vocabulario σ (ver la definición 2.10). Cada una de ellas es construida con símbolos mencionados al principio de la subsección 2.1.2. Además, las fórmulas primitivas de $\text{FO}(\sigma)$ son las fórmulas atómicas, **true**, y **false**, pues a partir de estas se construye el resto de fórmulas. Aquí, en primer lugar, extendemos el conjunto de símbolos añadiendo el conjunto de *variables de segundo orden*,

$$VAR_2 = \left\{ \mathbf{X}_{\mathbf{i}}^{\mathbf{j}} | \, \mathbf{i}, \mathbf{j} \in \mathbb{Z}^+ \right\}.$$

Para todo $\mathbf{i}$, $\mathbf{j}$, $\mathbf{X}_{\mathbf{i}}^{\mathbf{j}}$ es una variable de relación de aridad j.

En segundo lugar, extendemos $\text{FO}(\sigma)$ a $\text{SO}(\sigma)$, el conjunto de *fórmulas bien formadas*

(fbf) de segundo orden con vocabulario σ, permitiendo fórmulas atómicas $\mathbf{X_i^j}(t_1, \ldots, t_j)$, con $t_1, \ldots, t_j$ términos adecuados. Luego, si Φ y Ψ son miembros de $\mathrm{SO}(\sigma)$, la negación, la disyunción, la conjunción, etcétera, de estas, también son fbf de segundo orden con vocabulario σ; además, incluimos las fórmulas $((\exists \mathbf{X})\Phi)$ y $((\forall \mathbf{X})\Phi)$, con $\mathbf{X} \in VAR_2$. Solo las consideradas son fórmulas bien formadas de $\mathrm{SO}(\sigma)$. Si Φ es una fbf de $\mathrm{SO}(\sigma)$ para algún vocabulario σ, la llamaremos *fórmula bien formada de segundo orden*.

Los cuantificadores que cuantifican variables de segundo orden serán llamados *cuantificadores de segundo orden*; el resto serán *cuantificadores de primer orden*.

Notar que, dado un vocabulario σ, $\mathrm{FO}(\sigma) \subset \mathrm{SO}(\sigma)$.

Con el objetivo de facilitar la lectura de fórmulas, usaremos los símbolos $\mathbf{P^2}$, $\mathbf{S^5}$, y otros, para representar variables de segundo orden. Además, sin dar lugar a ambigüedades, omitiremos algunos paréntesis en las fórmulas.

Por otro lado, también extendemos los conceptos de variable libre y ligada a las variables de segundo orden (ver definición 2.11). Así, si Φ es miembro de $\mathrm{SO}(\sigma)$, y sus variables libres están en la unión de los conjuntos $\left\{\mathbf{X_{b_1}^{d_1}}, \ldots, \mathbf{X_{b_p}^{d_p}}\right\}$ y $\left\{y_{j_1}, \ldots, y_{j_q}\right\}$, podemos escribir Φ así:

$$\Phi\left(\mathbf{X_{b_1}^{d_1}}, \ldots, \mathbf{X_{b_p}^{d_p}}, y_{j_1}, \ldots, y_{j_q}\right);$$

y con

$$\Phi\left(\mathbf{Z_{e_1}^{d_1}}, \ldots, \mathbf{Z_{e_p}^{d_p}}, t_1, \ldots, t_q\right)$$

podemos notar a la fórmula de segundo orden que resulta de reemplazar cada ocurrencia libre de $\mathbf{X_{b_k}^{d_k}}$ en Φ por $\mathbf{Z_{e_k}^{d_k}}$, y cada ocurrencia libre de y_{j_m} por el término t_m.

Naturalmente, una *sentencia de segundo orden* es una fórmula bien formada de segundo orden que no tiene variables libres.

Definición 2.15 (Interpretación)**.** Sean σ un vocabulario, $\mathcal{A} \in \mathrm{STRUC}[\sigma]$, $W \subseteq VAR$, $W_2 \subseteq VAR_2$, y C el conjunto de símbolos de constante en σ. Una *interpretación en* $\mathcal{A}$ es una función i con dominio $W \cup W_2 \cup C$ tal que, a cada miembro de $W \cup C$ le asigna un miembro de $|\mathcal{A}|$, y a cada $\mathbf{X_k^j} \in W_2$ le asigna una relación de aridad j sobre $|\mathcal{A}|$.

Al igual que en la subsección anterior, mencionamos que, dados un vocabulario σ y $\mathcal{A} \in \mathrm{STRUC}[\sigma]$, si c_k es un símbolo de constante de σ, asumiremos que toda interpretación i en $\mathcal{A}$ es tal que $i(c_k) = c_k^{\mathcal{A}}$.

Ahora extendemos la definición 2.14 a la de satisfacción de fórmulas de segundo orden. Sean σ un vocabulario, $\mathcal{A} \in \text{STRUC}[\sigma]$, e i una interpretación en $\mathcal{A}$ que tiene en su dominio a las variables libres relevantes. Entonces,

$$\langle \mathcal{A}, i \rangle \models \mathbf{X}_{\mathbf{k}}^{\mathbf{j}} (t_1, \ldots, t_j) \iff (i(t_1), \ldots, i(t_j)) \in i\left(\mathbf{X}_{\mathbf{k}}^{\mathbf{j}}\right), \text{ y}$$
$$\langle \mathcal{A}, i \rangle \models \left(\left(\exists \mathbf{X}_{\mathbf{k}}^{\mathbf{j}}\right) \Phi\right) \iff \text{existe } X \subseteq |\mathcal{A}|^j \text{ tal que } \langle \mathcal{A}, i, X/\mathbf{X}_{\mathbf{k}}^{\mathbf{j}} \rangle \models \Phi,$$

con la interpretación $\left(i, X/\mathbf{X}_{\mathbf{k}}^{\mathbf{j}}\right)$ definida por

$$\left(i, X/\mathbf{X}_{\mathbf{k}}^{\mathbf{j}}\right)(T) = \begin{cases} i(T) & \text{si } T \neq \mathbf{X}_{\mathbf{k}}^{\mathbf{j}}, \\ X & \text{si } T = \mathbf{X}_{\mathbf{k}}^{\mathbf{j}}. \end{cases}$$

Además,

$$\langle \mathcal{A}, i \rangle \models \left(\left(\forall \mathbf{X}_{\mathbf{k}}^{\mathbf{j}}\right) \Phi\right) \iff \text{para todo } X \subseteq |\mathcal{A}|^j, \langle \mathcal{A}, i, X/\mathbf{X}_{\mathbf{k}}^{\mathbf{j}} \rangle \models \Phi.$$

Si $\langle \mathcal{A}, i \rangle \models \Phi$, diremos que $\langle \mathcal{A}, i \rangle$ *satisface* Φ.

De esta definición extendida sigue que, la disyunción, la conjunción, y la disyunción exclusiva son asociativas y conmutativas (ver observación 2.1). Así, cuando tengamos la disyunción, la conjunción, o la disyunción exclusiva de tres o más fórmulas, omitiremos los paréntesis asociados, pues la definición asegura que podemos restaurar paréntesis en estas expresiones de cualquier manera permitida sin alterar su satisfacción, o no satisfacción.

Dos fórmulas Ψ, Φ en $\text{SO}(\sigma)$ son *lógicamente equivalentes* si

$$\langle \mathcal{A}, i \rangle \models \Psi \iff \langle \mathcal{A}, i \rangle \models \Phi$$

para toda σ-estructura $\mathcal{A}$ y para toda interpretación i en $\mathcal{A}$ que incluye las variables libres relevantes. En este caso escribimos $\Psi \equiv \Phi$. Además, diremos que las fórmulas Ψ, Φ en $\text{SO}(\sigma)$ son *mutuamente excluyentes* si, dadas cualquier σ-estructura $\mathcal{A}$ y cualquier interpretación i en $\mathcal{A}$,

$$\langle \mathcal{A}, i \rangle \models \neg\Psi \vee \neg\Phi.$$

Ahora, dada una fbf de segundo orden con vocabulario σ,

$$\Phi\left(\mathbf{X}_{\mathbf{b_1}}^{\mathbf{d_1}}, \ldots, \mathbf{X}_{\mathbf{b_p}}^{\mathbf{d_p}}, y_{j_1}, \ldots, y_{j_q}\right),$$

y dadas una σ-estructura $\mathcal{A}$, y una interpretación i en $\mathcal{A}$ tal que $i\left(\mathbf{X}_{\mathbf{b_k}}^{\mathbf{d_k}}\right) = X_k$, y tal

que $i\left(y_{j_m}\right) = l_m$, si $\langle \mathcal{A}, i\rangle \models \Phi$, podemos expresar esto con la notación

$$\mathcal{A} \models \Phi\left(X_1, \ldots, X_p, l_1, \ldots, l_q\right).$$

Por otro lado, si $\mathcal{B}$ es una σ-estructura, y Ψ es una fórmula de segundo orden con vocabulario σ tal que, $\langle \mathcal{B}, i\rangle \models \Psi$ para toda interpretación i en $\mathcal{B}$, podemos escribir $\mathcal{B} \models \Psi$. Asimismo, si $\langle \mathcal{B}, i\rangle \not\models \Psi$ para toda interpretación i en $\mathcal{B}$, podemos expresar esto con $\mathcal{B} \not\models \Psi$. La sentencias de segundo orden verifican esta propiedad, es decir, su satisfacción, o no satisfacción, no depende de interpretación alguna.

Definición 2.16 (Modelo)**.** Sean $\mathcal{B}$ una σ-estructura, y Ψ un miembro de $\mathrm{SO}(\sigma)$ tal que, $\langle \mathcal{B}, i\rangle \models \Psi$ para toda interpretación i en $\mathcal{B}$. Entonces $\mathcal{B}$ es llamada un *modelo de* Ψ. Además, un *modelo finito de* Ψ es un modelo de Ψ de tamaño finito. Al conjunto de modelos finitos de Ψ lo notaremos con $\mathrm{MOD}(\Psi)$.

Ahora, tomando la notación de [2], sean

$$\mathrm{FO} = \bigcup_{\sigma} \mathrm{FO}(\sigma) \text{ y}$$
$$\mathrm{SO} = \bigcup_{\sigma} \mathrm{SO}(\sigma).$$

A FO, SO, y otros conjuntos de fórmulas bien formadas los llamaremos *lógicas*. Si $\mathcal{L}$ es una lógica, notaremos con $\mathcal{L}(\sigma)$ al conjunto de fómulas en $\mathcal{L}$ con vocabulario σ.

Definición 2.17 (Forma normal prenexa)**.** Una fórmula Φ en SO está en *forma normal prenexa*, si consiste de una cadena de cuantificadores seguida de una fórmula que no tiene ocurrencias de cuantificadores.

Ejemplo 2.9. Las siguientes fórmulas están en forma normal prenexa.

$$(\forall x)(\forall y)(\forall z)(\neg V^3(x, y, z))$$
$$(\forall x)(\neg V^3(x, y, z))$$
$$(\forall x)(\exists \mathbf{P})(\forall y)(\forall z)(\mathbf{P}(y, x, z) \wedge (\neg V^3(x, y, z)))$$

Es un hecho que toda fórmula $\varphi \in \mathrm{FO}$ es lógicamente equivalente a una fórmula $\psi \in \mathrm{FO}$ en forma normal prenexa (ver el teorema 3.5.11 de [5]). Es más, toda fórmula Φ en SO es lógicamente equivalente a una fórmula en forma normal prenexa, $\Psi \in \mathrm{SO}$, tal que cada cuantificador de segundo orden precede a todos los cuantificadores de primer orden; además, Ψ está en Σ^1_k si la cadena de cuantificadores de segundo orden consiste de k bloques que alternan entre bloques de cuantificadores

existenciales y bloques de cuantificadores universales, con el primer bloque siendo existencial (ver la sección 3.1 de [7]).

Ahora, notaremos con FO$\forall$ al extracto de FO que contiene las fórmulas cuyos cuantificadores aparecen todos al principio y son universales. Definimos SO$\exists$ como Σ^1_1, y SO$\exists\forall$ como Σ^1_2.

La siguiente definición fue tomada de la sección 7.1 de [7].

Definición 2.18. Si $\mathcal{L}_1$ y $\mathcal{L}_2$ son lógicas, escribiremos $\mathcal{L}_1 \leq \mathcal{L}_2$ si, para cada vocabulario σ y para cada sentencia $\Phi \in \mathcal{L}_1(\sigma)$, existe una sentencia $\Psi \in \mathcal{L}_2(\sigma)$ tal que $\mathrm{MOD}(\Phi) = \mathrm{MOD}(\Psi)$.

Con esta definición tenemos

$$\mathrm{FO}\forall \leq \mathrm{FO} \leq \mathrm{SO}\exists \leq \mathrm{SO}\exists\forall \leq \mathrm{SO}; \tag{2.12}$$

basta introducir, en lugares apropiados, cuantificadores que cuantifiquen variables que no aparecen en las respectivas fórmulas. Por ejemplo, si α es una sentencia en FO que está en forma normal prenexa, entonces $(\exists \mathbf{P})\alpha$ es una sentencia lógicamente equivalente que está en SO$\exists$. A su vez, esta última es lógicamente equivalente a $(\exists \mathbf{P})(\forall \mathbf{Q})\alpha$, la cual es una sentencia en SO$\exists\forall$.

Por otro lado, si $\mathcal{L}_1$, $\mathcal{L}_2$, y $\mathcal{L}_3$ son lógicas, entonces

$$\mathcal{L}_1 \leq \mathcal{L}_2 \text{ y } \mathcal{L}_2 \leq \mathcal{L}_3 \implies \mathcal{L}_1 \leq \mathcal{L}_3,$$

es decir, la relación $\leq$ es transitiva.

Finalmente, mencionamos que cuando estemos interpretando una fórmula bien formada de segundo orden, no expondremos todos los detalles que nos llevan desde la fórmula hasta la interpretación. En vez, procederemos directamente como en el siguiente ejemplo.

Ejemplo 2.10. Consideremos la sentencia con vocabulario $\mu = \langle Q^2 \rangle$:

$$(\exists \mathbf{P})(\forall x)(\exists y)(\exists z)(Q^2(x,y) \wedge Q^2(x,z) \wedge \mathbf{P}(y) \wedge \neg \mathbf{P}(z)).$$

Luego, una μ-estructura $\mathcal{A}$ es un modelo de la sentencia si y solo si existe un conjunto $P \subsetneq |\mathcal{A}|$ tal que, para todo $i \in |\mathcal{A}|$, existen $k, l \in |\mathcal{A}|$ con

$$(i,k), (i,l) \in Q^{\mathcal{A}},\ k \in P \text{ y } l \notin P.$$

Esta sentencia aparece en el lema 4.1, en un capítulo posterior.

2.2. Complejidad computacional

La complejidad computacional es una rama de la informática teórica que se encarga del estudio de la cantidad de recursos que usan los algoritmos. Para lograrlo hace uso de modelos matemáticos de cómputo como las máquinas de Turing.

En esta sección describiremos informalmente qué son los problemas de decisión, los programas para máquinas de Turing deterministas y no deterministas, las clases de complejidad P y NP, y las reducciones polinomiales entre problemas de decisión. El contenido está basado fundamentalmente en los capítulos 1 y 2 de [8].

En complejidad computacional, un *problema* se especifica con las descripciones de

1. sus parámetros o variables libres, y
2. la solución.

De esta manera, una *instancia* de un problema se obtiene especificando valores para todos los parámetros.

Ejemplo 2.11. El problema de satisfacibilidad booleana (SAT) se define así (ver la subsección 2.1.1):

Instancia: Un conjunto V de variables proposicionales y una colección C de cláusulas sobre V

Pregunta: ¿Existe una asignación de valores de verdad que satisface C?

En este problema, la solución es una respuesta a la pregunta planteada.

Ahora, un *algoritmo* que resuelve un problema es un procedimiento general que asigna a cada instancia del problema una solución. En el ejemplo anterior, podemos pensar en el método de las tablas de verdad como un algoritmo que resuelve SAT.

Además, la *complejidad* de un algoritmo es la cantidad de recursos, como tiempo y espacio, que se requiere para ejecutar el algoritmo. Aquí nos enfocaremos en la complejidad temporal en el peor de los casos; esta se mide en función del tamaño de los datos de entrada del algoritmo. Esos datos pueden ser especificados con un esquema de codificación que a cada instancia del problema en mano, asigna una cadena de símbolos. Luego, el tamaño de la cadena es el tamaño de los datos de entrada. De esta manera, un algoritmo de tiempo polinomial es uno cuya función de complejidad temporal puede ser acotada por un polinomio.

Entre los problemas estudiados por la complejidad computacional están los *problemas de decisión*. Estos son problemas con la propiedad de que la solución es una

respuesta “Sí”, o “No”, a una pregunta formulada en términos de los parámetros del problema. SAT (ver ejemplo 2.11) es un ejemplo de este tipo de problemas.

Dado un problema de decisión, tenemos las instancias cuya respuesta es “Sí” y aquellas cuya respuesta es “No”. Las primeras son las *instancias positivas* del problema, mientras las restantes son las *instancias negativas*.

Ahora, una *clase de complejidad computacional* puede definirse como una colección de problemas de decisión que comparten propiedades relacionadas justamente con la complejidad de algoritmos que los resuelven. Aquí estamos interesados en definir las clases P y NP. Para definir la primera, tomamos el modelo de cómputo de la *máquina de Turing determinista de una cinta (MTD)*. Esta está constituida por algunos elementos, en particular, por una cinta con un número infinito de celdas numeradas con los enteros, y un cabezal que puede leer el contenido de las celdas, así como escribir en ellas.

Un *programa M para una máquina determinista de Turing de una cinta* es una función matemática preestablecida a través de la cual, dada una cadena finita de símbolos permitidos en la cinta, instruye a la máquina cada paso a seguir. Estos programas son versiones formales del concepto de algoritmo. Para medir la complejidad temporal de uno de ellos se toma como unidad de tiempo cada instrucción ejecutada. Además, dada cualquier cadena finita x como entrada, el programa M siempre termina. Si termina en un estado específico, al que podemos notar con q_s, decimos que el programa *M acepta x*.

Dado un problema de decisión Π, y un esquema que codifica las instancias de Π como cadenas finitas de símbolos permitidos, un programa M para MTD resuelve Π si y solo si el conjunto de codificaciones de instancias positivas de Π es igual al conjunto de cadenas que M acepta. De esta manera, la clase de complejidad P puede definirse como el conjunto de problemas de decisión para los cuales existen programas para MTD de tiempo polinomial que los resuelven.

Ahora, un programa M para MTD también calcula una función, f_M, pues, dada una cadena de entrada x, se puede tomar como $f_M(x)$ la cadena escrita en la cinta una vez que la máquina para.

Cualquier cálculo que una computadora ordinaria puede hacer, puede ser realizado por un programa para MTD. Los algoritmos para computadora que no son de tiempo polinomial son considerados ineficientes pues, cuando el tamaño de los datos de entrada es muy grande, estos algoritmos toman mucho tiempo en terminar de manera que no resultan prácticos en la mayoría de los casos. Por otro lado, existen

otros modelos de cómputo a través de los cuales se puede definir P.

Para definir la clase de complejidad NP, introducimos la *máquina de Turing no determinista de una cinta (MTND)*. Esta tiene la misma estructura que una MTD y, adicionalmente, un módulo de adivinanza y un cabezal de solo escritura.

Un *programa para una máquina de Turing no determinista de una cinta* se especifica de manera similar que uno para MTD, es decir, con una función matemática, pero opera en dos etapas: la de adivinanza y la de verificación. Dada una instancia I de un problema de decisión Π, la etapa de adivinanza provee una estructura, S, y la de verificación comprueba de manera determinista que la respuesta es "Sí" para I y S. Específicamente, bajo un esquema de codificación para Π, dada la codificación x de I, el módulo de adivinanza decide en cada paso si el cabezal de solo escritura escribe un símbolo de entre los permitidos en la celda actual y pasa a la siguiente, o si para; una vez que este módulo para (si para), este deja de funcionar y entra en operación el de verificación, y el resto del proceso es como aquel para una MTD. Hacemos notar que el programa podría retornar "No" o no terminar.

En este caso, se dice que un cómputo acepta si termina en el estado q_s, y se dice que el programa M acepta x si M realiza un cómputo que acepta. Ahora, el tiempo que toma un programa M en aceptar x es el mínimo número de pasos que toman la etapa de adivinanza y la de verificación hasta llegar a q_s. El programa M es de tiempo polinomial si existe un polinomio que acote la función de complejidad temporal. Así, NP puede ser definida como la clase de problemas de decisión Π tales que existen programas M para MTND de tiempo polinomial, con el conjunto de codificaciones de instancias positivas de Π igual al conjunto de cadenas x que M acepta. Es un hecho bien conocido que $\mathrm{P} \subseteq \mathrm{NP}$: si Π es un problema en P y M es un programa para MTD de tiempo polinomial que lo resuelve, se puede usar este mismo programa en una MTND en la etapa de verificación, e ignorar la de adivinanza.

Por ejemplo, en el caso de SAT (ver el ejemplo 2.11), podemos tomar un programa M que adivine una asignación de valores de verdad, y tal que la de verificación compruebe que la asignación satisface el conjunto de cláusulas.

Ahora, informalmente, una reducción de un problema de decisión Π_1 a un problema de decisión Π_2 es una aplicación que transforma cada instancia de Π_1 en una instancia equivalente de Π_2. Así, si existe un algoritmo eficiente que resuelva Π_2, y el cómputo que hace la transformación también es eficiente, se tiene que Π_1 se puede resolver de manera eficiente concatenando el cómputo de la transformación con el algoritmo.

Para ser más precisos, una *reducción polinomial desde* Π_1 *a* Π_2 es una función $f : \Sigma_1^* \to \Sigma_2^*$ que puede ser calculada por un programa para MTD de tiempo polinomial, y tal que

$$x \in L_1 \subset \Sigma_1^* \iff f(x) \in L_2 \subset \Sigma_2^*,$$

con L_i el conjunto de codificaciones de instancias positivas de Π_i, y Σ_i^* el conjunto de cadenas finitas de símbolos permitidos. En caso de que exista una reducción polinomial de Π_1 a Π_2, escribiremos $\Pi_1 \leq_p \Pi_2$.

Un problema Π es C-*hard vía reducciones polinomiales* si, para todo problema Π' en la clase de complejidad C, $\Pi' \leq_p \Pi$. Si, adicionalmente, Π pertenece a C, Π es llamado C-*completo vía reducciones polinomiales*. SAT es NP-completo en este sentido. Además, mencionamos que en la clase NP, los problemas en P son considerados los más fáciles, mientras que los NP-completos los más difíciles.

La relación $\leq_p$ es transitiva. Si Π_1 es NP-hard vía reducciones polinomiales y $\Pi_1 \leq_p \Pi_2$, entonces Π_2 es NP-hard vía reducciones polinomiales.

2.3. Teoría descriptiva de la complejidad computacional

Varios modelos de cómputo, como las máquinas de Turing o el cálculo lambda, fueron desarrollados de manera independiente. Sin embargo, si se mide de manera apropiada la complejidad de un cálculo en cualquiera de estos modelos, se obtendrá valores equivalentes. La teoría descriptiva de la complejidad computacional ofrece, desde un punto de vista diferente, una teoría asimismo equivalente. Las herramientas fundamentales de esta provienen de la lógica matemática. Por ejemplo, los problemas de decisión suelen ser definidos a través de sentencias lógicas, y las instancias de un problema son estructuras derivadas de algún vocabulario. Con esta teoría se ha podido caracterizar clases de complejidad computacional importantes a través de lenguajes lógicos independientes de cualquier modelo de cómputo. Además, existen métodos de la lógica que permiten deducir qué puede ser expresado, y qué no, en un lenguaje dado.

El contenido de esta sección está basado en los capítulos 1, 2, 3, y 7 de [9], y en el artículo [2]. Para definir conceptos, se hará uso de la máquina de Turing multicinta; a diferencia de las máquinas de Turing de una cinta, estas poseen varias cintas. Los conceptos de la anterior sección no sufren modificaciones sustanciales al hacer uso de las máquinas de Turing multicinta.

Como se ha mencionado, en la teoría descriptiva las instancias de un problema corresponden a estructuras con algún vocabulario adecuado. De aquí en adelante consideraremos vocabularios relacionales (ver definición 2.10) que poseen el símbolo de relación binaria $=$. Este siempre será interpretado como la igualdad. Por otra parte asumiremos que, dado un vocabulario τ, si $\mathcal{A} \in \text{STRUC}[\tau]$, entonces $\mathcal{A}$ es de tamaño finito.

Ahora, notemos que cuando se introduce datos en una computadora, se impone un orden sobre los datos: los datos son cadenas de símbolos y, entonces, tenemos el primer símbolo de la cadena, el segundo, etcétera. Por ejemplo, si bien existen situaciones en las que los vértices de un grafo no necesitan estar ordenados, al introducir el grafo como dato en la computadora se impone un orden. Así, dado un vocabulario τ, asumiremos que el universo de las τ-estructuras está dotado de un orden total: τ incluirá el símbolo de relación binaria $\leq$; este siempre será interpretado como ese orden total. De esta manera, dada una τ-estructura $\mathcal{A}$, existe una biyección entre $|\mathcal{A}|$ y el conjunto $\{0, 1, \ldots, n-1\}$ cuando $\|\mathcal{A}\| = n$. Adicionalmente, los vocabularios incluirán los símbolos de constante $0, 1$, y max, que serán interpretados como el mínimo, el segundo, y el máximo elemento, respectivamente, engendrados por el orden total.

Algunas relaciones numéricas son útiles: usaremos los símbolos de relación binaria SUC y BIT, con $\mathcal{A} \models \text{SUC}(i, j)$ si y solo si j es el sucesor de i, y con $\mathcal{A} \models \text{BIT}(k, j)$ si y solo si el j-ésimo bit en la representación binaria de k es un 1 (el 0-ésimo bit es el menos significativo). Todos los vocabularios que consideremos incluirán estos símbolos.

Los símbolos $=, \leq, \text{SUC},$ y BIT serán llamados *símbolos de relaciones numéricas*, mientras $0, 1,$ y max, serán llamados *símbolos de constantes numéricas*. Dado un vocabulario τ, no especificaremos estos símbolos en la tupla que lo define. Al resto de símbolos de τ los llamaremos *símbolos de relación de entrada* y *símbolos de constante de entrada*.

En el contexto de lógica de primer orden, las fórmulas atómicas y las negaciones de fórmulas atómicas serán llamadas *literales*. Una *cláusula* será una disyunción $L_1 \vee \ldots \vee L_k$ de literales. Dado un vocabulario σ, un *σ-literal* será un literal cuyo símbolo de relación asociado está en σ (no es un símbolo de relación numérica). Por otro lado, una *fórmula numérica* será una fórmula que no tiene ocurrencias de símbolos de relación de entrada. Una fórmula de primer orden que no tiene cuantificadores está en *forma normal conjuntiva (FNC)* si es una conjunción de cláusulas. Con $FNC_r(\tau)$

notaremos a la clase de fórmulas de primer orden en FNC, con vocabulario τ, cuyas cláusulas tienen a lo mucho r τ-literales.

Asumiremos que $\|\mathcal{A}\| \geq 2$ para toda estructura $\mathcal{A}$. Se hace para evitar que la fórmula $0 = 1$ sea satisfecha. Finalmente, dada una sentencia Ψ, redefinimos $\text{MOD}(\Psi)$ (ver definición 2.16) tomando en cuenta las consideraciones hechas en esta sección; en particular, $\text{MOD}(\Psi)$ estará constituido por estructuras finitas ordenadas.

Ahora, recordemos el ejemplo 2.6. Ahí definimos τ_g, el vocabulario de los grafos. Con lo expuesto, τ_g incluye los símbolos de relaciones y de constantes numéricas; sin embargo, escribiremos $\tau_g = \langle E^2 \rangle$; luego, E^2 es el único símbolo de relación de entrada en τ_g. Podemos pensar en cada τ_g-estructura como una instancia de un grafo. Asimismo, podemos pensar en cada estructura con el vocabulario σ del ejemplo 2.5, como una instancia de SAT (ver el ejemplo 2.11).

Definición 2.19 (Query)**.** Una *query* es una aplicación $I : \text{STRUC}[\sigma] \to \text{STRUC}[\tau]$ tal que existe un polinomio p, con $\|I(\mathcal{A})\| \leq p\left(\|\mathcal{A}\|\right)$ para toda $\mathcal{A} \in \text{STRUC}[\sigma]$.

Definición 2.20 (Query booleana)**.** Una *query booleana* es una aplicación $I_b : \text{STRUC}[\sigma] \to \{0, 1\}$. Alternativamente, es el conjunto de todas las $\mathcal{A} \in \text{STRUC}[\sigma]$ tales que $I_b\left(\mathcal{A}\right) = 1$. I_Φ es una query booleana asociada a una sentencia Φ si

$$I_\Phi\left(\mathcal{A}\right) = 1 \iff \mathcal{A} \models \Phi.$$

De esta manera, $\text{MOD}(\Phi)$ es una query booleana.

Ejemplo 2.12. Podemos definir el problema de decisión SAT (ver los ejemplos 2.11 y 2.5) a través de la sentencia

$$\Phi_{SAT} = (\exists \mathbf{S})(\forall x)(\exists y)((P(x, y) \wedge \mathbf{S}(y)) \vee (N(x, y) \wedge \neg \mathbf{S}(y))).$$

Recordemos que una σ-estructura $\mathcal{A}$ de tamaño n, tiene asociados el conjunto de n variables proposicionales $V = \{x_0, x_1, \ldots, x_{n-1}\}$ y la tupla de n cláusulas $\langle Y_0, Y_1, \ldots, Y_{n-1} \rangle$. Además, $(k, l) \in P^{\mathcal{A}}$ si y solo si $x_l \in Y_k$, y $(j, m) \in N^{\mathcal{A}}$ si y solo si $\neg x_m \in Y_j$.

Para toda σ-estructura $\mathcal{A}$, $\mathcal{A} \models \Phi_{SAT}$ si y solo si existe $S \subseteq |\mathcal{A}|$ tal que, para todo $k \in |\mathcal{A}|$, existe $j \in |\mathcal{A}|$ tal que $(k, j) \in P^{\mathcal{A}}$ y $j \in S$, o tal que $(k, j) \in N^{\mathcal{A}}$ y $j \notin S$. Ahora, si definimos la asignación de valores de verdad $M = \{x_i \in V \,|\, i \in S\}$, tenemos que $\mathcal{A} \models \Phi_{SAT}$ implica que existe $M \subseteq V$ tal que, para toda cláusula Y_k en la tupla, existe $x_j \in V$ tal que $x_j \in Y_k$ y $x_j \in M$, o tal que $\neg x_j \in Y_k$ y $x_j \notin M$; es decir, $\mathcal{A} \models \Phi_{SAT}$ implica que existe una asignación de valores de verdad que satisface

todas las cláusulas de la tupla asociada a $\mathcal{A}$ (ver observación 2.2).

Recíprocamente, si existe una asignación de valores de verdad M que satisface todas las cláusulas de la tupla asociada a $\mathcal{A}$, entonces, para toda cláusula Y_k en la tupla, existe al menos un literal en Y_k que es satisfecho por M. Es decir, para toda cláusula Y_k en la tupla, existe $x_j \in V$ tal que $x_j \in Y_k$ y $x_j \in M$, o tal que $\neg x_j \in Y_k$ y $x_j \notin M$. Definiendo $S = \{i \in |\mathcal{A}|| \, x_i \in M\}$, vemos que $\mathcal{A} \models \Phi_{SAT}$.

Así, podemos definir SAT como MOD (Φ_{SAT}).

A continuación conectaremos la teoría descriptiva de la complejidad computacional con la contraparte que hace uso de modelos de cómputo. En primer lugar, se usará la notación $M(w) \downarrow$ para indicar que el programa M para máquina de Turing acepta la entrada w. Con esto definimos

$$L(M) = \left\{w \in \{0,1\}^* : M(w) \downarrow\right\},$$

es decir, $L(M)$ es el conjunto de cadenas binarias finitas que M acepta (aquí se está asumiendo que los datos de entrada para las máquina de Turing son cadenas binarias finitas). Además, usaremos la notación $T(w)$ para la cadena que la máquina de Turing T ha dejado sobre la cinta de salida una vez que ha parado, si para; si no para, $T(w)$ no está definida. ¿Cómo codificar estructuras de manera que sean alimentadas como datos en una máquina de Turing? Consideremos el siguiente ejemplo.

Ejemplo 2.13. Sean

$$\tau = \langle R_1^2, c_1 \rangle$$

un vocabulario relacional, y $\mathcal{A}$ una τ-estructura de tamaño 3, es decir, $|\mathcal{A}| = \{0,1,2\}$. Adicionalmente supongamos que

$$R_1^{\mathcal{A}} = \{(0,1),(1,2),(2,2)\},$$

y que $c_1^{\mathcal{A}} = 2$. Ahora, $R_1^{\mathcal{A}} \subset |\mathcal{A}|^2$, y los elementos de $|\mathcal{A}|^2$ pueden ser ordenados de acuerdo al orden lexicográfico. Luego, podemos codificar $R_1^{\mathcal{A}}$ con una cadena binaria de longitud $9 = 3^2 = \|\mathcal{A}\|^2$, con el j-ésimo bit igual 1 si y solo si el j-ésimo elemento de $|\mathcal{A}|^2$ es miembro de $R_1^{\mathcal{A}}$. De esta manera, la codificación binaria de $R_1^{\mathcal{A}}$ es 010001001.

Por otro lado, podemos codificar $c_1^{\mathcal{A}}$ con la cadena binaria 10, es decir, con su representación en binario; notar que la cadena tiene longitud $2 = \lceil \log_2 3 \rceil = \lceil \log_2 \|\mathcal{A}\| \rceil$.

Así, si concatenamos las cadenas, obtenemos la codificación binaria de $\mathcal{A}$, en este caso 01000100110.

Para ser precisos, dados un vocabulario relacional

$$\tau = \langle R_1^{a_1}, \ldots, R_r^{a_r}, c_1, \ldots, c_s \rangle,$$

y una τ-estructura $\mathcal{A}$ de tamaño n, se define $bin^{\mathcal{A}}(R_i)$ como una cadena binaria de longitud n^{a_i}, con el j-ésimo bit igual 1 si y solo si, en el orden lexicográfico, el j-ésimo elemento de $|\mathcal{A}|^{a_i}$ es miembro de $R_i^{\mathcal{A}}$. Además, se define $bin^{\mathcal{A}}(c_j)$ como la cadena binaria de longitud $\lceil \log_2 n \rceil$, que es la representación binaria de $c_j^{\mathcal{A}}$. Así, la codificación binaria de $\mathcal{A}$ es

$$bin(\mathcal{A}) = bin^{\mathcal{A}}(R_1) \ \ldots \ bin^{\mathcal{A}}(R_r)\, bin^{\mathcal{A}}(c_1) \ \ldots \ bin^{\mathcal{A}}(c_s).$$

La longitud de esta cadena es $n^{a_1} + \ldots + n^{a_r} + s\lceil \log_2 n \rceil$, un número relacionado polinomialmente con n. Notar que en $bin(\mathcal{A})$ no se toma en cuenta las relaciones o constantes numéricas, pues estas pueden ser calculadas fácilmente.

Ahora, sea $I : \text{STRUC}[\sigma] \to \text{STRUC}[\tau]$ una query. Diremos que la máquina de Turing T calcula I si y solo si para toda $\mathcal{A} \in \text{STRUC}[\sigma]$, $T\,(bin(\mathcal{A})) = bin\,(I(\mathcal{A}))$. Definimos DTIME$[t(n)]$ como la clase de queries booleanas que pueden ser calculadas por una máquina determinista de Turing multicinta en $O(t(n))$ pasos (n es el tamaño de las estructuras). Asimismo, definimos s NTIME$[t(n)]$ como la clase de queries booleanas que pueden ser calculadas por una máquina de Turing no determinista en $O(t(n))$ pasos. Luego,

$$\text{P} = \bigcup_{t \in \mathcal{F}} \text{DTIME}[t(n)] \text{ y}$$
$$\text{NP} = \bigcup_{t \in \mathcal{F}} \text{NTIME}[t(n)],$$

con $\mathcal{F}$ el conjunto de los polinomios.

El siguiente teorema dio inicio a la teoría descriptiva. Este da una caracterización de la clase de complejidad NP independiente de modelos de cómputo.

Teorema 2.1 (Teorema de Fagin)**.** *NP es igual al conjunto de queries booleanas asociadas a sentencias existenciales de segundo orden.*

Recordemos el ejemplo 2.12. La sentencia Φ_{SAT} está en el fragmento existencial de la lógica de segundo orden, es decir, en SO$\exists$. Así, decimos que Φ_{SAT} *define* SAT *en* SO$\exists$, y que SAT *puede ser definido* en SO$\exists$.

Para una lógica $\mathcal{L}$ y una clase de complejidad C, usaremos la notación $\text{C} \leq \mathcal{L}$ si todo problema en C puede ser definido en $\mathcal{L}$. Además, si MOD(Φ) es un problema en C

para cada sentencia Φ en $\mathcal{L}$, escribiremos $\mathcal{L} \leq$ C. Si C $\leq \mathcal{L}$ y $\mathcal{L} \leq$ C son ciertas, decimos que la lógica $\mathcal{L}$ *captura* C y escribimos $\mathcal{L} =$ C. Así, el teorema 2.1 puede ser expresado con SO$\exists$ = NP.

Ahora, nuestro interés es definir la jerarquía polinomial. Esta es una jerarquía de clases de complejidad computacional. Para hacerlo, introducimos el concepto de oráculo. Un *programa para máquina de Turing con oráculo para una query booleana* B, M^{B}, es un programa para una máquina de Turing multicinta que tiene una cinta especial a la que puede acceder en cualquier instante durante su ejecución, tal que, si la cadena $bin(\mathcal{A})$ es escrita sobre la cinta, la máquina retornará 1, en tiempo unitario, si y solo si $\mathcal{A} \in$ B.

Definición 2.21 (Jerarquía polinomial)**.** Sea $\Sigma_0^p =$P. Definimos, para $i \geq 0$,

$$\Sigma_{i+1}^p = \left\{ L\left(M^{\mathrm{B}}\right) : M \text{ es un programa para máquina de Turing no determinista con oráculo para B, B} \in \Sigma_i^p \right\}.$$

Así, Σ_i^p es llamado *el nivel i de la jerarquía polinomial*. La *jerarquía polinomial* es definida como la clase

$$\mathrm{PH} = \bigcup_{i=1}^{\infty} \Sigma_i^p.$$

De la anterior definición sigue que $\Sigma_1^p =$ NP.

El teorema a continuación da una caracterización, independiente de modelos de cómputo, de cada nivel de la jerarquía polinomial.

Teorema 2.2. *Sean $i > 0$ y $S \subset STRUC[\sigma]$ una query booleana. Entonces las siguientes condiciones son equivalentes.*

- $S \in \Sigma_i^p$
- $S = MOD(\Phi)$ *para alguna sentencia de segundo orden Φ con todos los cuantificadores de segundo orden al inicio de ella, específicamente con el prefijo*

$$\exists \overline{R_1} \forall \overline{R_2} \ldots Q_i \overline{R_i},$$

donde $Q_k \overline{R_k} = Q_k \mathbf{X}_{\mathbf{k_1}}^{\mathbf{b_1}} \ldots Q_k \mathbf{X}_{\mathbf{k_j}}^{\mathbf{b_j}}$ para algún j, con Q_k el cuantificador existencial si k es impar, y el universal si es par.

Con esta definición, SO$\exists\forall = \Sigma_2^p$.

A continuación introducimos el concepto de reducciones en el contexto de la teoría descriptiva. Para lograrlo, primero establecemos la siguiente definición.

Definición 2.22 (Query de primer orden)**.** Sean σ, τ dos vocabularios con

$$\tau = \langle R_1^{a_1}, \ldots, R_r^{a_r}, c_1, \ldots, c_s \rangle.$$

Sea $k \geq 1$ un entero. Una *query de primer orden k-aria de* STRUC$[\sigma]$ *en* STRUC$[\tau]$ es una aplicación I : STRUC$[\sigma] \to$ STRUC$[\tau]$ definida por una tupla de fórmulas en FO(σ),

$$\langle \varphi_0, \varphi_1, \ldots, \varphi_r, \psi_1, \ldots, \psi_s \rangle.$$

Para toda $\mathcal{A} \in$ STRUC$[\sigma]$, esta tupla de fórmulas define una τ-estructura

$$I(\mathcal{A}) = \langle |I(\mathcal{A})|, R_1^{I(\mathcal{A})}, \ldots, R_r^{I(\mathcal{A})}, c_1^{I(\mathcal{A})}, \ldots, c_s^{I(\mathcal{A})} \rangle,$$

con

$$|I(\mathcal{A})| = \left\{ (b_1, \ldots, b_k) \in |\mathcal{A}|^k : \mathcal{A} \models \varphi_0(b_1, \ldots, b_k) \right\},$$

$R_i^{I(\mathcal{A})}$ igual a

$$\left\{ \left(b_1^1, \ldots, b_1^k, \ldots, b_{a_i}^1, \ldots, b_{a_i}^k \right) \in |I(\mathcal{A})|^{a_i} : \mathcal{A} \models \varphi_i \left(b_1^1, \ldots, b_1^k, \ldots, b_{a_i}^1, \ldots, b_{a_i}^k \right) \right\},$$

para $i = 1, \ldots, r$, y con $c_j^{I(\mathcal{A})}$ el único $(b_1, \ldots, b_k) \in |I(\mathcal{A})|$ tal que $\mathcal{A} \models \psi_j(b_1, \ldots, b_k)$, esto para $j = 1, \ldots, s$.

El conjunto de queries de primer orden es cerrado bajo la operación de composición. Además hacemos notar que, si I es una query de primer orden k-aria con dominio STRUC$[\sigma]$, y $\varphi_0 \equiv$ **true**, entonces $|I(\mathcal{A})| = |\mathcal{A}|^k$ para toda $\mathcal{A} \in$ STRUC$[\sigma]$.

Ahora, sean σ, τ dos vocabularios. Recordemos que estos incluyen los símbolos de relaciones y de constantes numéricas a pesar de que no consten en las tuplas respectivas. Así, una query de primer orden k-aria I : STRUC$[\sigma] \to$ STRUC$[\tau]$ debe incluir fórmulas en FO(σ) que definan, para toda $\mathcal{A} \in$ STRUC$[\sigma]$, las relaciones y constantes numéricas en $I(\mathcal{A})$. Si $\varphi_0 \equiv$ **true**, siempre se puede hacer esto. Por ejemplo, es claro que se puede establecer fórmulas de manera que

$$\begin{aligned} 0^{I(\mathcal{A})} &= (0^{\mathcal{A}}, \ldots, 0^{\mathcal{A}}), \\ 1^{I(\mathcal{A})} &= (0^{\mathcal{A}}, \ldots, 1^{\mathcal{A}}), \text{ y} \\ max^{I(\mathcal{A})} &= (max^{\mathcal{A}}, \ldots, max^{\mathcal{A}}), \end{aligned}$$

donde las tuplas son k-tuplas. En efecto, la constante numérica $0^{I(\mathcal{A})}$ puede ser definida a través de la fórmula $\psi(x_1, \ldots x_k)$ igual a

$$x_1 = 0 \wedge \ldots \wedge x_k = 0.$$

Además, se puede definir la relación correspondiente al símbolo $\leq$ de manera que se imponga el orden lexicográfico en $|I(\mathcal{A})|$ (ver el capítulo 1 de [9]). Por otro lado, se puede definir la relación $\mathrm{SUC}^{I(\mathcal{A})}$ por la fórmula $\varphi\left(x_1, \ldots x_k, y_1, \ldots y_k\right)$ igual a

$$\bigvee_{i=1}^{k}\left(\left(\bigwedge_{j<i} x_j = y_j\right) \wedge \mathrm{SUC}\left(x_i, y_i\right) \wedge \left(\bigwedge_{m=i+1}^{k} x_m = max \wedge y_m = 0\right)\right).$$

Definición 2.23 (Reducción de primer orden)**.** Sean A y B dos queries booleanas, con $\mathrm{A} \subset \mathrm{STRUC}[\sigma]$ y $\mathrm{B} \subset \mathrm{STRUC}[\tau]$. Una *reducción de primer orden de* A *en* B es una query de primer orden $I : \mathrm{STRUC}[\sigma] \to \mathrm{STRUC}[\tau]$ tal que, para toda $\mathcal{A} \in \mathrm{STRUC}[\sigma]$,

$$\mathcal{A} \in \mathrm{A} \iff I(\mathcal{A}) \in \mathrm{B}.$$

En este caso usaremos la notación $\mathrm{A} \leq_{fo} \mathrm{B}$.

Si B es una query booleana, y $\mathrm{A} \leq_{fo} \mathrm{B}$ para toda query booleana A en la clase de complejidad C, decimos que B es C-*hard vía reducciones de primer orden*. Si adicionalmente B está en C, diremos que B es C-*completo vía reducciones de primer orden*. SAT (ver el ejemplo 2.12) es NP-completo vía reducciones de primer orden.

En la teoría descriptiva es posible formalizar el concepto de reducción polinomial (ver la sección anterior). Es un hecho que si un problema es completo vía reducciones de primer orden, entonces es completo vía reducciones polinomiales.

Ahora, existe un tipo de reducciones muy débiles en el sentido de poseer muy poco poder computacional. Estas son de hecho un tipo particular de reducciones de primer orden. A continuación su definición.

Definición 2.24 (Proyecciones de primer orden)**.** Sean σ, τ dos vocabularios con

$$\tau = \langle R_1^{a_1}, \ldots, R_r^{a_r}, c_1, \ldots, c_s \rangle.$$

Adicionalmente, sean A y B dos queries booleanas, con $\mathrm{A} \subset \mathrm{STRUC}[\sigma]$ y $\mathrm{B} \subset \mathrm{STRUC}[\tau]$. Una *proyección de primer orden de* A *en* B es una reducción de primer orden $\rho : \mathrm{STRUC}[\sigma] \to \mathrm{STRUC}[\tau]$ con la tupla asociada

$$\langle \varphi_0, \varphi_1, \ldots, \varphi_r, \psi_1, \ldots, \psi_s \rangle$$

tal que φ_0 es una fórmula numérica, y tal que, para todo $i \geq 1$ y para todo j, φ_i y ψ_j son lógicamente equivalentes a fórmulas

$$\alpha_0 \vee (\alpha_1 \wedge \lambda_1) \vee \ldots \vee (\alpha_e \wedge \lambda_e), \tag{2.13}$$

donde las subfórmulas α_k son numéricas y mutuamente excluyentes dos a dos, y donde las subfórmulas λ_k son literales, es decir, son fórmulas atómicas o negaciones de fórmulas atómicas. Así, si $\mathcal{A} \in \text{STRUC}[\sigma]$ e i es una interpretación en $\mathcal{A}$ que incluye las variables libres relevantes, la fórmula 2.13 será satisfecha por $\langle \mathcal{A}, i \rangle$ si y solo si, o bien $\langle \mathcal{A}, i \rangle$ satisface α_0, o bien satisface α_k y λ_k para exactamente un $k \geq 1$. De esta manera, cualquier bit en $bin\,(\rho(\mathcal{A}))$ depende a lo mucho de un bit de $bin(\mathcal{A})$. Es en este sentido que estas reducciones poseen muy poco poder computacional.

Si existe una proyección de primer orden de A en B, escribiremos $A \leq_{fop} B$ y diremos que *A se reduce a B vía fops* (fops es el acrónimo de la traducción al inglés de "proyecciones de primer orden"). Las fórmulas del tipo 2.13 serán llamadas *fórmulas proyectivas*.

Ejemplo 2.14. Sea $\sigma = \langle P^2 \rangle$. Entonces $\varphi(x, y) = \textbf{false} \vee \big(\textbf{true} \wedge P^2(x, y)\big)$ es una fórmula proyectiva con $\alpha_0 = \textbf{false}$, $\alpha_1 = \textbf{true}$, y $\lambda_1 = P^2(x, y)$. Además, $\varphi(x, y) \equiv P^2(x, y)$, de manera que podemos considerar a $P^2(x, y)$ como una fórmula proyectiva.

Si C es una clase de complejidad, y la query booleana B es tal que, para toda query booleana A en C, $A \leq_{fop} B$, diremos que B es C-*hard vía fops*. Si adicionalmente B está en C, diremos que B es C-*completo vía fops*. SAT (ver ejemplo 2.12) es NP-completo vía fops.

La relación $\leq_{fop}$ es transitiva. Así, si A es C-hard vía fops y $A \leq_{fop} B$, entonces B es C-hard vía fops. En la proposición B.3 se ilustra una proyección de primer orden.

En la sección 4.1 usaremos un operador para probar NP-completitud del problema de decisión correspondiente. Necesitamos las siguientes definiciones. Estas han sido tomadas del capítulo IV de [1].

Definición 2.25 (Operador sintáctico)**.** Sean $\mathcal{L}$ una lógica, y σ y τ dos vocabularios. Un *operador sintáctico* definido sobre $\mathcal{L}(\sigma)$ es una función $T : \mathcal{L}(\sigma) \to \mathcal{L}(\tau)$ (toma fórmulas bien formadas de $\mathcal{L}(\sigma)$ y las transforma en fórmulas bien formadas de $\mathcal{L}(\tau)$).

Definición 2.26 (Operador sintáctico acotado)**.** Un operador sintáctico $T : \mathcal{L}(\sigma) \to \mathcal{L}(\tau)$ es *acotado* si la imagen de toda sentencia en $\mathcal{L}(\sigma)$ es una sentencia en $\mathcal{L}(\tau)$, y la preimagen de toda sentencia en $\mathcal{L}(\tau)$ es una sentencia en $\mathcal{L}(\sigma)$.

A continuación, con $\Psi\,(\theta_1/\varphi_1, \ldots, \theta_m/\varphi_m)$ notamos a la fórmula que resulta de reemplazar cada ocurrencia de la fórmula θ_i en Ψ por la fórmula φ_i, para $1 \leq i \leq m$. Además, con $\bar{x}$ notamos a una tupla de variables

Definición 2.27 (Operador de sustitución de predicados)**.** Un operador sintáctico $T : \mathcal{L}(\sigma) \to \mathcal{L}(\tau)$ es de *sustitución de predicados* si

$$T\Psi = \Psi\left(P_1^{a_1}(\bar{x}_1)/\varphi_1(\bar{x}_1), \ldots, P_m^{a_m}(\bar{x}_m)/\varphi_m(\bar{x}_m)\right),$$

con $P_i^{a_i}$ en σ y $\varphi_i\left(x_1^i, \ldots, x_{a_i}^i\right)$ en $\mathrm{FO}(\tau)$ para $1 \leq i \leq m$.

Ejemplo 2.15. Si $\mathcal{L} = \mathrm{SO}$, $\sigma = \langle P^2 \rangle$ y $\tau = \langle Q^2 \rangle$, entonces el siguiente es un operador de sustitución de predicados:

$$T\Psi = \Psi\left(P(x,y)/Q(x,y)\right).$$

Es más, este operador es acotado. En efecto, sea Ψ una sentencia en $\mathrm{SO}(\sigma)$. Si Ψ no tiene ocurrencias de P, entonces $T\Psi$ será Ψ, es decir, una sentencia en $\mathrm{SO}(\tau)$ que no contienen ocurrencias de Q. Si Ψ tiene ocurrencias de P, cada ocurrencia de $P(x,y)$ en Ψ será reemplazada, a través de T, por $Q(x,y)$, pero todas las variables permanecerán cuantificadas. Luego, $T\Psi$ es una sentencia de $\mathrm{SO}(\tau)$.

Ahora, sea Φ es una sentencia en $\mathrm{SO}(\tau)$. Si Φ no tiene ocurrencias de Q, la preimagen de Φ no puede tener como miembros a fórmulas que tengan ocurrencias de P pues, en caso contrario, Φ tendría al menos una ocurrencia de Q. Así, en este caso, si Ψ está en la preimagen, $T\Psi = \Psi = \Phi$. De esta manera, la preimagen de Φ es el conjunto con miembro único Φ. Por otro lado, si Φ tiene ocurrencias de Q, todas las fórmulas en su preimagen deben tener ocurrencias de P. Para ser concretos, debe ser el conjunto unitario que contiene la sentencia $\Phi\left(Q(x,y)/P(x,y)\right)$.

Definición 2.28 (Operador que sustenta completitud)**.** Sea C una clase de complejidad capturada por la lógica $\mathcal{L}$. Entonces el operador sintáctico acotado $T : \mathcal{L}(\sigma) \to \mathcal{L}(\tau)$ *sustenta* C*-completitud* si, para toda sentencia Ψ en $\mathcal{L}(\sigma)$,

$$\mathrm{MOD}(T\Psi) \text{ es C-completo vía fops} \implies \mathrm{MOD}(\Psi) \text{ es C-competo vía fops.}$$

La siguiente es la proposición 29 en [1].

Proposición 2.2. *Si $T : \mathcal{L}(\sigma) \to \mathcal{L}(\tau)$ es un operador que sustituye predicados por fórmulas proyectivas, entonces T sustenta completitud.*

2.3.1. Superfluidad

La siguiente es una generalización de la definición 3.1 en [2]. A grosso modo, nos servirá de la siguiente manera: si C es una de las clases de complejidad mencionadas en el corolario 2.2, ψ es una sentencia en FO$\forall$, y Φ es una sentencia de una lógica

adecuada, luego

$$\text{MOD}(\Phi \wedge \psi) \text{ es C-hard vía fops } \implies \text{ MOD}(\Phi) \text{ es C-hard vía fops;}$$

es decir, con el mencionado corolario podemos probar C-hardness de MOD(Φ), si conocemos que MOD($\Phi \wedge \psi$) define un problema C-hard.

Definición 2.29 (Superfluidad)**.** Sean σ, τ dos vocabularios, C una clase de complejidad capturada por $\mathcal{L}^*$, y $\mathcal{L}$ una lógica. Entonces

1. Una sentencia Ψ en $\mathcal{L}(\tau)$ es *superflua con respecto a una fop* ρ : STRUC$[\sigma] \to$ STRUC$[\tau]$ si $\rho(\mathcal{A}) \models \Psi$ para toda $\sigma-$estructura finita $\mathcal{A}$.
2. Una sentencia Ψ en $\mathcal{L}(\tau)$ es *superflua con respecto a* C *y* $\mathcal{L}$ si para toda sentencia Φ en $\mathcal{L}(\tau)$

$$\text{MOD}(\Phi \wedge \Psi) \text{ es C-hard vía fops } \implies \text{ MOD}(\Phi) \text{ es C-hard vía fops.}$$

3. Un fragmento $\mathcal{L}' \leq \mathcal{L}$ es *superfluo con respecto a* C *y* $\mathcal{L}$ si, para todo vocabulario τ, para toda sentencia $\psi \in \mathcal{L}'(\tau)$, y para toda sentencia $\Psi \in \mathcal{L}(\tau)$ tal que MOD$(\Psi) =$ MOD(ψ), Ψ es superflua con respecto a C y $\mathcal{L}$.
4. Un fragmento $\mathcal{L}' \leq \mathcal{L}^*$ es *superfluo con respecto a* C, si para toda lógica $\mathcal{L}$ con $\mathcal{L}^* \leq \mathcal{L}$, $\mathcal{L}'$ es superfluo con respecto a C y $\mathcal{L}$.

Se hace uso de las siguientes definiciones en el lema 2.1. A su vez, este lema nos permite probar el teorema 2.3. Con $\bar{u}$ notamos a una tupla de elementos de algún conjunto, y con $\bar{t}$ a una tupla de términos. Además, la notación $|\bar{a}|$ será usada para la cardinalidad de la tupla $\bar{a}$, y con $[m]$ notaremos al conjunto $\{0, 1, \ldots, m-1\}$

Definición 2.30. Sean σ un vocabulario, $\varphi(\bar{x})$ una fórmula en FO(σ), n un número natural, y $\bar{u} \in [n]^{|\bar{x}|}$ una tupla de números naturales. Decimos que $\langle \varphi(\bar{x}), \bar{u} \rangle$ es *n-consistente* si existe una σ-estructura $\mathcal{A}$, con $\|\mathcal{A}\| = n$, tal que $\mathcal{A} \models \varphi(\bar{u})$. Si S es un subconjunto de STRUC$[\sigma]$, decimos que $\langle \varphi(\bar{x}), \bar{u} \rangle$ es *n-consistente en S* si existe una σ-estructura $\mathcal{A}$ en S, con $\|\mathcal{A}\| = n$, tal que $\mathcal{A} \models \varphi(\bar{u})$.

Definición 2.31 (Uniformidad)**.** Sea $\sigma = \langle R_1^{a_1}, \ldots, R_s^{a_s}, c_1, \ldots, c_t \rangle$ un vocabulario, y sea $S \subset$ STRUC$[\sigma]$. Sean n y k dos números naturales. Decimos que S es (n,k)-*uniforme* si para

- cualquier entero $m \geq n$ y cualesquier enteros no negativos p y q tales que $p + q \leq k$,
- cualquier sucesión $L_1(\bar{t}_1), \ldots, L_p(\bar{t}_p)$ de σ-literales,

- cualquier sucesión $\bar{u}_1, \ldots, \bar{u}_p$ de tuplas, con $\bar{u}_j \in [m]^{|\bar{t}_j|}$ para $1 \leq j \leq p$,
- cualquier sucesión $c_{t_1}, \ldots, c_{t_q}$ de símbolos de constante en σ,
- cualquier sucesión $b_1, \ldots, b_q$ de enteros, con $b_j \in [m]$ para $1 \leq j \leq q$, se tiene lo siguiente:

Si

$$\varphi\left(\bar{u}, \bar{b}\right) = \bigwedge_{j=1}^{p} L_j\left(\bar{u}_j\right) \wedge \bigwedge_{j=1}^{q} c_{t_j} = b_j,$$

es m-consistente, entonces también es m-consistente en S.

El siguiente lema es el lema 4.4 en [2].

Lema 2.1. *Sean σ y $\tau = \langle R_1^{a_1}, \ldots, R_s^{a_s}, c_1, \ldots, c_t \rangle$ dos vocabularios, donde σ tiene constantes de entrada $\{c'_1, \ldots, c'_{t'}\}$, y sea $n \geq 0$ un entero. Adicionalmente, sean*

- *$S \subset STRUC[\sigma]$ un subconjunto de estructuras que contiene todas las estructuras $\mathcal{A}$ con $\|\mathcal{A}\| < n$,*
- *$\psi = \forall \bar{x} \theta\left(\bar{x}\right)$ una sentencia de primer orden con $\theta\left(\bar{x}\right) \in FNC_r(\tau)$ para algún entero r, y*
- *Φ una sentencia en $\mathcal{L}(\tau)$.*

Si S es (n,k)-uniforme para $k \geq t + t' + r$, y ρ es una fop que reduce S a $MOD(\Phi \wedge \psi)$, entonces toda sentencia $\Psi \in \mathcal{L}$ tal que $MOD(\Psi) = MOD(\psi)$, es superflua con respecto a ρ.

El siguiente corolario es usado en la demostración del teorema 2.3.

Corolario 2.1. *Con las hipótesis del lema 2.1, la fop ρ reduce S a $MOD(\Phi)$. Luego, si S es C-hard vía fops, entonces $MOD(\Phi)$ es C-hard en el mismo sentido.*

Demostración. Asumamos las hipótesis del lema 2.1. Supongamos, además, que S es (n,k)-uniforme para $k \geq t + t' + r$ y que ρ es una fop que reduce S a MOD$(\Phi \wedge \psi)$. Vamos a probar que $\mathcal{A} \in S$ si y solo si $\rho(\mathcal{A}) \models \Phi$, para toda σ-estructura $\mathcal{A}$, es decir, que la fop ρ reduce S a MOD(Φ). Supongamos que $\mathcal{A} \in S$. Luego, ya que ρ reduce S a MOD$(\Phi \wedge \psi)$, $\rho(\mathcal{A}) \models \Phi \wedge \psi$. Así, $\rho(\mathcal{A}) \models \Phi$.

Recíprocamente, supongamos que $\rho(\mathcal{A}) \models \Phi$. Sea $\Psi \in \mathcal{L}$ tal que MOD$(\Psi) =$ MOD(ψ). Por el lema 2.1, $\rho(\mathcal{A}) \models \Psi$. Luego, $\rho(\mathcal{A}) \models \psi$. Entonces, $\rho(\mathcal{A}) \models \Phi \wedge \psi$, y ya que ρ reduce S a MOD$(\Phi \wedge \psi)$, $\mathcal{A} \in S$.

De esta manera, si S es C-hard vía fops, entonces MOD(Φ) es C-hard en el mismo sentido. □

La siguiente definición aparece en el teorema 2.3.

Definición 2.32. Una familia $\mathcal{F}$ de problemas sobre el vocabulario

$$\sigma = \langle R_1^{a_1}, \ldots, R_s^{a_s}, c_1, \ldots, c_t \rangle$$

es *completa y uniforme* para una clase de complejidad C, si cada problema en $\mathcal{F}$ es C-completo vía fops, y existen una sucesión $\{n_k\}_{k\geq 0}$ y un número natural m tales que, para todo $k \geq m$, existe un problema (n_k, k)-uniforme S_{n_k} en $\mathcal{F}$ que contiene todas las estructuras $\mathcal{A} \in \text{STRUC}[\sigma]$ con $\|\mathcal{A}\| < n_k$.

El siguiente teorema es uno de los aportes de este proyecto. Su demostración está basada en aquella del teorema 4.7 de [2].

Teorema 2.3. *Sea C una clase de complejidad capturada por $\mathcal{L}^*$, con FO $\leq \mathcal{L}^*$. Si C contiene una familia uniforme y completa $\mathcal{F}$, entonces FO$\forall$ es superflua con respecto a C.*

Demostración. Sea $\mathcal{L}$ una lógica tal que $\mathcal{L}^* \leq \mathcal{L}$. Vamos a probar que FO$\forall$ es superflua con respecto a C y $\mathcal{L}$.

Sean τ un vocabulario, ψ una sentencia en FO$\forall(\tau)$, Ψ una sentencia en $\mathcal{L}(\tau)$ tal que MOD(Ψ) = MOD(ψ), y Φ una sentencia en $\mathcal{L}(\tau)$. Supongamos que MOD$(\Phi \wedge \Psi)$ es C-hard vía fops. Demostraremos que MOD(Φ) es C-hard en el mismo sentido.

Notemos que ψ es lógicamente equivalente a una sentencia ψ' en FO$\forall(\tau)$ cuya parte libre de cuantificadores está en FNC$_r(\tau)$ para algún natural r. En efecto, la parte libre de cuantificadores de ψ es equivalente a un fórmula en FNC con las mismas variables libres.

Sea t el número de símbolos de constante de entrada en σ, el vocabulario de $\mathcal{F}$; y sean $\{n_k\}_{k\geq 0}$ y m la sucesión y el número natural asociados a $\mathcal{F}$. Adicionalmente, sea t' el número de símbolos de constante de entrada en τ. Definamos $k =$ máx $\{m, r + t + t'\}$. Ya que $k \geq m$, existe un problema (n_k, k)-uniforme S_{n_k} en $\mathcal{F}$, C-completo vía fops, que contiene todas las σ-estructuras $\mathcal{A}$ con $\|\mathcal{A}\| < n_k$.

Dado que MOD(Ψ) = MOD(ψ) = MOD(ψ'), y en vista de que MOD$(\Phi \wedge \Psi)$ es C-hard vía fops, entonces MOD$(\Phi \wedge \psi')$ es C-hard vía fops. Luego, existe una fop ρ que reduce S_{n_k} a MOD$(\Phi \wedge \psi')$. Tomando $n = n_k$ y $S = S_{n_k}$, todas las hipótesis del lema 2.1 son satisfechas y, gracias al corolario 2.1, MOD(Φ) es C-hard vía fops. □

Con la siguiente definición y con la proposición a continuación podemos arribar al corolario que mencionamos al principio de la subsección.

Definición 2.33. Si S es un problema sobre el vocabulario σ, se define, para cada

natural n,

$$S_n = S \cup \{\mathcal{A} \in \text{STRUC}[\sigma] : \|\mathcal{A}\| < n\},$$

y se define

$$\mathcal{F}(S) = \{S_n\}_{n \geq 2}.$$

La siguiente proposición es la colección del teorema 10 en el artículo [2], y de los teoremas 4.8 y 4.11 en el artículo [3]. Los problemas REACH, ALT-REACH, 0M-HP, y CO-MONO-TRIANGLE son descritos en el apéndice del primer artículo. Los problemas 2-CLIQUE-COLOURING (2CC) y $(2CC)^c$ son descritos en el segundo artículo. NL, coNP, y Π_2^p son clases de complejidad computacional.

Proposición 2.3. *$\mathcal{F}(REACH)$, $\mathcal{F}(ALT\text{-}REACH)$, $\mathcal{F}(0M\text{-}HP)$, $\mathcal{F}(CO\text{-}MONO\text{-}TRIANGLE)$, $\mathcal{F}(2CC)$, y $\mathcal{F}\left((2CC)^c\right)$ son familias uniformes y completas para NL, P, NP, coNP, Σ_2^p, y Π_2^p respectivamente.*

Todas las clases de complejidad computacional mencionadas en la proposición anterior pueden ser capturadas por lógicas $\mathcal{L}^*$, con FO $\leq \mathcal{L}^*$ (ver el teorema 7.5.2 de [7]). El siguiente corolario es el resultado del teorema 2.3 y la proposición anterior.

Corolario 2.2. *FO$\forall$ es superflua con respecto a NL, P, NP, coNP, Σ_2^p, y Π_2^p.*

Así, si C es una de las clases de complejidad mencionadas en el anterior corolario, la lógica $\mathcal{L}^*$ captura C, y $\mathcal{L}^* \leq \mathcal{L}$, entonces, si ψ es una sentencia en FO$\forall$ y Φ es una sentencia en $\mathcal{L}$ tal que MOD$(\Phi \wedge \psi)$ es C-hard vía fops, luego, existe una sentencia $\Psi \in \mathcal{L}$ tal que MOD(Ψ) = MOD(ψ), de manera que MOD$(\Phi \wedge \Psi)$ es C-hard vía fops y, gracias a lo expuesto, MOD(Φ) es C-hard vía fops. En particular, si $\mathcal{L}$ es $\mathcal{L}^*$, la conclusión es que MOD(Φ) es C-completo vía fops. Es en este sentido que la definición de superfluidad aquí presentada generaliza aquella en [2], pues en aquel artículo se probó, bajo hipótesis adecuadas, que

$$\text{MOD}(\Phi \wedge \psi) \text{ es C-completo vía fops} \implies \text{MOD}(\Phi) \text{ es C-completo vía fops.}$$

El corolario será usado en la sección 4.3.

Capítulo 3

Distribución justa de recursos indivisibles

Un problema de distribución justa consiste en la búsqueda de una manera de asignar bienes en una colección, a varios agentes que han declarado preferencias con respecto a los paquetes de bienes, de modo que la asignación satisfaga criterios de eficiencia y justicia, usualmente matemáticamente formulados. Por ejemplo, los agentes podrían ser una pareja en trámites de divorcio, Alicia y Carlos, y los bienes las posesiones materiales de los dos. Supongamos que los bienes son una casa, un auto, y un terreno, y que Alicia asigna utilidades de 1 para la casa, 2 para el auto y 2 para el terreno, mientras que Carlos declara 2, 5, y 3, respectivamente. Supongamos además que la utilidad que percibe un agente por un paquete de bienes es la suma de los valores de los bienes en el paquete. Si la autoridad asigna a Alicia la casa y el auto, mientras a Carlos le otorga el terreno, Carlos se sentirá perjudicado, pues la utilidad que percibe por el paquete recibido es 3, un número menor que 7, la utilidad que Carlos percibe por el paquete asignado a Alicia. Aquí existe una noción de injusticia, pues Carlos sentiría envidia hacia Alicia. Por otro lado, Alicia no siente envidia hacia Carlos, ya que ella percibe una utilidad de 3 por lo recibido, que es mayor que 2, el valor que ella declaró por el terreno.

Si la autoridad otorga el auto a Carlos, y el terreno y la casa a Alicia, ya no existiría envidia, pues la utilidad que Carlos percibiría, 5, es igual a aquella que percibe por el paquete de Alicia; mientras Alicia percibiría 3 y no se sentiría perjudicada, pues el bien otorgado a Carlos para ella tiene un valor menor, 2. Desde este punto de vista, podríamos considerar que la asignación es justa al no existir envidia en agente alguno. Es precisamente esta la noción de justicia que será considerada en este

trabajo.

Ahora, los bienes o recursos pueden ser divisibles como la energía eléctrica, o indivisibles, y podrían compartirse o no. En este trabajo nos concentraremos exclusivamente en bienes indivisibles que no pueden compartirse. El problema de asignación de bienes indivisibles surge, por ejemplo, en las subastas, la asignación de frecuencias, y el uso de satélites de observación de la Tierra [6].

Para ser precisos, usando la notación en [4], tenemos las siguientes definiciones.

Definición 3.1 (Problema de distribución justa)**.** Un *problema de distribución justa* es una tupla $\mathcal{P} = (A, O, \mathcal{R})$ con

- $A = \{0, 1, \ldots, N-1\}$ un conjunto de N agentes, con $N \geq 2$,
- $O = \{x_0, \ldots, x_{m-1}\}$ un conjunto de m objetos o bienes, con $m \geq 1$, y
- $\mathcal{R} = (\preceq_0, \ldots, \preceq_{N-1})$ un perfil de preferencias, con $\preceq_i$ una relación reflexiva, transitiva y completa sobre 2^O ($\preceq_i$ es un preorden total sobre 2^O), que describe las preferencias del agente i.

$C \preceq_i B$ significa que el agente i considera al paquete B al menos tan bueno como el paquete C. Usaremos $C \sim_i B$ para indicar que $C \preceq_i B$ y $B \preceq_i C$, es decir, en caso de que el agente i manifieste indiferencia con respecto al par de paquetes C y B. Y usaremos $C \prec_i B$ para notar preferencia estricta, es decir, en caso de que $C \preceq_i B$, pero $B \not\preceq_i C$.

Definición 3.2 (Preferencias monótonas)**.** Sean $(A, O, \mathcal{R})$ un problema de distribución justa e $i \in A$. Entonces $\preceq_i$ es *monótona* si

$$C \subseteq B \subseteq O \implies C \preceq_i B.$$

$\mathcal{R}$ es *monótono* si $\preceq_i$ es monótona para todo $i \in A$. Si $\mathcal{R}$ es monótono, diremos que las preferencias son *monótonas*.

Definición 3.3 (Agente indiferente)**.** Sean $(A, O, \mathcal{R})$ un problema de distribución justa e $i \in A$. Diremos que i es *indiferente* si para cualesquier paquetes $C, B \in 2^O$, $C \sim_i B$.

Definición 3.4 (Bien de valor nulo)**.** Sean $(A, O, \mathcal{R})$ un problema de distribución justa y $x_k \in O$. Diremos que x_k es un *bien de valor nulo* si para todo paquete $B \in 2^O$ y para todo $i \in A$, $B \cup \{x_k\} \sim_i B$.

Definición 3.5 (Asignación)**.** Una *asignación* para el problema de distribución justa $(A, O, \mathcal{R})$, es una función $\pi : A \to 2^O$ tal que para cualesquier $i, j \in A$ con $i \neq j$, $\pi(i) \cap \pi(j) = \emptyset$, es decir, los bienes no pueden ser compartidos.

Definición 3.6 (Pareto-dominar)**.** Si π y π' son asignaciones para $(A, O, \mathcal{R})$, π' *Pareto-domina* π si $\pi(i) \preceq_i \pi'(i)$ para todo $i \in A$ y existe $j \in A$ tal que $\pi(j) \prec_j \pi'(j)$.

Que π' Pareto-domine π significa que π' mejora, con respecto a π, la condición de al menos un agente sin perjudicar al resto.

Definición 3.7 (Asignación Pareto-eficiente)**.** Sea π una asignación para $(A, O, \mathcal{R})$. Entonces π será llamada *Pareto-eficiente* si ninguna asignación π' para $(A, O, \mathcal{R})$ Pareto-domina π.

π es Pareto-eficiente si y solo si toda asignación π' que mejora la condición de al menos un agente con respecto a π, necesariamente empeora la de otro.

Definición 3.8 (Asignación completa)**.** Sea π una asignación para $(A, O, \mathcal{R})$. Entonces diremos que π es *completa* si para todo $x_k \in O$ existe $i \in A$ tal que $x_k \in \pi(i)$.

Definición 3.9 (Asignación libre de envidia)**.** Sea π una asignación para $(A, O, \mathcal{R})$. Entonces π está *libre de envidia* si para cualesquier $i, j \in A$, $\pi(j) \preceq_i \pi(i)$.

Intuitivamente, una asignación π está libre de envidia si y solo si cada agente considera al paquete recibido al menos tan bueno como el otorgado a cada otro. Este es el único criterio de justicia que consideraremos en este trabajo.

Los problemas que analizaremos en el siguiente capítulo involucran asignaciones Pareto-eficientes y libres de envidia. Así, abreviaremos la frase "Pareto-eficiente y libre de envidia" con el acrónimo PELE.

La Pareto-eficiencia es una propiedad intuitivamente deseable, pero una asignación que posee esta propiedad no necesariamente es justa. Para ilustrarlo consideremos el siguiente ejemplo.

Ejemplo 3.1. Sean

- $A = \{0, 1, \dots 10\}$,
- $O = \{x_0, \dots, x_{19}\}$, y
- $\mathcal{R} = (\preceq_0, \dots, \preceq_{10})$ un perfil de preferencias, con $\preceq_i$ estrictamente monótona para todo agente i, es decir,

$$C \subsetneq B \subseteq O \implies C \prec_i B$$

  para todo $i \in A$

Asumamos π es una asignación tal que $\pi(0) = O$, de modo que $\pi(i) = \emptyset$ para $i \neq 0$. Si π' es otra asignación factible, no le otorga el paquete O al agente 0, o le

otorga un paquete no vacío a otro agente. Así, necesariamente

$$\pi'(0) \subsetneq O = \pi(0),$$

de manera que $\pi'(0) \prec_0 \pi(0)$. Sigue que π es Pareto-eficiente pues, para cualquier otra asignación π', existe un agente $i = 0$ tal que $\pi(i) \not\preceq_i \pi'(i)$. Sin embargo, asigna todo el conjunto de bienes al agente 0; luego, en vista de que $\emptyset \prec_1 O$ por monotonía estricta, tenemos

$$\pi(0) = O \not\preceq_1 \emptyset = \pi(1),$$

es decir, π no está libre de envidia.

Las siguientes proposiciones serán útiles en el desarrollo del trabajo.

Proposición 3.1. *Dado un problema de distribución justa $(A, O, \mathcal{R})$ en el que las preferencias son monótonas, si I es un conjunto de agentes indiferentes con $|I| \leq N - 2$, entonces, existe una asignación PELE π para $(A, O, \mathcal{R})$ si y solo si existe una asignación PELE π' para el problema restringido que no considera los agentes en I.*

Demostración. Primero, hacemos notar que la condición $|I| \leq N - 2$ simplemente asegura que el problema restringido tenga al menos dos agentes. Si I es vacío, el resultado es inmediato. Así, suponemos I es no vacío.

La idea de la demostración es la siguiente. Para demostrar ($\Rightarrow$), se restringe el dominio de π a $A \setminus I$, y para probar ($\Leftarrow$), se extiende π' definiendo $\pi(i) = \emptyset$ para $i \in I$, con la obtención del resultado gracias al hecho de que a los agentes indiferentes les resulta igual, desde el punto de vista de las preferencias, cualquier paquete recibido.

($\Rightarrow$) Asumamos existe una asignación PELE π para $(A, O, \mathcal{R})$. Definamos π' como la restricción $\pi|_{A \setminus I}$. Luego π' es asignación. Ya que π está libre de envidia, para cualesquier $i, j \in A$ se tiene $\pi(j) \preceq_i \pi(i)$, en particular para $i, j \in A \setminus I$. Luego para todo $i, j \in A \setminus I$, $\pi'(j) \preceq_i \pi'(i)$, es decir, π' está libre de envidia.

Supongamos, con el objetivo de llegar a una contradicción, que π' no es Pareto-eficiente. Sigue que existe una asignación para el problema restringido $\widehat{\pi}$ que Pareto-domina π'. Definamos la asignación $\widetilde{\pi}$ para $(A, O, \mathcal{R})$ por la fórmula

$$\widetilde{\pi}(i) = \begin{cases} \widehat{\pi}(i) & \text{si } i \in A \setminus I, \\ \emptyset & \text{si } i \in I. \end{cases}$$

Entonces, existe $i \in A \setminus I$ tal que

$$\pi(i) = \pi'(i) \prec_i \widehat{\pi}(i) = \widetilde{\pi}(i).$$

Es más, para todo $j \in A \setminus I$,

$$\pi(j) = \pi'(j) \preceq_j \widehat{\pi}(j) = \widetilde{\pi}(j).$$

Ahora, para $j \in I$,

$$\pi(j) \preceq_j \varnothing = \widetilde{\pi}(j),$$

pues j es indiferente. En conclusión, $\pi(j) \preceq_j \widetilde{\pi}(j)$ para todo $j \in A$, y existe $i \in A$ tal que $\pi(i) \prec_i \widetilde{\pi}(i)$; en otras palabras, $\widetilde{\pi}$ Pareto-domina π, lo que contradice el hecho de que π es Pareto-eficiente.

($\Leftarrow$) Supongamos existe una asignación PELE π' para el problema restringido. Tomemos π con

$$\pi(i) = \begin{cases} \pi'(i) & \text{si } i \in A \setminus I, \\ \varnothing & \text{si } i \in I. \end{cases}$$

Entonces π es asignación para $(A, O, \mathcal{R})$.

Ahora, afirmamos que π está libre de envidia. En efecto, sean $i, j \in A$. Probaremos que $\pi(j) \preceq_i \pi(i)$; lo haremos por casos. Si $i, j \in A \setminus I$, $\pi(j) \preceq_i \pi(i)$, pues π' está libre de envidia. Si $i \in I$,

$$\pi(j) \preceq_i \varnothing = \pi(i)$$

para todo agente j, pues i es indiferente. En el caso $i \in A \setminus I$ y $j \in I$, se tiene

$$\varnothing = \pi(j) \preceq_i \pi(i),$$

pues $\varnothing \subseteq \pi(i)$ y las preferencias son monótonas. Vemos entonces que en todos los casos $\pi(j) \preceq_i \pi(i)$. En vista de que $i, j \in A$ fueron tomados arbitrarios, π está libre de envidia.

Ahora supongamos, por contradicción, que π no es Pareto-eficiente. Luego, existe $\widetilde{\pi}$ asignación para $(A, O, \mathcal{R})$ que domina π. Tomemos $\widehat{\pi} = \widetilde{\pi}|_{A \setminus I}$ asignación para el problema restringido. Sabemos existe $i \in A$ tal que $\pi(i) \prec_i \widetilde{\pi}(i)$. Así, i no es indiferente. Si todos los agentes en A son indiferentes, tenemos una contradicción. En caso contrario, $i \in A \setminus I$. De esta manera,

$$\pi'(i) = \pi(i) \prec_i \widetilde{\pi}(i) = \widehat{\pi}(i).$$

También es cierto que para todo $j \in A$, $\pi(j) \preceq_j \widetilde{\pi}(j)$. En particular, para $j \in A \setminus I$ se tiene

$$\pi'(j) = \pi(j) \preceq_j \widetilde{\pi}(j) = \widehat{\pi}(j).$$

Esto implica que π' no es Pareto-eficiente, lo cual es contradictorio. □

Proposición 3.2. *Cuando hay a lo mucho un agente que no es indiferente, siempre existe una asignación PELE.*

Demostración. Sea $(A, O, \mathcal{R})$ un problema de distribución justa en el que hay a lo mucho un agente que no es indiferente. Definamos π con $\pi(i) = \emptyset$ para todo agente i indiferente. Si existe $j \in A$ que no es indiferente, hacemos $\pi(j) = B$, con $B \in 2^O$ tal que $C \preceq_j B$ para todo $C \in 2^O$. El paquete B existe ya que 2^O es un conjunto finito no vacío, y por la proposición A.1.

Sigue que $\pi(k) \sim_i \pi(i)$ para todo agente indiferente i y para todo agente k, lo que implica $\pi(k) \preceq_i \pi(i)$ para estos agentes. Además, si j no es indiferente, $\pi(k) \preceq_j \pi(j)$ para todo agente k. Luego π está libre de envidia. También es cierto que para toda asignación π' y para todo agente k, $\pi'(k) \preceq_k \pi(k)$, con lo cual π es Pareto eficiente. □

3.1. Preferencias binarias

Sea $(A, O, \mathcal{R})$ un problema de distribución justa. Las preferencias del agente i son *binarias* si existe un conjunto $Good_i \subseteq 2^O$ tal que, para cualesquier paquetes $C, B \in 2^O$, $C \preceq_i B$ si y solo si $C \notin Good_i$ o $B \in Good_i$. La idea es que los paquetes en $Good_i$ son aquellos que el agente i desea o prefiere. El perfil de preferencias $\mathcal{R}$ es *binario* si las preferencias de i son binarias para todo $i \in A$.

Notar que, para todo $C, B \in Good_i$, $C \sim_i B$; asimismo, para todo $C, B \notin Good_i$, $C \sim_i B$. En este sentido podríamos decir que el preorden total sobre 2^O que genera $Good_i$ está compuesto por dos niveles: en uno de ellos están todos los paquetes que no pertenecen a $Good_i$ y en el otro aquellos que sí pertenecen.

Proposición 3.3. *Sea $(A, O, \mathcal{R})$ un problema de distribución justa. $i \in A$ es indiferente si y solo si $Good_i = 2^O$ o $Good_i = \emptyset$.*

Demostración. Supongamos $Good_i = 2^O$. Sean B, C miembros de 2^O. Entonces B y C son miembros de $Good_i$. Luego $B \preceq_i C$ y $C \preceq_i B$, es decir, $B \sim_i C$. Si $Good_i = \emptyset$, tomando otra vez B, C en 2^O vemos que B y C no son miembros de $Good_i$, lo que implica $B \preceq_i C$ y $C \preceq_i B$. Recíprocamente, asumamos que i es indiferente. Para argumentar por contradicción supongamos $Good_i \neq 2^O$ y $Good_i \neq \emptyset$. Sigue que existen B y C en 2^O con $C \notin Good_i$ y $B \in Good_i$, es decir, $B \not\preceq_i C$, con la consecuencia de que $B \sim_i C$ no es cierta. Así i no es indiferente, contradicción. □

Siguiendo a [4], las preferencias binarias pueden ser representadas de manera compacta con fórmulas de la lógica proposicional (ver la subsección 2.1.1). Dado un problema de distribución justa $(A, O, \mathcal{R})$, tomando $V = O = \{x_0, \ldots, x_{m-1}\}$, se dice que la fórmula φ_i en L_V *captura* $\preceq_i$ si y solo si $\{M \subseteq O | \, M \models \varphi_i\} = Good_i$.

La siguiente es la proposición 1 en [4].

Proposición 3.4. *Sea $(A, O, \mathcal{R})$ un problema de distribución justa. Supongamos que las preferencias del agente i son binarias. Entonces las siguientes condiciones son equivalentes:*

1. *Las preferencias de i son monótonas.*
2. *Para cualesquier $B, C \in 2^O$, $C \in Good_i$ y $C \subseteq B$ implican $B \in Good_i$.*
3. *Las preferencias del agente i pueden ser capturadas por una fórmula positiva φ_i.*

En el contexto de preferencias binarias monótonas, es posible que $\emptyset \in Good_i$, y en ese caso todo paquete de bienes satisface al agente i, es decir, $Good_i = 2^O$. También es posible que $Good_i = \emptyset$, lo que significa que ningún paquete satisface a i.

Además, las preferencias binarias monótonas pueden ser capturadas por fórmulas positivas que están en FND, o con las constantes lógicas $\top, \bot$. Si $\emptyset \neq Good_i \neq 2^O$ y consideramos solo los paquetes minimales en él, es decir, aquellos que no contienen a ningún otro, entonces $Good_i$ es igual a la colección de todos los superconjuntos de esos conjuntos o paquetes minimales. Así, si $O = \{x_0, \ldots, x_{10}\}$ y $Good_i$ está conformado por todos los superconjuntos de $\{x_1\}$ y los de $\{x_2, x_4\}$, podemos representar estas preferencias de manera compacta con la fórmula $\varphi_i = x_1 \vee (x_2 \wedge x_4)$. Por otro lado, usamos $\varphi_i = \top$ cuando $Good_i = 2^O$, y $\varphi_i = \bot$ cuando $Good_i = \emptyset$.

Si ψ_i es una fórmula positiva en FNC, por ejemplo, $\psi_i = (x_1 \vee x_2) \wedge (x_1 \vee x_4)$, se la puede leer de la siguiente manera: el agente i desea los paquetes que tengan al menos un bien de $\{x_1, x_2\}$ y al menos uno de $\{x_1, x_4\}$.

La demostración de la siguiente proposición está basada en aquella de la proposición 7 en [4].

Proposición 3.5. *Dado un problema de distribución justa en el que las preferencias son binarias, no necesariamente monótonas, y ningún agente es indiferente, si existe un conjunto de paquetes Good tal que para todo agente i, $Good_i = Good$, entonces una asignación π es PELE si y solo si para todo agente i, $\pi(i) \in Good$.*

Demostración. Sea $(A, O, \mathcal{R})$ un problema de distribución justa que satisface las hipótesis del enunciado. En vista de que los agentes no son indiferentes, por la proposición 3.3, $\emptyset \neq Good$. Supongamos que π es una asignación PELE para $(A, O, \mathcal{R})$. Existen dos casos: o bien para todo i, $\pi(i) \notin Good$, o bien existe j tal que $\pi(j) \in Good$.

En el primer caso, definamos una asignación π' con $\pi'(0) = B \in \mathit{Good}$ y $\pi'(i) = \emptyset$ para $i \neq 0$. Entonces $\pi(0) \prec_0 \pi'(0)$ y $\pi(i) \preceq_i \pi'(i)$ para todo i, lo cual contradice la Pareto-eficiencia de π. Así, nos concentramos en el segundo caso. Ya que π está libre de envidia, para todo agente i, $\pi(j) \preceq_i \pi(i)$, es decir, para todo i, $\pi(j) \notin \mathit{Good}$ o $\pi(i) \in \mathit{Good}$. Ya que $\pi(j) \in \mathit{Good}$, necesariamente $\pi(i) \in \mathit{Good}$ para todo i.

Recíprocamente, supongamos que para todo agente i, $\pi(i) \in \mathit{Good}$. Así, para todo paquete $B \in 2^O$ y para todo i, $B \preceq_i \pi(i)$. De esta manera para cualesquier i, j, $\pi(j) \preceq_i \pi(i)$. Sigue que π está libre de envidia. Además, si π' es otra asignación factible, necesariamente $\pi'(i) \preceq_i \pi(i)$ para todo i. Luego π es Pareto-eficiente. □

3.2. Preferencias aditivas

Si en un problema de distribución justa, $(A, O, \mathcal{R})$, cada agente declara un valor por cada bien, de manera que el valor de un paquete arbitrario es la suma de los valores de los bienes en él, entonces el problema es denominado *problema con utilidades aditivas*. Tomando la notación de [6], en este tipo de problemas existe una función $v : A \times O \to \mathbb{R}$ tal que para cualquier agente i y cualquier paquete B, la utilidad que i percibe por B es $u_i(B) = \sum_{o \in B} v(i, o)$. Una consecuencia de esta definición es que $u_i(\emptyset) = 0$. De esta manera, si C y B son paquetes e i es un agente, $C \preceq_i B$ si y solo si $u_i(C) \leq u_i(B)$; así, $C \sim_i B$ si y solo si $u_i(C) = u_i(B)$, y $C \prec_i B$ si y solo si $u_i(C) < u_i(B)$. Notar que de esta forma podemos describir el problema con el triple (A, O, v).

Aquí, un agente i es indiferente si y solo si $u_i(\{x_k\}) = 0$ para todo k, ya que la utilidad que i percibe por cualquier paquete es 0. Por otro lado, una condición suficiente para que un bien x_k tenga valor nulo es que $u_i(x_k) = 0$ para todo agente i.

El hecho de que una asignación π' Pareto-domine otra, π, está caracterizado por la afirmación: existe i tal que $u_i(\pi(i)) < u_i(\pi'(i))$, y para todo j, $u_j(\pi(j)) \leq u_j(\pi'(j))$. Recordando la definición de Pareto-eficiencia de una asignación π, esta es equivalente a la condición: para toda asignación π', existe i tal que $u_i(\pi'(i)) < u_i(\pi(i))$ o, para todo j, $u_j(\pi'(j)) = u_j(\pi(j))$.

La propiedad de una asignación π de estar libre de envidia, en este contexto equivale a: para todo i, j, $u_i(\pi(j)) \leq u_i(\pi(i))$.

En este trabajo consideraremos solo funciones v con imagen en $\mathbb{Z}_{\geq 0}$, es decir, en el conjunto de enteros no negativos. Esta restricción garantiza que las preferencias

sean monótonas, pues si i es un agente y $C \subseteq B \subseteq O$, entonces

$$u_i(C) = \sum_{o \in C} v(i,o) \leq \sum_{o \in C} v(i,o) + \sum_{o \in B \setminus C} v(i,o) = \sum_{o \in B} v(i,o) = u_i(B).$$

La siguiente proposición asegura que si para todo agente i y para todo bien x_j, $u_i(\{x_j\}) \in \{0,1\}$, entonces una asignación es Pareto eficiente si y solo si todo bien deseado es asignado a un agente que lo desea. Naturalmente, $o \in O$ es un bien deseado si existe al menos un agente i con $v(i,o) = u_i(\{o\}) = 1$, es decir, si existe al menos un agente que lo desea. La demostración está basada en aquella de la proposición 21 en [4].

Proposición 3.6. *En caso de que para todo agente i y para todo bien o, $v(i,o) \in \{0,1\}$, una asignación π para el problema (A,O,v) es Pareto-eficiente si y solo si para todo bien o con $v(i,o) = 1$ para algún i, existe un agente k tal que $v(k,o) = 1$ y $o \in \pi(k)$.*

Demostración. ($\Rightarrow$) La prueba es por contradicción. Supongamos que π es Pareto-eficiente, y que existe un bien deseado o tal que para todo agente k que lo desea, $o \notin \pi(k)$. Hay dos casos: o bien existe un agente j tal que $o \in \pi(j)$, o bien para todo l, $o \notin \pi(l)$. En el primero, necesariamente $v(j,o) = 0$. Consideremos la asignación $\widehat{\pi}$ con $\widehat{\pi}(j) = \pi(j) \setminus \{o\}$, $\widehat{\pi}(k) = \pi(k) \cup \{o\}$ para algún agente k que desea o, y $\widehat{\pi}(l) = \pi(l)$ para $l \neq j,k$. Luego, $u_j(\pi(j)) = u_j(\widehat{\pi}(j))$, $u_k(\pi(k)) < u_k(\widehat{\pi}(k))$ y $u_l(\pi(l)) = u_l(\widehat{\pi}(l))$, con lo cual $\widehat{\pi}$ domina π. Esto contradice la Pareto-eficiencia de π.

Para el otro caso, tomemos $\widehat{\pi}(k) = \pi(k) \cup \{o\}$ para algún agente k que desea o, y $\widehat{\pi}(l) = \pi(l)$ para $l \neq k$. En este caso también $\widehat{\pi}$ domina π. Ya que hemos considerado todos los casos, y en todos existe contradicción, concluimos que todo bien deseado es asignado por π a un agente que lo desea.

($\Leftarrow$) La demostración es por contradicción. Supongamos que todo bien deseado es asignado por π a un agente que lo desea, y que existe una asignación $\widehat{\pi}$ que Pareto-domina π. Entonces

$$0 \leq \sum_{i \in A} u_i(\pi(i)) < \sum_{i \in A} u_i(\widehat{\pi}(i)).$$

Así, si ningún agente desea bien alguno, es decir, si para todo $o \in O$ y para todo i, $v(i,o) = 0$, entonces

$$\sum_{i \in A} u_i(\widehat{\pi}(i)) = 0,$$

lo cual es contradictorio. Por otro lado, si existe al menos un bien deseado, considerando la cardinalidad $|B|$ del conjunto $B \subseteq O$ de todos los bienes deseados, ya

que todos los agentes declaran valor 0 por los bienes no deseados, y en vista de que todos los bienes deseados son asignados por π a agentes que los desean, tenemos

$$\sum_{i \in A} u_i(\pi(i)) = |B| < \sum_{i \in A} u_i(\widehat{\pi}(i)),$$

lo cual es imposible, pues $\widehat{\pi}$ asigna cada objeto deseado a lo mucho a un agente. En cualquier caso existe contradicción, por lo cual π es Pareto-eficiente. □

Capítulo 4

Caracterización descriptiva de la complejidad computacional de algunos problemas de distribución justa

Estudiaremos, con técnicas sintácticas, tres problemas de decisión en el contexto de distribución justa de recursos indivisibles. Todos involucran la existencia de una asignación PELE. Trataremos primero un problema en el que las preferencias son binarias y monótonas. Luego cambiaremos de escenario: las preferencias de los agentes serán aditivas. Analizaremos, desde el punto de vista descriptivo, la complejidad computacional de dos problemas en ese contexto, siendo uno una restricción del otro. Todos estos problemas aparecen en el artículo [4].

La idea es plantear, por cada problema, una sentencia que lo caracterice; la sentencia será una fórmula de algún lenguaje lógico. Los modelos de la sentencia corresponderán a las instancias positivas del problema. Una vez que se tenga la sentencia, se procederá a probar hardness.

4.1. Problema 1: dos agentes con preferencias binarias monótonas idénticas capturadas por una fórmula proposicional

Comenzamos con el problema de la existencia de una asignación PELE cuando el número de agentes es dos y ambos tienen las mismas preferencias. Este está relacionado con la proposición 7 en [4], donde se prueba que el problema respectivo es NP-completo vía reducciones polinomiales. Probaremos en esta sección que nuestro problema es NP-completo vía fops. Los resultados que presentamos a continuación están basados en la demostración de la proposición 7.

El vocabulario que utilizaremos para este problema es

$$\nu = \langle R^2 \rangle.$$

Si $\mathcal{B} = \langle |\mathcal{B}|, R^{\mathcal{B}} \rangle$ es una ν-estructura con $\|\mathcal{B}\| = n$, los elementos de $|\mathcal{B}|$ representarán los bienes del conjunto

$$O = \{x_0, x_1, \ldots, x_{n-1}\},$$

y representarán las cláusulas en la tupla

$$\langle Y_0, Y_1, \ldots, Y_{n-1} \rangle,$$

con $Y_k \subseteq O$. Esta tupla da lugar a una fórmula proposicional ψ (ver la observación 2.2). La fórmula ψ, a su vez, da lugar a un conjunto

$$Good = Good_0 = Good_1 = \{M \subseteq O |\, M \models \psi\},$$

es decir, captura las preferencias de los dos agentes, 0 y 1. Además, $(j,k) \in R^{\mathcal{B}}$ si y solo si la cláusula Y_j contiene x_k. De esta manera, $R^{\mathcal{B}}$ genera la fórmula ψ.

Para ser más precisos, la instancia del problema que asociaremos a una ν-estructura $\mathcal{B}$ de tamaño n, será $(A, O, \mathcal{R})$ con

- $A = \{0, 1\}$,
- $O = \{x_0, \ldots, x_{n-1}\}$ y
- $\mathcal{R}$ el perfil de preferencias generado por ψ.

Ejemplo 4.1. La estructura $\mathcal{B}$ con

$$|\mathcal{B}| = \{0,1,2,3\} \text{ y}$$
$$R^{\mathcal{B}} = \{(0,0),(1,2),(1,3),(2,1),(2,2),(3,1),(3,3)\},$$

tiene asociados el conjunto de bienes

$$O = \{x_0, x_1, x_2, x_3\},$$

y la tupla de cláusulas

$$\langle Y_0, Y_1, Y_2, Y_3 \rangle,$$

con

$$Y_0 = \{x_0\}\,,\ Y_1 = \{x_2, x_3\}\,,\ Y_2 = \{x_1, x_2\}\,,\ \text{y}\ \ Y_3 = \{x_1, x_3\}\,.$$

Así, el triple asociado a $\mathcal{B}$ está conformado por

- $A = \{0,1\}$,
- $O = \{x_0, x_1, x_2, x_3\}$ y
- $\mathcal{R}$ generado por

$$\psi = x_0 \wedge (x_2 \vee x_3) \wedge (x_1 \vee x_2) \wedge (x_1 \vee x_3).$$

Ejemplo 4.2. Ahora, si la estructura $\mathcal{B}$ es tal que

$$|\mathcal{B}| = \{0,1,2,3\} \text{ y}$$
$$R^{\mathcal{B}} = \{(0,0),(1,2),(1,3),(2,1),(2,2)\},$$

entonces

$$O = \{x_0, x_1, x_2, x_3\},$$

y la tupla

$$\langle Y_0, Y_1, Y_2, Y_3 \rangle$$

es tal que

$$Y_0 = \{x_0\}\,,\ Y_1 = \{x_2, x_3\}\,,\ Y_2 = \{x_1, x_2\}\,,\ \text{y}\ \ Y_3 = \{\}\,.$$

Notemos que existe $j = 3$ tal que, para todo $k \in |\mathcal{B}|$, $(j,k) \notin R^{\mathcal{B}}$. Equivalentemente, la cláusula Y_3 es vacía. Por la proposición 2.1, $\psi \equiv \bot$, es decir, $Good = \emptyset$. Entonces el triple asociado a $\mathcal{B}$ está conformado por

- $A = \{0,1\}$,

- $O = \{x_0, x_1, x_2, x_3\}$, y
- $\mathcal{R}$ generado por $\psi \equiv \bot$.

Proposición 4.1. *Sean $\mathcal{B}$ una ν-estructura de tamaño n, y $(A, O, \mathcal{R})$ su triple asociado. Entonces, al menos una de las cláusulas en*

$$\langle Y_0, \ldots, Y_{n-1} \rangle$$

es vacía si y solo si los miembros de A son indiferentes.

Demostración. Primero, notemos no es posible que $Good = 2^O$. Supongamos por contradicción lo contrario. Entonces $\emptyset \in Good$, es decir, $\emptyset \models \psi$, lo que implica $\psi \not\equiv \bot$. Luego, gracias a la proposición 2.1, no hay cláusulas vacías. Así, tomando $M = \emptyset$ o, en otras palabras, asignando el valor false a todas las variables en O, dado que ψ es positiva, $M \not\models \psi$, contradicción.

Ahora, supongamos existe al menos una cláusula vacía. Por la proposición 2.1, $\psi \equiv \bot$. Entonces

$$Good_0 = Good_1 = Good = \emptyset,$$

y por la proposición 3.3, los agentes son indiferentes. Recíprocamente, si los agentes son indiferentes, gracias a la misma proposición, $Good = \emptyset$ o $Good = 2^O$. Por lo expuesto en el párrafo anterior, necesariamente $Good = \emptyset$, lo que implica $\psi \equiv \bot$. Invocando la proposición 2.1, existe al menos una cláusula vacía. □

Observemos que, dada una ν-estructura, hay solo dos posibilidades: o bien todas las cláusulas son no vacías, o bien existe al menos una cláusula vacía. En el primer caso, la fórmula ψ es positiva; en el segundo caso, $Good = \emptyset$. Luego, en ambos casos, gracias a la proposición 3.4, las preferencias son monótonas.

Podemos entonces plantear el problema.

PROBLEMA 1:

Instancia: triple $(A, O, \mathcal{R})$ conformado por

- $A = \{0, 1\}$,
- $O = \{x_0, \ldots, x_{n-1}\}$ y
- $\mathcal{R} = (\preceq_0, \preceq_1)$ el perfil de preferencias generado por ψ.

Pregunta: ¿Los agentes no son indiferentes y existe asignación PELE para $(A, O, \mathcal{R})$?

Notemos que si algún agente es indiferente, gracias a la proposición 3.2, siempre existe una asignación PELE.

Observemos, además, que todas las instancias del PROBLEMA 1 son generadas por ν-estructuras. Así, en todas, existe un número de cláusulas igual al número de bienes. Se puede pensar entonces que el PROBLEMA 1 es un problema restringido. ¿Podemos, para cada triple $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$ con

- $\widehat{A} = \{0, 1\}$,
- $\widehat{O} = \{x_0, \ldots, x_{m-1}\}$, y
- $\widehat{\mathcal{R}} = (\lesssim_0, \lesssim_1)$ generado por una fórmula positiva $\widehat{\psi}$ en FNC, con p disyunciones, y con variables en $\widehat{O}$, o generado por un conjunto

$$\widehat{Good} = \widehat{Good}_0 = \widehat{Good}_1 = \emptyset,$$

crear una ν-estructura $\mathcal{B}$, que sea una instancia positiva del PROBLEMA 1 si y solo si existe asignación PELE para $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$ y los agentes en este triple no son indiferentes? La proposición B.1 resuelve estas situaciones. Notar que m podría ser distinto de p.

Ahora vamos a probar que el PROBLEMA 1 es NP-completo vía fops. Para probar NP-hardness usaremos el problema SET-SPLITTING, el cual es NP-completo vía fops [10] (teorema 4.74).

PROBLEMA: SET-SPLITTING

Instancia: Un conjunto S con n elementos y una tupla

$$C = \langle X_0, X_1, \ldots, X_{n-1} \rangle$$

de n subconjuntos de S.

Pregunta: ¿Existe una partición de S en dos subconjuntos, S_1 y S_2, tal que ningún subconjunto en C está totalmente contenido en S_1 o S_2? En otras palabras, ¿existe una partición de S en dos subconjuntos, S_1 y S_2, tal que todo subconjunto en C tiene al menos un miembro en S_1 y uno en S_2?

El vocabulario usado para este problema es $\mu = \langle Q^2 \rangle$. Dada una estructura $\mathcal{A}$ con ese vocabulario, los miembros de $|\mathcal{A}|$ representan a la vez los elementos de S y los conjuntos en C. Así, $(i, j) \in Q^{\mathcal{A}}$ si y solo si $j \in X_i$.

Lema 4.1. *La sentencia*

$$\Psi_1 = (\exists \mathbf{P})(\forall x)(\exists y)(\exists z)(Q^2(x, y) \wedge Q^2(x, z) \wedge \mathbf{P}(y) \wedge \neg \mathbf{P}(z))$$

define SET-SPLITTING.

Demostración. Sea $\mathcal{A}$ una μ-estructura. Vamos a probar que

$$\mathcal{A} \models \Psi_1 \tag{4.1}$$

si y solo si existe una partición de $S = |\mathcal{A}|$ en dos subconjuntos, S_1 y S_2, tal que todo subconjunto en C tiene al menos un miembro en S_1 y uno en S_2.

Asumamos 4.1. Entonces existe un conjunto $P \subsetneq |\mathcal{A}|$ tal que, para todo $i \in |\mathcal{A}|$, existen $k, l \in |\mathcal{A}|$ con

$$(i,k), (i,l) \in Q^{\mathcal{A}},\ k \in P \text{ y } l \notin P. \tag{4.2}$$

Definamos $S_1 = P$ y $S_2 = |\mathcal{A}| \setminus P$. Así, para todo $i \in |\mathcal{A}|$, existen $k, l \in |\mathcal{A}|$ tales que

$$k \in X_i,\ l \in X_i,\ k \in S_1 \text{ y } l \in S_2;$$

en otras palabras, existe una partición de S en dos subconjuntos, S_1 y S_2, tal que todo subconjunto en C tiene al menos un miembro en S_1 y uno en S_2.

Recíprocamente, supongamos existe una partición de S en dos subconjuntos, S_1 y S_2, tal que todo subconjunto en C tiene al menos un miembro en S_1 y uno en S_2. Definamos $P = S_1$. Entonces, para todo $i \in |\mathcal{A}|$, existen $k, l \in |\mathcal{A}|$ tales que

$$k \in X_i,\ l \in X_i,\ k \in P \text{ y } l \notin P;$$

en otra palabras, existe $P \subsetneq |\mathcal{A}|$ tal que, para todo $i \in |\mathcal{A}|$, existen $k, l \in |\mathcal{A}|$ tales que 4.2. Se tiene así 4.1. □

Teorema 4.1. *El PROBLEMA 1 es NP-completo vía fops.*

Demostración. Proponemos la siguiente sentencia para definir el problema:

$$\Psi_2 = (\exists \mathbf{P})(\forall x)(\exists y)(\exists z)(R^2(x,y) \wedge R^2(x,z) \wedge \mathbf{P}(y) \wedge \neg \mathbf{P}(z)).$$

Los modelos de Ψ_2 corresponden precisamente a las instancias positivas del PROBLEMA 1. En efecto, sea $\mathcal{B} = \langle |\mathcal{B}|, R^{\mathcal{B}} \rangle$ una ν-estructura con $(A, O, \mathcal{R})$ su triple asociado. Asumamos que los agentes no son indiferentes y que existe asignación PELE π para $(A, O, \mathcal{R})$. Gracias a la proposición 3.5, $\pi(i) \in \mathit{Good}$ para $i = 0, 1$, es decir, $\pi(i) \models \psi$ para $i = 0, 1$. Además, por la proposición 4.1, todas las cláusulas de la tupla asociada a $\mathcal{B}$ son no vacías. De esta manera, toda cláusula tiene variables x_k, x_l a las cuales $\pi(0)$ y $\pi(1)$ respectivamente asignan el valor de verdad true, es

decir, tales que $x_k \in \pi(0)$ y $x_l \in \pi(1)$. Definiendo

$$P = \{k \in |\mathcal{B}|| \, x_k \in \pi(0)\},$$

en vista de que los bienes no pueden ser compartidos, vemos que, para todo i, existen k, l tales que

$$x_k \in Y_i,\ x_l \in Y_i,\ k \in P \text{ y } l \notin P, \tag{4.3}$$

es decir, existe $P \subsetneq |\mathcal{B}|$ tal que, para todo i, existen k, l tales que

$$(i,k), (i,l) \in R^{\mathcal{B}},\ k \in P \text{ y } l \notin P; \tag{4.4}$$

equivalentemente, $\mathcal{B} \models \Psi_2$.

Recíprocamente, supongamos $\mathcal{B} \models \Psi_2$. Luego, existe $P \subsetneq |\mathcal{B}|$ tal que, para todo i, existen k, l tales que 4.4. Así, para todo i, existen k, l tales que 4.3. Tomemos

$$\begin{aligned}\pi(0) =& \{x_k \in O |\, k \in P\} \text{ y}\\ \pi(1) =& \{x_l \in O |\, l \in |\mathcal{B}| \setminus P\}.\end{aligned}$$

Notar que $\pi(0) \cap \pi(1) = \emptyset$. Además, toda cláusula de la tupla asociada a $\mathcal{B}$ tiene variables $x_k \in \pi(0)$ y $x_l \in \pi(1)$. Luego, en vista de que todas las cláusulas son no vacías, gracias a la proposición 4.1 los agentes no son indiferentes. También es cierto que $\pi(i) \models \psi$ para $i = 0, 1$. Sigue que $\pi(i) \in \mathit{Good}$ para $i = 0, 1$ y, gracias a la proposición 3.5, π es una asignación PELE para $(A, O, \mathcal{R})$.

Por lo expuesto, para toda ν-estructura $\mathcal{B}$, su triple asociado $(A, O, \mathcal{R})$ es una instancia positiva del PROBLEMA 1 si y solo si $\mathcal{B} \models \Psi_2$.

Ψ_2 es una sentencia existencial de segundo orden y, por lo tanto, gracias al teorema 2.1, el PROBLEMA 1 está en NP. A continuación demostraremos que este es NP-completo vía fops.

Definamos el operador sintáctico $T : \mathrm{SO}\exists(\nu) \to \mathrm{SO}\exists(\mu)$ por la fórmula

$$T\Psi = \Psi\left(R^2(x,y)/Q^2(x,y)\right).$$

T es un operador acotado (ver ejemplo 2.15). Además, T es un operador que sustituye el predicado $R^2(x,y)$ por la fórmula proyectiva $Q^2(x,y)$ (ver ejemplo 2.14). Luego, gracias a la proposición 2.2, T sustenta completitud. Gracias al lema 4.1, $\mathrm{MOD}(\Psi_1)$ es NP-completo vía fops, pero

$$\mathrm{MOD}(\Psi_1) = \mathrm{MOD}(T\Psi_2)$$

Luego, en vista de que T sustenta completitud (definición 2.28), MOD (Ψ_2) es NP-completo vía fops, es decir, el PROBLEMA 1 es NP-completo en ese sentido. □

4.2. Problema 2: preferencias aditivas restringidas

Las preferencias de los agentes ahora serán aditivas, es decir, cada uno asignará un valor a cada bien y la utilidad de un paquete será la suma de los valores de los bienes en él. El problema consiste en decidir la existencia de una asignación PELE cuando para todo bien x_j y para todo agente i, $v(i, x_j) \in \{0, 1\}$. Este problema es tratado en la proposición 21 de [4], donde se muestra es NP-completo vía reducciones polinomiales. Demostraremos, basándonos en la proposición 21, que este problema es NP-completo vía fops.

Usaremos el vocabulario

$$\sigma = \langle V^3 \rangle.$$

Si $\mathcal{B} = \langle |\mathcal{B}|, V^{\mathcal{B}} \rangle$ es una estructura con vocabulario σ, los miembros de $|\mathcal{B}|$ representarán, además de números, bienes y agentes, es decir, la instancia tendrá $\|\mathcal{B}\| = n$ agentes y bienes. Por otro lado, siguiendo a [1] (ver definición 43), $(i, j, k) \in V^{\mathcal{B}}$ significará que el k-ésimo bit de $v(i, x_j)$ es un 1; el 0-ésimo bit será el más significativo y el $(n-1)$-ésimo el menos significativo.

Ejemplo 4.3. Consideremos la σ- estructura $\mathcal{B}$ con

$$|\mathcal{B}| = \{0, 1, 2\} \text{ y}$$

$$\begin{aligned} V^{\mathcal{B}} = \{&(0,0,2), (0,1,2), (0,2,1), \\ &(1,0,0), (1,0,1), (1,0,2), (1,1,0), (1,1,1), (1,2,0), (1,2,1)\}. \end{aligned}$$

El triple asociado a $\mathcal{B}$ está conformado por

- $A = \{0, 1, 2\}$,
- $O = \{x_0, x_1, x_2\}$, y

$\mathcal{R}$ generado por v, con

$$\begin{array}{lll} v(0, x_0) = 001, & v(1, x_0) = 111, & v(2, x_0) = 000, \\ v(0, x_1) = 001, & v(1, x_1) = 110, & v(2, x_1) = 000, \\ v(0, x_2) = 010, & v(1, x_2) = 110, \text{ y} & v(2, x_2) = 000, \end{array}$$

en binario; es decir, en el sistema decimal tenemos

$$\begin{aligned} &v(0,x_0)=1, && v(1,x_0)=7, && v(2,x_0)=0,\\ &v(0,x_1)=1, && v(1,x_1)=6, && v(2,x_1)=0,\\ &v(0,x_2)=2, && v(1,x_2)=6, \text{ y} && v(2,x_2)=0. \end{aligned}$$

Ahora plantearemos el problema.

PROBLEMA 2:

Instancia: triple $(A, O, \mathcal{R})$ con

- $A = \{0, 1, \ldots, n-1\}$,
- $O = \{x_0, \ldots, x_{n-1}\}$ y
- $\mathcal{R}$ el perfil de preferencias generado a partir de v.

Pregunta: ¿Para todo agente i y para todo bien x_j se tiene $v(i, x_j) \in \{0, 1\}$, y existe asignación PELE para $(A, O, \mathcal{R})$?

Notemos que todas las instancias de este problema tienen un número de agentes igual al número de bienes. ¿Qué hacemos si tenemos un triple $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$ con número de agentes distinto del número de bienes? En esta y otras situaciones la proposición B.2 ayuda.

4.2.1. Construcción de una sentencia que define el problema

Ahora vamos a probar que el PROBLEMA 2 está en NP. Construiremos la sentencia

$$\begin{aligned} \Phi_1 =&(\exists \mathbf{P^2})(\exists \mathbf{S^5})\\ &(\gamma_{0-1} \wedge \Phi_{\text{asignación}}(\mathbf{P^2}) \wedge \Phi_{sum-asociada}(\mathbf{S^5}, \mathbf{P^2}) \wedge \Phi_{LE}(\mathbf{S^5}) \wedge \Phi_{PE}(\mathbf{P^2})). \end{aligned}$$

A grosso modo: γ_{0-1} impondrá la restricción de que la imagen de v esté en $\{0, 1\}$, $\Phi_{LE}(\mathbf{S^5})$ asegurará que la asignación esté libre de envidia, y $\Phi_{PE}(\mathbf{P^2})$ asegurará que sea Pareto-eficiente. Como se verá, Φ_1 es equivalente a una sentencia en SO$\exists$, de manera que el problema que define está en NP.

Una de las propiedades que deben satisfacer la instancias positivas del PROBLEMA 2 es capturada por la fórmula en el siguiente lema.

Lema 4.2. *Sea $\mathcal{B}$ una σ-estructura de tamaño n. Consideremos la sentencia*

$$\gamma_{0-1} = (\forall x)(\forall y)(\forall z)(z \neq max \rightarrow \neg V^3(x, y, z)).$$

Luego, $\mathcal{B} \models \gamma_{0-1}$ si y solo si, para todo agente i y para todo bien x_j, $v(i, x_j) \in \{0, 1\}$.

Demostración. Supongamos $\mathcal{B} \models \gamma_{0-1}$. Entonces, para todo $i, j, k \in |\mathcal{B}|$, si $k \neq n-1$, el k-ésimo bit de $v(i, x_j)$ es un 0. Sean i un agente y x_j un bien. Luego $i, j \in |\mathcal{B}|$. De esta manera, para todo $k \in |\mathcal{B}|$, si $k \neq n-1$, el k-ésimo bit de $v(i, x_j)$ es un 0. Esto implica que si $k < n-1$, el k-ésimo bit de $v(i, x_j)$ es un 0; y que si $k = n-1$, es decir, en el caso del bit menos significativo, ese bit podría ser 0 o 1. Así, en el sistema decimal, $v(i, x_j) = 0$ o $v(i, x_j) = 1$. Recíprocamente, sean $i, j, k \in |\mathcal{B}|$. Luego, $i \in A$ y $x_j \in O$. Por hipótesis, $v(i, x_j) \in \{0, 1\}$. Si $k \neq n-1$, el k-ésimo bit de $v(i, x_j)$ necesariamente es un 0. Ya que $i, j, k \in |\mathcal{B}|$ fueron tomados arbitrarios, concluimos $\mathcal{B} \models \gamma_{0-1}$. □

Ahora capturaremos la noción de asignación a través de la fórmula en el siguiente lema.

Lema 4.3. *Sea $\mathcal{B}$ una σ-estructura. Consideremos la fórmula*

$$\Phi_{asignación}(\mathbf{P^2}) = (\forall x)(\forall y)(\forall u)((\mathbf{P^2}(x, y) \wedge x \neq u) \rightarrow \neg \mathbf{P^2}(u, y)).$$

Sea P una relación binaria sobre $|\mathcal{B}|$. Definamos $\pi(s) = \{x_t \in O | \, (s, t) \in P\}$ para todo $s \in |\mathcal{B}|$. Entonces, $\mathcal{B} \models \Phi_{asignación}(P)$ si y solo si π es una asignación.

Demostración. Notemos que para cualesquier $h, j \in |\mathcal{B}|$, $x_j \in \pi(h)$ si y solo si $(h, j) \in P$. Observemos además que a cada $h \in |\mathcal{B}| = A$, π asigna un paquete, $\pi(h) \in 2^O$, que podría ser vacío. Además es cierto que π, tal como está definida, asigna un solo paquete a cada agente h. De esta manera, π es una función de A en 2^O. Así, para que π sea asignación es necesario y suficiente que, para todo $i, h \in A$ con $h \neq i$, $\pi(h) \cap \pi(i) = \emptyset$. En efecto, esta condición garantiza que los bienes no sean compartidos. Pero $\pi(h) \cap \pi(i) = \emptyset$ es equivalente a $\pi(i) \subseteq \pi(h)^c$.

Notemos ahora que $\mathcal{B} \models \Phi_{\text{asignación}}(P)$ si y solo si, para todo $i, j, h \in |\mathcal{B}|$, si $(i, j) \in P$ e $i \neq h$, entonces $(h, j) \notin P$. Equivalentemente, para todo $i, j, h \in |\mathcal{B}|$, $i = h$, o $(i, j) \notin P$, o $(h, j) \notin P$, es decir, para todo $i, h \in A$, $i = h$ o para todo $x_j \in O$, $x_j \notin \pi(i)$ o $x_j \notin \pi(h)$. Lo último es equivalente a aseverar que para todo $i, h \in A$, si $i \neq h$, entonces $\pi(i) \subseteq \pi(h)^c$. Así, concluimos que $\mathcal{B} \models \Phi_{\text{asignación}}(P)$ si y solo si π es asignación. □

La fórmula en el siguiente lema permite expresar la propiedad de una asignación de ser Pareto-eficiente.

Lema 4.4. *Sean $\mathcal{B}$ una σ-estructura de tamaño n tal que $\mathcal{B} \models \gamma_{0-1}$, y P una relación tal*

que $\mathcal{B} \models \Phi_{asignación}(P)$. *Definamos* $\pi(s) = \{x_t \in O | (s,t) \in P\}$ *para todo* $s \in |\mathcal{B}|$. *Consideremos la fórmula*

$$\Phi_{PE}(\mathbf{P^2}) = (\forall y)(((\exists x)V^3(x,y,max)) \rightarrow (\exists x)(V^3(x,y,max) \wedge \mathbf{P^2}(x,y))).$$

Entonces, $\mathcal{B} \models \Phi_{PE}(P)$ *si y solo si* π *es una asignación Pareto-eficiente.*

Demostración. Primero, gracias al lema 4.3, π es asignación. Además, por el lema 4.2, el hecho de que $\mathcal{B} \models \gamma_{0-1}$ garantiza que para todo agente i y para todo bien x_j, $v(i,x_j) \in \{0,1\}$.

Ahora, $\mathcal{B} \models \Phi_{PE}(P)$ si y solo si, para todo $j \in |\mathcal{B}|$, si existe $i \in |\mathcal{B}|$ tal que el $n-1$-ésimo bit de $v(i,x_j)$ es un 1, entonces existe $k \in |\mathcal{B}|$ tal que el $n-1$-ésimo bit de $v(k,x_j)$ es un 1 y tal que $(k,j) \in P$. Equivalentemente, para todo $x_j \in O$, si existe $i \in A$ tal que $v(i,x_j) = 1$, entonces existe $k \in A$ tal que $v(k,x_j) = 1$ y tal que $x_j \in \pi(k)$. De esta manera, por la proposición 3.6, $\mathcal{B} \models \Phi_{PE}(P)$ si y solo si π es Pareto-eficiente. □

Ahora queremos expresar, a través de fórmulas lógicas, la propiedad de una asignación de estar libre de envidia. Recordemos que en el contexto actual, una asignación π está libre de envidia si y solo si $u_i(\pi(h)) \leq u_i(\pi(i))$ para cualesquier agentes i,h. Entonces requerimos poder expresar la cantidad $u_i(\pi(h))$ para cualesquier agentes i,h.

Construiremos una fórmula,

$$\Phi_{sum-asociada}(\mathbf{S^5}, \mathbf{P^2}), \tag{4.5}$$

tal que si S y P son relaciones sobre el universo de una σ-estructura $\mathcal{B}$ de tamaño n, con $\mathcal{B} \models \Phi_{\text{asignación}}(P)$, si definimos $\pi(s) = \{x_t \in O | (s,t) \in P\}$ para todo $s \in |\mathcal{B}|$, entonces $\mathcal{B} \models \Phi_{sum-asociada}(S,P)$ equivaldrá a afirmar: para todo $h,i,j,k_1,k_2 \in |\mathcal{B}|$, $(h,i,j,k_1,k_2) \in S$ si y solo si, en el orden lexicográfico de pares de números, el (k_1,k_2)-ésimo bit en la expansión binaria de la cantidad

$$\sum_{s=0}^{j} v(i,x_s) \cdot \mathbb{1}_{\pi(h)}(x_s)$$

es un 1. Hacemos notar que el $(0,0)$-ésimo bit será el más significativo, y el $(n-1,n-1)$-ésimo el menos significativo. Además, observemos que

$$u_i(\pi(h)) = \sum_{o \in \pi(h)} v(i,o) = \sum_{s=0}^{n-1} v(i,x_s) \cdot \mathbb{1}_{\pi(h)}(x_s).$$

Surge la necesidad de tener más bits disponibles que n porque, por ejemplo, para algún agente i y dos bienes distintos x_j, x_l, se podría tener que todos los bits de $v(i, x_j)$ y de $v(i, x_l)$ sean 1, de manera que si $x_j, x_l \in \pi(h)$, podemos expresar la suma $u_i(\pi(h)) = \sum_{o \in \pi(h)} v(i, o)$ sin problema. Pero, ¿bastan n^2 bits para expresar el máximo valor $u_i(\pi(h))$? El máximo valor que cualquier agente puede declarar por un bien a través de $V^{\mathcal{B}}$ es $2^n - 1$ en el sistema decimal. Dado que hay n bienes, y considerando que es posible que a algún agente h se le asigne todos los bienes, el valor máximo en cuestión es $n(2^n - 1)$. Ahora notemos que si $n \geq 2$, con $2n$ bits podemos expresar el número $2^{2n} - 1$. Pero, $2^{2n} - 1 \geq n(2^n - 1)$ para $n \geq 2$, pues $2^n > n$. Así, en vista de que $n^2 \geq 2n$ para $n \geq 2$, el número de bits disponibles es suficiente y, de hecho, habrá bits que no serán usados, es decir, estos serán 0.

La fórmula en el siguiente lema está basada en la fórmula (IV.5) de [1]. Con esta queremos tratar la inicialización de las sumas, es decir, con esta queremos expresar cada bit de la cantidad

$$\sum_{s=0}^{0} v(i, x_s) \cdot \mathbb{1}_{\pi(h)}(x_s) = v(i, x_0) \cdot \mathbb{1}_{\pi(h)}(x_0), \tag{4.6}$$

para cualesquier agentes h, i.

Lema 4.5. *Asumamos que $\mathcal{B}$ es una σ-estructura de tamaño n, y que P es una relación tal que $\mathcal{B} \models \Phi_{asignación}(P)$. Definamos $\pi(s) = \{x_t \in O | \, (s,t) \in P\}$ para todo $s \in |\mathcal{B}|$. Consideremos la fórmula*

$$\begin{aligned}\Phi_{sum_{inic}}(\mathbf{S^5}, \mathbf{P^2}) =& (\forall u)(\forall x)(\forall z_1)(\forall z_2) \\ & (\mathbf{S^5}(u, x, 0, z_1, z_2) \leftrightarrow (\mathbf{P^2}(u, 0) \wedge z_1 = max \wedge V^3(x, 0, z_2))).\end{aligned}$$

Supongamos además que S es una relación sobre $|\mathcal{B}|$. Luego, $\mathcal{B} \models \Phi_{sum_{inic}}(S, P)$ equivale a aseverar: para cualesquier $h, i, k_1, k_2 \in |\mathcal{B}|$, el (k_1, k_2)-ésimo bit de 4.6 es un 1 si y solo si $(h, i, 0, k_1, k_2) \in S$.

Demostración. El lema 4.3 asegura que π es asignación. Sean $h, i, k_1, k_2 \in |\mathcal{B}|$. Entonces $h, i \in A$ y el par (k_1, k_2) corresponde a uno de los n^2 bits de la cantidad 4.6. Si $x_0 \in \pi(h)$, 4.6 debe ser igual a la cantidad $v(i, x_0)$. Pero notemos que $v(i, x_0)$ consta de n bits, y nosotros disponemos de n^2. Luego, si $x_0 \in \pi(h)$, los últimos n bits de 4.6 deben ser los mismos de $v(i, x_0)$; el resto de bits debe ser 0. Pero los últimos n bits corresponden a las posiciones de la forma $(n - 1, k_2)$.

Ahora, supongamos que el (k_1, k_2)-ésimo bit de 4.6 es un 1. Sigue que $x_0 \in \pi(h)$ pues, en caso contrario, los n^2 bits serían 0. Además, por lo expuesto en el anterior

párrafo, el (k_1, k_2)-ésimo bit debe ser uno de los n menos significativos, es decir, $k_1 = n-1$. También debe ser cierto que el k_2-ésimo bit de $v(i, x_0)$ es un 1. Recíprocamente, si $x_0 \in \pi(h)$, $k_1 = n-1$ y el k_2-ésimo bit de $v(i, x_0)$ es un 1, necesariamente el (k_1, k_2)-ésimo bit de 4.6 es un 1. Ver el cuadro 4.1.

Supongamos $\mathcal{B} \models \Phi_{sum_{inic}}(S, P)$. Esto equivale a afirmar que para todo $h, i, k_1, k_2 \in |\mathcal{B}|$, $(h, i, 0, k_1, k_2) \in S$ si y solo si $(h, 0) \in P$, $k_1 = n-1$ y $(i, 0, k_2) \in V^{\mathcal{B}}$; y esto equivale a aseverar que para cualesquier $h, i, k_1, k_2 \in |\mathcal{B}|$, $(h, i, 0, k_1, k_2) \in S$ si y solo si $x_0 \in \pi(h)$, $k_1 = n-1$, y el k_2-ésimo bit de $v(i, x_0)$ es un 1. Tomando en cuenta el anterior párrafo, concluimos que para cualesquier $h, i, k_1, k_2 \in |\mathcal{B}|$, el (k_1, k_2)-ésimo bit de 4.6 es un 1 si y solo si $(h, i, 0, k_1, k_2) \in S$.

Recíprocamente, asumamos que para todo $h, i, k_1, k_2 \in |\mathcal{B}|$, el (k_1, k_2)-ésimo bit de 4.6 es un 1 si y solo si $(h, i, 0, k_1, k_2) \in S$. Vamos a probar que $\mathcal{B} \models \Phi_{sum_{inic}}(S, P)$. Sean $h, i, k_1, k_2 \in |\mathcal{B}|$. Por lo expuesto en los dos primeros párrafos de esta demostración, el (k_1, k_2)-ésimo bit de 4.6 es un 1 si y solo si $x_0 \in \pi(h)$, $k_1 = n-1$ y el k_2-ésimo bit de $v(i, x_0)$ es un 1. Así, $(h, i, 0, k_1, k_2) \in S$ si y solo si $x_0 \in \pi(h)$, $k_1 = n-1$ y el k_2-ésimo bit de $v(i, x_0)$ es un 1. Luego, $(h, i, 0, k_1, k_2) \in S$ si y solo si $(h, 0) \in P$, $k_1 = n-1$ y $(i, 0, k_2) \in V^{\mathcal{B}}$. Ya que $h, i, k_1, k_2 \in |\mathcal{B}|$ fueron tomados arbitrarios, la asunción hecha implica que $\mathcal{B} \models \Phi_{sum_{inic}}(S, P)$. □

Consideremos el siguiente caso.

Ejemplo 4.4. Sea $\mathcal{B}$ una σ-estructura tal que $|\mathcal{B}| = \{0, 1, 2\}$, y tal que $v(1, x_0) = 111$, $v(1, x_1) = 110$, y $v(1, x_2) = 110$ en binario. Supongamos, además, que π es una asignación con $\pi(0) = \{x_0, x_1\}$. Notemos con h e i, respectivamente, a los agentes 0 y 1. Notar que $v(1, x_0) = 7$, $v(1, x_1) = 6$, y $v(1, x_2) = 6$ en decimal, de manera que $u_i(\pi(h)) = 13$.

Ilustraremos el efecto de la fórmula en el lema 4.5, y de las siguientes, en este caso. Naturalmente esperamos aquí que el término $v(i, x_0) \cdot \mathbb{1}_{\pi(h)}(x_0)$ sea 111, pero tomemos en cuenta que el número de bits es $n^2 = 9$, de manera que esperamos que el término en cuestión sea 111 con seis 0 a la izquierda. Los bits en las posiciones $(n-1, k_2)$, para $k_2 \in |\mathcal{B}|$, son todos 1, pues $x_0 \in \pi(h)$ y todos los bits de $v(i, x_0)$ son 1. El resto de bits, aquellos en las primeras seis posiciones desde la izquierda o, en otras palabras, aquellos en las posiciones (k_1, k_2) con $k_1 < n-1$, son todos 0. Ver el cuadro 4.1.

Ya que sumaremos en el sistema binario, un paso previo a la adición de cada término es el cálculo del "carry" en cada posición pues, siguiendo a [9] (ver la prueba de la proposición 1.9), usaremos el algoritmo "carry-lookahead". En vista de que

	$k_1 < n-1$						$k_1 = n-1$		
(k_1, k_2)	(0,0)	(0,1)	(0,2	(1,0)	(1,1)	(1,2)	(2,0)	(2,1)	(2,2)
$v(i, x_0)$							1	1	1
$v(i, x_0) \cdot \mathbb{1}_{\pi(h)}(x_0)$	0	0	0	0	0	0	1	1	1

Cuadro 4.1: Inicializando la suma

será necesario comparar pares ordenados de números en el orden lexicográfico, utilizaremos las siguientes fórmulas. La primera está basada en la observación 1.32 de [9].

$$(z_1, z_2) \leq^2 (w_1, w_2) = z_1 < w_1 \vee (z_1 = w_1 \wedge z_2 \leq w_2), \text{ y}$$

$$(z_1, z_2) <^2 (w_1, w_2) = (z_1, z_2) \leq^2 (w_1, w_2) \wedge (z_1 \neq w_1 \vee z_2 \neq w_2).$$

Naturalmente, la fórmula $z_1 < w_1$ es una abreviación de $z_1 \leq w_1 \wedge z_1 \neq w_1$.

La fórmula en el siguiente lema también está basada en una fórmula en [9] (ver prueba de la proposición 1.9).

Lema 4.6. *Asumamos que $\mathcal{B}$ es una σ-estructura de tamaño n, y que $h, i, j, l \in |\mathcal{B}|$ con $j = l + 1$. Supongamos adicionalmente que S es una relación sobre $|\mathcal{B}|$ tal que para todo $f_1, f_2 \in |\mathcal{B}|$, $(h, i, l, f_1, f_2) \in S$ si y solo si el (f_1, f_2)-ésimo bit de*

$$\sum_{s=0}^{l} v(i, x_s) \cdot \mathbb{1}_{\pi(h)}(x_s) \tag{4.7}$$

es un 1. Consideremos la fórmula

$$\begin{aligned}\Phi_{carry}(\mathbf{S^5}, u, x, w, y, z_1, z_2) =& (\exists s)((z_1, z_2) <^2 (max, s) \wedge \\ & \mathbf{S^5}(u, x, w, max, s) \wedge V^3(x, y, s) \wedge \\ & (\forall r_1)(\forall r_2)(((z_1, z_2) <^2 (r_1, r_2) \wedge (r_1, r_2) <^2 (max, s)) \\ & \rightarrow ((r_1 = max \rightarrow (\mathbf{S^5}(u, x, w, r_1, r_2) \vee V^3(x, y, r_2))) \\ & \wedge (r_1 \neq max \rightarrow \mathbf{S^5}(u, x, w, r_1, r_2))))).\end{aligned}$$

Sean $k_1, k_2 \in |\mathcal{B}|$. Luego,

$$\mathcal{B} \models \Phi_{carry}(S, h, i, l, j, k_1, k_2)$$

si y solo si el carry en la posición (k_1, k_2) de la suma

$$\left(\sum_{s=0}^{l} v(i, x_s) \cdot \mathbb{1}_{\pi(h)}(x_s)\right) + v(i, x_{l+1}) \tag{4.8}$$

es un 1.

Demostración. Primero, consideremos la suma 4.8. Recordemos que la cantidad $v(i, x_{l+1})$ está descrita por n bits, pero el otro término de la suma en cuestión, es decir, la cantidad 4.7, está descrita por n^2. Así, tomando en cuenta que disponemos de n^2 posiciones de las cuales una es (k_1, k_2), si un carry es generado, necesariamente es generado en una de las últimas n. Ver el cuadro 4.2.

Lo expuesto en el anterior párrafo implica que si el carry en la posición (k_1, k_2) de 4.8 es un 1, entonces ese carry es generado en una posición a la derecha de (k_1, k_2) correspondiente a una de las últimas n, es decir, esa posición es de la forma $(n-1, d)$ para algún $d \in |\mathcal{B}|$. Que sea generado implica que el $(n-1, d)$-ésimo bit de 4.7 es un 1 y que el d-ésimo bit de $v(i, x_{l+1})$ es un 1. Además, el carry así generado debe ser propagado hasta la posición (k_1, k_2). Eso implica que en cada una de las posiciones entre (k_1, k_2) y $(n-1, d)$, al menos un bit de los sumandos es un 1; en otras palabras, para todo (e_1, e_2) entre (k_1, k_2) y $(n-1, d)$, en la posición (e_1, e_2) al menos uno de los bits de los sumandos es un 1. Si (e_1, e_2) es una de las últimas n posiciones, es decir, si $e_1 = n-1$, lo anterior implica que el (e_1, e_2)-ésimo bit de 4.7 es un 1 o el e_2-ésimo bit de $v(i, x_{l+1})$ es un 1. Si (e_1, e_2) no es una de las últimas n posiciones, necesariamente el (e_1, e_2)-ésmio bit de 4.7 debe ser un 1.

De esta manera, si el carry en la posición (k_1, k_2) de 4.8 es un 1, entonces existe $d \in |\mathcal{B}|$ tal que $(n-1, d)$ está a la derecha de (k_1, k_2), el bit $(n-1, d)$-ésimo de 4.7 es un 1, el d-ésimo bit de $v(i, x_{l+1})$ es un 1, y para todo (e_1, e_2) entre (k_1, k_2) y $(n-1, d)$, si $e_1 = n-1$, el (e_1, e_2)-ésimo bit de 4.7 es un 1 o el e_2-ésimo bit de $v(i, x_{l+1})$ es un 1 y, si $e_1 < n-1$, el (e_1, e_2)-ésimo bit de 4.7 es un 1. Equivalentemente, gracias a las hipótesis, existe $d \in |\mathcal{B}|$ tal que (k_1, k_2) es menor que $(n-1, d)$, $(h, i, l, n-1, d) \in S$, $(i, j, d) \in V^{\mathcal{B}}$, y para todo (e_1, e_2) mayor que (k_1, k_2) y menor que $(n-1, d)$, $(h, i, l, e_1, e_2) \in S$ o $(i, j, e_2) \in V^{\mathcal{B}}$ si $e_1 = n-1$, y $(h, i, l, e_1, e_2) \in S$ si $e_1 \neq n-1$. Esto es equivalente a $\mathcal{B} \models \Phi_{carry}(S, h, i, l, j, k_1, k_2)$.

Recíprocamente, $\mathcal{B} \models \Phi_{carry}(S, h, i, l, j, k_1, k_2)$ implica que un carry es generado y propagado hasta la posición (k_1, k_2), es decir, que el carry en esa posición es un 1. □

Continuando con el ejemplo 4.4, tenemos

$$\sum_{s=0}^{0} v(i, x_s) \cdot \mathbb{1}_{\pi(h)}(x_s) = v(i, x_0) \cdot \mathbb{1}_{\pi(h)}(x_0) = 000000111$$

y $v(i, x_1) = 110$ en binario. Consideremos el cuadro 4.2. En el caso $(k_1, k_2) = (2, 2)$, notemos que para todo $d \in |\mathcal{B}|$, $(2, d)$ no está a la derecha de (k_1, k_2), es decir, $(2, d)$

es menor o igual que (k_1, k_2), de modo que el carry en la posición (k_1, k_2) es 0. En el caso $(k_1, k_2) = (2, 0)$, existe $d = 1$ tal que $(2, d)$ está a la derecha de (k_1, k_2), el $(2, d)$-ésimo bit del sumando arriba es un 1, el d-ésimo bit del sumando abajo es un 1, y no existen bits entre (k_1, k_2) y $(2, d)$; de esta manera el carry en la posición (k_1, k_2) es un 1.

	$k_1 < n-1$						$k_1 = n-1$		
(k_1, k_2)	(0,0)	(0,1)	(0,2	(1,0)	(1,1)	(1,2)	(2,0)	(2,1)	(2,2)
$v(i, x_0) \cdot \mathbb{1}_{\pi(h)}(x_0)$	0	0	0	0	0	0	1	1	1
$v(i, x_1)$							1	1	0
carry	0	0	0	0	0	1	1	0	0

Cuadro 4.2: Cálculo del carry

Con lo hecho podemos proceder a operar. Las fórmulas en el siguiente lema están basadas en una fórmula en la prueba de la proposición 1.9 de [9].

Lema 4.7. *Asumamos que $\mathcal{B}$ es una σ-estructura de tamaño n, y que $h, i, j, l \in |\mathcal{B}|$ con $j = l + 1$. Supongamos además que S es una relación sobre $|\mathcal{B}|$ tal que, para todo $f_1, f_2 \in |\mathcal{B}|$, $(h, i, l, f_1, f_2) \in S$ si y solo si el (f_1, f_2)-ésimo bit de la cantidad 4.7 es un 1. Consideremos las fórmulas*

$$\Phi_{sum_{\neq max}}(\mathbf{S^5}, u, x, w, y, z_1, z_2) = \mathbf{S^5}(u, x, w, z_1, z_2) \oplus \Phi_{carry}(\mathbf{S^5}, u, x, w, y, z_1, z_2)\ y$$

$$\begin{aligned}\Phi_{sum_{max}}(\mathbf{S^5}, u, x, w, y, z_2) =& \mathbf{S^5}(u, x, w, max, z_2) \oplus V^3(x, y, z_2) \\ & \oplus \Phi_{carry}(\mathbf{S^5}, u, x, w, y, max, z_2).\end{aligned}$$

Sean $k_1, k_2 \in |\mathcal{B}|$, con $k_1 < n - 1$. Entonces

$$\mathcal{B} \models \Phi_{sum_{\neq max}}(S, h, i, l, j, k_1, k_2)$$

si y solo si el (k_1, k_2)-ésimo bit de la suma 4.8 es un 1. Además,

$$\mathcal{B} \models \Phi_{sum_{max}}(S, h, i, l, j, k_2)$$

si y solo si el $(n - 1, k_2)$-ésimo bit de la suma 4.8 es un 1.

Demostración. Primero, consideremos la suma 4.8. Esta será descrita por n^2 bits. En vista de que estamos haciendo uso del algoritmo carry-lookahead, en esta suma tenemos tres términos. Uno corresponde a $v(i, x_{l+1})$, el cual está descrito por n bits. Otro es 4.7, descrito por n^2 bits. Y el término final, también descrito por n^2 bits, corresponde al carry. Esto implica que para calcular cualquiera de los n bits menos

significativos de 4.8, debemos sumar tres bits, uno por cada término; para calcular el resto debemos sumar solo dos, aquellos correspondientes al carry y a 4.7.

Ahora,

$$\mathcal{B} \models \Phi_{sum_{max}}(S, h, i, l, j, k_2)$$

si y solo si exactamente un número impar de las condiciones

$$(h, i, l, n-1, k_2) \in S,$$
$$(i, j, k_2) \in V^{\mathcal{B}},$$
$$\mathcal{B} \models \Phi_{carry}(S, h, i, l, j, n-1, k_2)$$

es cierta. Por las hipótesis y el lema 4.6, la última afirmación equivale a que exactamente un número impar de los siguientes bits son 1: el $(n-1, k_2)$-ésimo bit de 4.7, el k_2-ésimo bit de $v(i, x_{l+1})$ y el $(n-1, k_2)$-ésimo carry de 4.8; pero esto es lo mismo que aseverar que precisamente un número impar de los tres bits involucrados en la suma 4.8 son 1. En otras palabras, el $(n-1, k_2)$-ésimo bit de la suma 4.8 es un 1.

Por otro lado,

$$\mathcal{B} \models \Phi_{sum_{\neq max}}(S, h, i, l, j, k_1, k_2)$$

si y solo si exactamente una de las dos condiciones

$$(h, i, l, k_1, k_2) \in S,$$
$$\mathcal{B} \models \Phi_{carry}(S, h, i, l, j, k_1, k_2)$$

es cierta. Gracias a las hipótesis y al lema 4.6, exactamente una de las dos condiciones mencionadas es cierta si y solo si el (k_1, k_2)-ésimo bit de 4.7 es un 1 y el (k_1, k_2)-ésimo carry de 4.8 es un 0, o el (k_1, k_2)-ésimo bit de 4.7 es un 0 y el (k_1, k_2)-ésimo carry de 4.8 es un 1; y esto equivale a afirmar que exactamente uno de los dos bits involucrados en la suma 4.8 ($k_1 < n-1$) es un 1, es decir, equivale a aseverar que el (k_1, k_2)-ésimo bit de la suma 4.8 es un 1. □

Continuando con el ejemplo 4.4, veamos el cuadro 4.3.

Ahora establecemos una fórmula que trate el resto de sumas parciales. Esta está basada en la fórmula $\theta_{12}(i, k)$ del capítulo IV de [1].

Lema 4.8. *Sean $\mathcal{B}$ una σ-estructura de tamaño n, y P una relación tal que $\mathcal{B} \models \Phi_{asignación}(P)$. Definamos $\pi(s) = \{x_t \in O | (s, t) \in P\}$ para todo $s \in |\mathcal{B}|$. Supongamos además que S es*

	$k_1 < n-1$						$k_1 = n-1$		
(k_1, k_2)	(0,0)	(0,1)	(0,2	(1,0)	(1,1)	(1,2)	(2,0)	(2,1)	(2,2)
$v(i, x_0) \cdot \mathbb{1}_{\pi(h)}(x_0)$	0	0	0	0	0	0	1	1	1
$v(i, x_1)$							1	1	0
carry	0	0	0	0	0	1	1	0	0
$v(i, x_0) \cdot \mathbb{1}_{\pi(h)}(x_0)$ $+v(i, x_1)$	0	0	0	0	0	1	1	0	1

Cuadro 4.3: Sumando el término $v(i, x_{l+1})$

una relación tal que $\mathcal{B} \models \Phi_{sum_{inic}}(S, P)$. *Consideremos la fórmula*

$$\begin{aligned}\Phi_{sum-operar}(\mathbf{S^5}, \mathbf{P^2}) =& (\forall u)(\forall x)(\forall w)(\forall y)(\forall z_2) \\ & (SUC(w, y) \rightarrow \\ & ((\mathbf{S^5}(u, x, y, max, z_2) \leftrightarrow ((\neg \mathbf{P^2}(u, y) \wedge \mathbf{S^5}(u, x, w, max, z_2)) \\ & \vee (\mathbf{P^2}(u, y) \wedge \Phi_{sum_{max}}(\mathbf{S^5}, u, x, w, y, z_2)))) \\ & \wedge (\forall z_1)(z_1 \neq max \rightarrow \\ & (\mathbf{S^5}(u, x, y, z_1, z_2) \leftrightarrow ((\neg \mathbf{P^2}(u, y) \wedge \mathbf{S^5}(u, x, w, z_1, z_2)) \\ & \vee (\mathbf{P^2}(u, y) \wedge \Phi_{sum_{\neq max}}(\mathbf{S^5}, u, x, w, y, z_1, z_2)))))))).\end{aligned}$$

Luego, $\mathcal{B} \models \Phi_{sum-operar}(S, P)$ *equivale a la siguiente afirmación: para todo* h, i, j, k_1, k_2 *en* $|\mathcal{B}|$, *con* $0 < j$, *el* (k_1, k_2)*-ésimo bit de la cantidad*

$$\sum_{s=0}^{j} v(i, x_s) \cdot \mathbb{1}_{\pi(h)}(x_s) \tag{4.9}$$

es un 1 *si y solo si* $(h, i, j, k_1, k_2) \in S$.

Demostración. Primero, gracias al lema 4.3, π es asignación. A continuación caracterizaremos, para todo h, i, j, k_1, k_2 en $|\mathcal{B}|$, con $0 < j$, la condición: el (k_1, k_2)-ésimo bit de 4.9 es un 1. Posteriormente haremos uso de esa caracterización para probar el lema.

Sean $h, i, j, k_1, k_2 \in |\mathcal{B}|$, con $0 < j$. Luego, existe $l \in |\mathcal{B}|$ tal que $j = l + 1$. Afirmamos que el (k_1, k_2)-ésimo bit de 4.9 es un 1 si y solo si $x_j \notin \pi(h)$ y el (k_1, k_2)-ésimo bit de

$$\sum_{s=0}^{l} v(i, x_s) \cdot \mathbb{1}_{\pi(h)}(x_s) \tag{4.10}$$

es un 1, o $x_j \in \pi(h)$ y el (k_1, k_2)-ésimo bit de

$$\left(\sum_{s=0}^{l} v(i, x_s) \cdot \mathbb{1}_{\pi(h)}(x_s) \right) + v(i, x_{l+1}). \tag{4.11}$$

es un 1. En efecto, asumamos que el (k_1, k_2)-ésimo bit de 4.9 es un 1, y que $x_j \in \pi(h)$ o el (k_1, k_2)-ésimo bit de 4.10 es un 0. Probaremos que $x_j \in \pi(h)$ y que el (k_1, k_2)-ésimo bit de 4.11 es un 1. Si $x_j \in \pi(h)$, entonces 4.9 es igual a 4.11, de manera que, por la asunción hecha, el (k_1, k_2)-ésimo bit de 4.11 es un 1. Por otro lado, si el (k_1, k_2)-ésimo bit de 4.10 es un 0, entonces la cantidad 4.10 es distinta de 4.9. Sigue que $x_j \in \pi(h)$ pues, en caso contrario, las cantidades 4.10 y 4.9 serían iguales. Además, ya que las cantidades 4.9 y 4.11 deben ser las mismas, el (k_1, k_2)-ésimo bit de 4.11 es un 1. Así, en todos los casos, $x_j \in \pi(h)$ y el (k_1, k_2)-ésimo bit de 4.11 es un 1.

Recíprocamente, si $x_j \notin \pi(h)$ y el (k_1, k_2)-ésimo bit de 4.10 es un 1, o si $x_j \in \pi(h)$ y el (k_1, k_2)-ésimo bit de 4.11 es un 1, entonces el (k_1, k_2)-ésimo bit de 4.9 es un 1.

($\Rightarrow$) Supongamos que $\mathcal{B} \models \Phi_{sum-operar}(S, P)$. Demostraremos, por inducción matemática sobre j, que para todo h, i, j, k_1, k_2 en $|\mathcal{B}|$, con $0 < j$, el (k_1, k_2)-ésimo bit de la cantidad 4.9 es un 1 si y solo si $(h, i, j, k_1, k_2) \in S$. Por el lema 4.5, la hipótesis $\mathcal{B} \models \Phi_{sum_{inic}}(S, P)$ garantiza que la afirmación es cierta cuando $j = 0$. A pesar de que no es parte de lo que queremos probar, esta hipótesis garantiza que tenemos un caso base. Supongamos entonces que para algún $0 \leq l < n - 1$, el (k_1, k_2)-ésimo bit de la cantidad 4.10 es un 1 si y solo si $(h, i, l, k_1, k_2) \in S$, esto para cualesquier $h, i, k_1, k_2 \in |\mathcal{B}|$.

Sea $j = l + 1$. Vamos a probar que para todo $h, i, k_1, k_2 \in |\mathcal{B}|$, el (k_1, k_2)-ésimo bit de 4.9 es un 1 si y solo si $(h, i, j, k_1, k_2) \in S$.

Tenemos que $\mathcal{B} \models \Phi_{sum-operar}(S, P)$ es equivalente a aseverar: para todo $h, i, l, j, k_2 \in |\mathcal{B}|$, si $j = l + 1$, entonces $(h, i, j, n - 1, k_2) \in S$ si y solo si $(h, j) \notin P$ y $(h, i, l, n - 1, k_2) \in S$, o $(h, j) \in P$ y $\mathcal{B} \models \Phi_{sum_{max}}(S, h, i, l, j, k_2)$; además, para todo $k_1 \in |\mathcal{B}|$ con $k_1 \neq n - 1$, $(h, i, j, k_1, k_2) \in S$ si y solo si $(h, j) \notin P$ y $(h, i, l, k_1, k_2) \in S$, o $(h, j) \in P$ y $\mathcal{B} \models \Phi_{sum_{\neq max}}(S, h, i, l, j, k_1, k_2)$.

Sean $h, i, k_1, k_2 \in |\mathcal{B}|$. Si $k_1 = n - 1$, gracias a la hipótesis de inducción, el lema 4.7 implica que el (k_1, k_2)-ésimo bit de la suma 4.11 es un 1 si y solo si $\mathcal{B} \models \Phi_{sum_{max}}(S, h, i, l, j, k_2)$. En caso de que $k_1 \neq n - 1$, el mismo lema asegura que el (k_1, k_2)-ésimo bit de la suma 4.11 es un 1 si y solo si $\mathcal{B} \models \Phi_{sum_{\neq max}}(S, h, i, l, j, k_1, k_2)$.

Por lo expuesto en los dos párrafos anteriores, y gracias a la hipótesis inductiva, tanto si $k_1 = n - 1$ como si no, $(h, i, j, k_1, k_2) \in S$ si y solo si $x_j \notin \pi(h)$ y el (k_1, k_2)-

ésimo bit de 4.10 es un 1, o $x_j \in \pi(h)$ y el (k_1, k_2)-ésimo bit de 4.11 es un 1. Entonces, por la caracterización hecha al inicio, $(h, i, j, k_1, k_2) \in S$ si y solo si el (k_1, k_2)-ésimo bit de 4.9 es un 1; esto es precisamente lo que queríamos probar.

Ya que $h, i, k_1, k_2 \in |\mathcal{B}|$ fueron tomados arbitrarios, y gracias al principio de inducción matemática, concluimos que $\mathcal{B} \models \Phi_{sum-operar}(S, P)$ implica: para cualesquier $h, i, j, k_1, k_2 \in |\mathcal{B}|$, con $0 < j$, el (k_1, k_2)-ésimo bit de 4.9 es un 1 si y solo si $(h, i, j, k_1, k_2) \in S$.

($\Leftarrow$) Supongamos ahora que para cualesquier $h, i, j, k_1, k_2 \in |\mathcal{B}|$, con $0 < j$, el (k_1, k_2)-ésimo bit de 4.9 es un 1 si y solo si $(h, i, j, k_1, k_2) \in S$. Vamos a probar que $\mathcal{B} \models \Phi_{sum-operar}(S, P)$.

Sean $h, i, l, j, k_2 \in |\mathcal{B}|$, con $j = l + 1$. Por la asunción hecha, y gracias a la caracterización hecha al principio de la demostración del lema, para todo $k_1 \in |\mathcal{B}|$, $(h, i, j, k_1, k_2) \in S$ si y solo si $x_j \notin \pi(h)$ y el (k_1, k_2)-ésimo bit de 4.10 es un 1, o $x_j \in \pi(h)$ y el (k_1, k_2)-ésimo bit de 4.11 es un 1. Así, otra vez por la asunción, y gracias a los lemas 4.5 y 4.7, $(h, i, j, n-1, k_2) \in S$ si y solo si $(h, j) \notin P$ y $(h, i, l, n-1, k_2) \in S$, o $(h, j) \in P$ y $\mathcal{B} \models \Phi_{sum_{max}}(S, h, i, l, j, k_2)$. Además, si $k_1 \in |\mathcal{B}|$ con $k_1 \neq n-1$, $(h, i, j, k_1, k_2) \in S$ si y solo si $(h, j) \notin P$ y $(h, i, l, k_1, k_2) \in S$, o $(h, j) \in P$ y $\mathcal{B} \models \Phi_{sum_{\neq max}}(S, h, i, l, j, k_1, k_2)$. Ya que $h, i, l, j, k_2 \in |\mathcal{B}|$ fueron tomados arbitrarios, concluimos que $\mathcal{B} \models \Phi_{sum-operar}(S, P)$. □

Siguiendo con el ejemplo 4.4, en vista de que $x_1 \in \pi(h)$, la última fila en el cuadro 4.3 es precisamente

$$\sum_{s=0}^{1} v(i, x_s) \cdot \mathbb{1}_{\pi(h)}(x_s) = v(i, x_0) \cdot \mathbb{1}_{\pi(h)}(x_0) + v(i, x_1) \cdot \mathbb{1}_{\pi(h)}(x_1).$$

Ver el cuadro 4.4. Además, la cantidad

$$u_i(\pi(h)) = \sum_{s=0}^{2} v(i, x_s) \cdot \mathbb{1}_{\pi(h)}(x_s)$$

debe ser igual a la anterior, pues $x_2 \notin \pi(h)$; esa cantidad es 13, de manera que el cálculo es correcto.

Podemos entonces tomar la fórmula 4.5 como

$$\Phi_{sum-asociada}(\mathbf{S^5}, \mathbf{P^2}) = \Phi_{sum_{inic}}(\mathbf{S^5}, \mathbf{P^2}) \wedge \Phi_{sum-operar}(\mathbf{S^5}, \mathbf{P^2}).$$

En efecto, tenemos lo siguiente.

Lema 4.9. *Sean $\mathcal{B}$ una σ-estructura, P una relación tal que $\mathcal{B} \models \Phi_{asignación}(P)$, y S una*

	$k_1 < n-1$						$k_1 = n-1$		
(k_1, k_2)	(0,0)	(0,1)	(0,2	(1,0)	(1,1)	(1,2)	(2,0)	(2,1)	(2,2)
$v(i, x_0) \cdot \mathbb{1}_{\pi(h)}(x_0)$	0	0	0	0	0	0	1	1	1
$v(i, x_1)$							1	1	0
carry	0	0	0	0	0	1	1	0	0
$v(i, x_0) \cdot \mathbb{1}_{\pi(h)}(x_0)$ $+v(i, x_1)$	0	0	0	0	0	1	1	0	1
$v(i, x_0) \cdot \mathbb{1}_{\pi(h)}(x_0)$ $+$ $v(i, x_1) \cdot \mathbb{1}_{\pi(h)}(x_1)$	0	0	0	0	0	1	1	0	1

Cuadro 4.4: Sumando el término $v(i, x_{l+1}) \cdot \mathbb{1}_{\pi(h)}(x_{l+1})$

relación sobre $|\mathcal{B}|$. *Definamos* $\pi(s) = \{x_t \in O | \ (s,t) \in P\}$ *para todo* $s \in |\mathcal{B}|$. *Luego,*

$$\mathcal{B} \models \Phi_{sum-asociada}(S, P)$$

es equivalente a afirmar que, para todo $h, i, j, k_1, k_2 \in |\mathcal{B}|$, $(h, i, j, k_1, k_2) \in S$ *si y solo si, en el orden lexicográfico de pares de números, el* (k_1, k_2)*-ésimo bit en la expansión binaria de la cantidad 4.9 es un* 1.

Demostración. ($\Rightarrow$) Supongamos $\mathcal{B} \models \Phi_{sum-asociada}(S, P)$. Sean $h, i, j, k_1, k_2 \in |\mathcal{B}|$. La asunción es equivalente a

$$\mathcal{B} \models \Phi_{sum_{inic}}(S, P) \text{ y}$$
$$\mathcal{B} \models \Phi_{sum-operar}(S, P).$$

Entonces, por el lema 4.5, $(h, i, j, k_1, k_2) \in S$ si y solo si el (k_1, k_2)-ésimo bit en la expansión binaria de la cantidad 4.9 es un 1, siempre que $j = 0$. Por otro lado, el lema 4.8 garantiza la condición cuando $0 < j$.

($\Leftarrow$) Asumamos que, para todo $h, i, j, k_1, k_2 \in |\mathcal{B}|$, $(h, i, j, k_1, k_2) \in S$ si y solo si el (k_1, k_2)-ésimo bit en la expansión binaria de la cantidad 4.9 es un 1. Cuando $j = 0$, gracias a las hipótesis, el lema 4.5 asegura que $\mathcal{B} \models \Phi_{sum_{inic}}(S, P)$, pero esta condición y el lema 4.8 nos permiten inferir que $\mathcal{B} \models \Phi_{sum-operar}(S, P)$. De esta manera, $\mathcal{B} \models \Phi_{sum-asociada}(S, P)$. □

A continuación proponemos fórmulas que permiten expresar $u_i(\pi'(h)) < u_i(\pi(i))$ y $u_i(\pi'(h)) = u_i(\pi(i))$, con π y π' asignaciones no necesariamente iguales. Esto nos ayudará en la siguiente sección a capturar la noción de Pareto-eficiencia; en la presente nos permitirá expresar la propiedad de una asignación de estar libre de envidia. La primera de las fórmulas está basada en la fórmula LESS(A, B) en el

capítulo 1 de [9].

Lema 4.10. *Sean $\mathcal{B}$ una σ-estructura de tamaño n, y P, Q, S, T relaciones tales que*

$$\mathcal{B} \models \Phi_{asignación}(P), \tag{4.12}$$
$$\mathcal{B} \models \Phi_{asignación}(Q), \tag{4.13}$$
$$\mathcal{B} \models \Phi_{sum-asociada}(S, P) \text{ y}$$
$$\mathcal{B} \models \Phi_{sum-asociada}(T, Q).$$

Definamos, para todo $s \in |\mathcal{B}|$,

$$\pi(s) = \{x_t \in O |\, (s,t) \in P\} \text{ y}$$
$$\pi'(s) = \{x_t \in O |\, (s,t) \in Q\}.$$

Consideremos las fórmulas

$$\begin{aligned}\Phi_{compara_{menor}}(\mathbf{T^5}, \mathbf{S^5}, u, x) =& (\exists z_1)(\exists z_2) \\ & (\neg \mathbf{T^5}(u, x, max, z_1, z_2) \wedge \mathbf{S^5}(x, x, max, z_1, z_2) \\ & \wedge (\forall r_1)(\forall r_2)((r_1, r_2) <^2 (z_1, z_2) \rightarrow \\ & (\mathbf{T^5}(u, x, max, r_1, r_2) \rightarrow \mathbf{S^5}(x, x, max, r_1, r_2)))) \text{ y}\end{aligned}$$

$$\begin{aligned}\Phi_{compara_{=}}(\mathbf{T^5}, \mathbf{S^5}, u, x) =& (\forall z_1)(\forall z_2) \\ & (\mathbf{T^5}(u, x, max, z_1, z_2) \leftrightarrow \mathbf{S^5}(x, x, max, z_1, z_2)).\end{aligned}$$

Sean $h, i \in |\mathcal{B}|$. Entonces,

$$\mathcal{B} \models \Phi_{compara_{menor}}(T, S, h, i) \tag{4.14}$$

si y solo si $u_i(\pi'(h)) < u_i(\pi(i))$. Además,

$$\mathcal{B} \models \Phi_{compara_{=}}(T, S, h, i) \tag{4.15}$$

si y solo si $u_i(\pi'(h)) = u_i(\pi(i))$.

Demostración. Gracias a las hipótesis 4.12 y 4.13 y al lema 4.3, π y π' son asignaciones.

Ahora, tenemos 4.14 si y solo si existen $k_1, k_2 \in |\mathcal{B}|$ tales que $(h, i, n-1, k_1, k_2) \notin T$, $(i, i, n-1, k_1, k_2) \in S$ y, para todo $e_1, e_2 \in |\mathcal{B}|$, si (e_1, e_2) es menor que (k_1, k_2) en el orden lexicográfico, entonces $(h, i, n-1, e_1, e_2) \in T$ solo si $(i, i, n-1, e_1, e_2) \in S$.

Gracias al lema 4.9 y a las hipótesis, sigue que 4.14 es cierta si y solo si existen $k_1, k_2 \in |\mathcal{B}|$ tales que el bit en la posición (k_1, k_2) de

$$\sum_{s=0}^{n-1} v(i, x_s) \cdot \mathbb{1}_{\pi'(h)}(x_s) = u_i(\pi'(h)) \tag{4.16}$$

es un 0, el bit en la posición (k_1, k_2) de

$$\sum_{s=0}^{n-1} v(i, x_s) \cdot \mathbb{1}_{\pi(i)}(x_s) = u_i(\pi(i)) \tag{4.17}$$

es un 1, y para todo $e_1, e_2 \in |\mathcal{B}|$, si (e_1, e_2) es menor que (k_1, k_2) en el orden lexicográfico de pares de números, entonces el (e_1, e_2)-ésimo bit de 4.16 es un 1 solo si el (e_1, e_2)-ésimo bit de 4.17 es un 1. Lo último equivale a afirmar que existe un bit con valor 0 en $u_i(\pi'(h))$, tal que en la misma posición, el bit correspondiente de $u_i(\pi(i))$ es un 1, y para toda posición a la izquierda, si el bit correspondiente de $u_i(\pi'(h))$ es 1, entonces el de $u_i(\pi(i))$ también es un 1; en otras palabras, $u_i(\pi'(h)) < u_i(\pi(i))$.

Por otro lado, 4.15 equivale a: para todo $k_1, k_2 \in |\mathcal{B}|$, $(h, i, n-1, k_1, k_2) \in T$ si y solo si $(i, i, n-1, k_1, k_2) \in S$. De esta manera, las hipótesis y el lema 4.9 permiten inferir que 4.15 equivale a aseverar que, para todo $k_1, k_2 \in |\mathcal{B}|$, el bit en la posición (k_1, k_2) de 4.16 es un 1 si y solo si el bit en la posición (k_1, k_2) de 4.17 es un 1. En conclusión, 4.15 equivale a afirmar que $u_i(\pi'(h)) = u_i(\pi(i))$. □

Consideremos las siguientes fórmulas.

$$\Phi_{compara_{\leq}}(\mathbf{S^5}, u, x) = \Phi_{compara_{menor}}(\mathbf{S^5}, \mathbf{S^5}, u, x) \vee \Phi_{compara_{=}}(\mathbf{S^5}, \mathbf{S^5}, u, x) \text{ y} \tag{4.18}$$

$$\Phi_{LE}(\mathbf{S^5}) = (\forall u)(\forall x)\Phi_{compara_{\leq}}(\mathbf{S^5}, u, x). \tag{4.19}$$

Hacemos notar que para construir la fórmula 4.18, se ha reemplazado $\mathbf{T^5}$ por $\mathbf{S^5}$ en las fórmulas del lema anterior. Así, por ese lema, la fórmula 4.18 nos permitirá expresar $u_i(\pi(h)) \leq u_i(\pi(i))$, y la fórmula 4.19 nos permitirá capturar la propiedad de una asignación de estar libre de envidia.

Lema 4.11. *Sean $\mathcal{B}$ una σ-estructura de tamaño n, y P y S relaciones tales que $\mathcal{B} \models \Phi_{asignación}(P)$ y $\mathcal{B} \models \Phi_{sum-asociada}(S, P)$. Definamos $\pi(s) = \{x_t \in O | (s, t) \in P\}$ para todo $s \in |\mathcal{B}|$. Consideremos las fórmulas 4.18 y 4.19. Entonces,*

$$\mathcal{B} \models \Phi_{LE}(S)$$

si y solo si π está libre de envidia.

Demostración. La afirmación $\mathcal{B} \models \Phi_{LE}(S)$ equivale a la siguiente: para todo $h, i \in |\mathcal{B}|$, $\mathcal{B} \models \Phi_{compara_{\leq}}(S, h, i)$; es decir, equivale a aseverar que, para todo $h, i \in |\mathcal{B}|$,

$$\begin{aligned}\mathcal{B} \models &\Phi_{compara_{menor}}(S, S, h, i) \text{ o}\\ \mathcal{B} \models &\Phi_{compara_{=}}(S, S, h, i).\end{aligned}$$

De esta manera, por el lema 4.10, $\mathcal{B} \models \Phi_{LE}(S)$ si y solo si, para todo $h, i \in |\mathcal{B}|$, $u_i(\pi(h)) \leq u_i(\pi(i))$, es decir, si y solo si π está libre de envidia. □

Teorema 4.2. *El PROBLEMA 2 está en NP.*

Demostración. Una sentencia que define el problema es

$$\begin{aligned}\Phi_1 =&(\exists \mathbf{P^2})(\exists \mathbf{S^5})\\ &(\gamma_{0-1} \wedge \Phi_{\text{asignación}}(\mathbf{P^2}) \wedge \Phi_{sum-asociada}(\mathbf{S^5}, \mathbf{P^2}) \wedge \Phi_{LE}(\mathbf{S^5}) \wedge \Phi_{PE}(\mathbf{P^2})).\end{aligned}$$

En efecto, supongamos que $\mathcal{B}$ es una σ-estructura tal que $\mathcal{B} \models \Phi_1$. Sigue que existen relaciones P, S sobre $|\mathcal{B}|$ tales que

$$\mathcal{B} \models \gamma_{0-1}, \tag{4.20}$$

$$\mathcal{B} \models \Phi_{\text{asignación}}(P), \tag{4.21}$$

$$\mathcal{B} \models \Phi_{sum-asociada}(S, P), \tag{4.22}$$

$$\mathcal{B} \models \Phi_{LE}(S) \text{ y} \tag{4.23}$$

$$\mathcal{B} \models \Phi_{PE}(P). \tag{4.24}$$

Definamos $\pi(s) = \{x_t \in O |\, (s, t) \in P\}$ para todo $s \in |\mathcal{B}|$. Vamos a probar que para todo agente i y para todo bien x_j, $v(i, x_j) \in \{0, 1\}$, y que π es PELE.

El lema 4.2 y la condición 4.20 aseguran que para todo $i, j \in |\mathcal{B}|$, $v(i, x_j) \in \{0, 1\}$. Por otro lado, el lema 4.3 y 4.21 garantizan que π es asignación. Además, las condiciones 4.21, 4.22 y 4.23, y el lema 4.11, implican que π está libre de envidia. Finalmente, 4.20, 4.21, 4.24, y el lema 4.4 nos permiten inferir que π es Pareto-eficiente. Así, si $\mathcal{B} \models \Phi_1$, entonces el triple $(A, O, \mathcal{R})$ asociado a $\mathcal{B}$ es una instancia positiva del PROBLEMA 2.

Recíprocamente, supongamos que $\mathcal{B}$ es una σ-estructura con triple asociado $(A, O, \mathcal{R})$ que es una instancia positiva del PROBLEMA 2. Luego, para todo agente i y para todo bien x_j, $v(i, x_j) \in \{0, 1\}$, y existe asignación PELE π para $(A, O, \mathcal{R})$. Definamos $P = \{(i, j) \in |\mathcal{B}|^2 |\, x_j \in \pi(i)\}$. De esta manera, $\pi(i) = \{x_j \in O |\, (i, j) \in P\}$ para todo $i \in |\mathcal{B}|$. Por otro lado, notemos que podemos construir una relación S tal que, para

todo $h, i, j, k_1, k_2 \in |\mathcal{B}|$, $(h, i, j, k_1, k_2) \in S$ si y solo si, en el orden lexicográfico de pares de números, el (k_1, k_2)-ésimo bit en la expansión binaria de la cantidad 4.9 es un 1.

Ahora, el hecho de que para todo agente i y para todo bien x_j, $v(i, x_j) \in \{0, 1\}$, junto al lema 4.2, garantizan que la condición 4.20 es cierta. Además, gracias al lema 4.3, la condición 4.21 es verdadera. Por la manera en la que se ha tomado S y por el lema 4.9 se tiene 4.22. Ahora, el lema 4.11 y el hecho de que π está libre de envidia implican 4.23. El que π sea Pareto-eficiente y el lema 4.4 permiten inferir 4.24. Entonces la asunción hecha tiene como consecuencia que $\mathcal{B} \models \Phi_1$.

Concluimos que $\mathcal{B} \models \Phi_1$ si y solo si el triple $(A, O, \mathcal{R})$ asociado a $\mathcal{B}$ es una instancia positiva del PROBLEMA 2.

Φ_1 es equivalente a una sentencia existencial de segundo orden. En efecto, usando la prueba del teorema 3.5.11 en [5] (solo necesitamos mover los cuantificadores de primer orden al principio, luego de $(\exists \mathbf{P^2})(\exists \mathbf{S^5})$), existe una fórmula $\Phi_1' \in \text{SO}\exists$ tal que $\Phi_1 \equiv \Phi_1'$. Por el teorema 2.1, el problema que define está en NP. □

4.2.2. Reducción

Ahora probaremos que el PROBLEMA 2 es NP-hard vía fops; lo haremos basándonos en la proposición 21 de [4]. Plantearemos una reducción desde una versión modificada del siguiente problema.

PROBLEMA: 3-DIMENSIONAL-MATCHING (3DM)

Instancia: Un conjunto C con n elementos y un conjunto de triples $M \subseteq C^3$

Pregunta: ¿Existe $M' \subseteq M$ tal que, para todo $i \in C$, existen triples

$$(i, a_1, b_1), (a_2, i, b_2), (a_3, b_3, i) \in M',$$

y tal que para todo par de triples diferentes

$$(i_1, j_1, k_1), (i_2, j_2, k_2) \in M' \implies i_1 \neq i_2, j_1 \neq j_2 \text{ y } k_1 \neq k_2?$$

El vocabulario usado para 3DM es

$$\omega = \langle D^3 \rangle.$$

Si $\mathcal{A} = \langle |\mathcal{A}|, D^{\mathcal{A}} \rangle$ es una ω-estructura, $|\mathcal{A}|$ es C y $D^{\mathcal{A}}$ es M. 3DM es completo vía fops [10] (teorema 4.74).

En la versión modificada que usaremos, se asume que toda ω-estructura $\mathcal{A}$ satisface: para todo $i \in |\mathcal{A}|$ existen $j_l, k_l \in |\mathcal{A}|$, con $l = 1, 2, 3$, tales que

$$(i, j_1, k_1), (j_2, i, k_2), (j_3, k_3, i) \in D^{\mathcal{A}}.$$

Esta versión, que notaremos con 3DMx, es un problema NP-completo vía fops (ver la proposición B.3).

Comenzaremos planteando las fórmulas que definen la reducción de primer orden, y luego presentaremos los argumentos que permiten inferir que esta es en efecto una proyección.

Definamos las fórmulas

$$\begin{aligned}
agente_{001} = &\quad x_1 = 0 \wedge x_2 = 0 \wedge x_3 = 1 \quad \wedge \quad D(x_4, x_5, x_6),\\
agente_{010} = &\quad x_1 = 0 \wedge x_2 = 1 \wedge x_3 = 0 \quad \wedge \quad D(x_4, x_5, x_6),\\
agente_{100} = &\quad x_1 = 1 \wedge x_2 = 0 \wedge x_3 = 0 \quad \wedge \quad D(x_4, x_5, x_6),
\end{aligned}$$

$$\begin{aligned}
bien_{110} = &\quad y_1 = 1 \wedge y_2 = 1 \wedge y_3 = 0 \quad \wedge \quad y_4 = x_4 \wedge y_5 = x_5 \wedge y_6 = x_6,\\
bien_{101} = &\quad y_1 = 1 \wedge y_2 = 0 \wedge y_3 = 1 \quad \wedge \quad y_4 = x_4 \wedge y_5 = x_5 \wedge y_6 = x_6,\\
bien_{011} = &\quad y_1 = 0 \wedge y_2 = 1 \wedge y_3 = 1 \quad \wedge \quad y_4 = x_4 \wedge y_5 = x_5 \wedge y_6 = x_6,
\end{aligned}$$

$$\begin{aligned}
bien_{001} = &\quad y_1 = 0 \wedge y_2 = 0 \wedge y_3 = 1 \quad \wedge \quad y_4 = 0 \wedge y_5 = 0 \wedge y_6 = x_6,\\
bien_{010} = &\quad y_1 = 0 \wedge y_2 = 1 \wedge y_3 = 0 \quad \wedge \quad y_4 = 0 \wedge y_5 = x_5 \wedge y_6 = 0,\\
bien_{100} = &\quad y_1 = 1 \wedge y_2 = 0 \wedge y_3 = 0 \quad \wedge \quad y_4 = x_4 \wedge y_5 = 0 \wedge y_6 = 0, \text{ y}
\end{aligned}$$

$$valor = \quad z_1 = max \wedge z_2 = max \wedge z_3 = max \wedge z_4 = max \wedge z_5 = max \wedge z_6 = max.$$

Notemos que todas las fórmulas, excepto las tres primeras, son numéricas. Observemos además, que hay una correspondencia entre el subíndice de la expresión $agente_{001}$ y la subfórmula

$$x_1 = 0 \wedge x_2 = 0 \wedge x_3 = 1$$

de la fórmula $agente_{001}$. Existen correspondencias de este tipo en todas las fórmulas definidas excepto en la última, $valor$.

Ahora, sean

$$\begin{aligned}\bar{x} =&(x_1, x_2, x_3, x_4, x_5, x_6),\\ \bar{y} =&(y_1, y_2, y_3, y_4, y_5, y_6), \text{ y}\\ \bar{z} =&(z_1, z_2, z_3, z_4, z_5, z_6),\end{aligned}$$

y consideremos la reducción de primer orden 6-aria ρ : STRUC$[\omega] \to$ STRUC$[\sigma]$, definida por

$$\varphi_0 = \textbf{true} \text{ y}$$

$$\begin{aligned}\varphi_1(\bar{x}, \bar{y}, \bar{z}) = \quad & \textbf{false} \quad \vee\\ & (agente_{001} \wedge bien_{110} \wedge valor) \quad \vee\\ & (agente_{001} \wedge bien_{101} \wedge valor) \quad \vee\\ & (agente_{001} \wedge bien_{011} \wedge valor) \quad \vee\\ & (agente_{001} \wedge bien_{001} \wedge valor) \quad \vee\\ & (agente_{001} \wedge bien_{010} \wedge valor) \quad \vee\\ & (agente_{001} \wedge bien_{100} \wedge valor) \quad \vee\\ & (agente_{010} \wedge bien_{110} \wedge valor) \quad \vee\\ & (agente_{010} \wedge bien_{101} \wedge valor) \quad \vee\\ & (agente_{010} \wedge bien_{011} \wedge valor) \quad \vee\\ & (agente_{010} \wedge bien_{001} \wedge valor) \quad \vee\\ & (agente_{010} \wedge bien_{010} \wedge valor) \quad \vee\\ & (agente_{010} \wedge bien_{100} \wedge valor) \quad \vee\\ & (agente_{100} \wedge bien_{110} \wedge valor) \quad \vee\\ & (agente_{100} \wedge bien_{101} \wedge valor) \quad \vee\\ & (agente_{100} \wedge bien_{011} \wedge valor) \quad \vee\\ & (agente_{100} \wedge bien_{001} \wedge valor) \quad \vee\\ & (agente_{100} \wedge bien_{010} \wedge valor) \quad \vee\\ & (agente_{100} \wedge bien_{100} \wedge valor).\end{aligned}$$

Notemos que la definida es una fop (ver definición 2.24). En efecto, φ_0 es numérica.

Además, φ_1 puede ser reescrita en la forma

$$\alpha_0 \vee (\alpha_1 \wedge \lambda_1) \vee (\alpha_2 \wedge \lambda_2) \vee \ldots \vee (\alpha_{18} \wedge \lambda_{18}),$$

con α_s una fórmula numérica para todo s, y $\lambda_t = D(x_4, x_5, x_6)$ para todo t. Por ejemplo, $\alpha_1 \wedge \lambda_1$ se obtiene a partir de la fórmula

$$agente_{001} \wedge bien_{110} \wedge valor,$$

con

$$\begin{aligned} \lambda_1 = &\quad D(x_4, x_5, x_6) \text{ y} \\ \alpha_1 = &\quad x_1 = 0 \wedge x_2 = 0 \wedge x_3 = 1 \wedge bien_{110} \wedge valor. \end{aligned}$$

También es cierto que dadas las fórmulas

$$\alpha_s \wedge \lambda_s,$$
$$\alpha_t \wedge \lambda_t,$$

con $s \neq t$, las subfórmulas α_s y α_t son mutuamente excluyentes. Por ejemplo, las fórmulas

$$\alpha_{17} \wedge \lambda_{17},$$
$$\alpha_{18} \wedge \lambda_{18},$$

son obtenidas a partir de la dos últimas conjunciones de φ_1, es decir, a partir de

$$agente_{100} \wedge bien_{010} \wedge valor \text{ y}$$
$$agente_{100} \wedge bien_{100} \wedge valor,$$

respectivamente. Notemos que ambas comienzan con la fórmula $agente_{100}$. En este caso, $bien_{010}$ y $bien_{100}$ garantizan que las fórmulas numéricas α_{17} y α_{18} sean mutuamente excluyentes.

Consideremos ahora

$$agente_{010} \wedge bien_{100} \wedge valor \text{ y}$$
$$agente_{100} \wedge bien_{100} \wedge valor.$$

La subfórmula $bien_{100}$ aparece en ambas. Aquí, son las partes numéricas de $agente_{010}$ y de $agente_{100}$ las que garantizan la condición.

Ahora, recordemos que φ_0 define el universo de las σ-estructuras en la imagen de la fop ρ, y que $\varphi_1(\bar{x}, \bar{y}, \bar{z})$ define las relaciones de entrada de esas estructuras. Dada una ω-estructura de tamaño n,

$$\mathcal{A} = \langle |\mathcal{A}|, D^{\mathcal{A}} \rangle,$$

ρ la transforma en una σ-estructura

$$\rho(\mathcal{A}) = \langle |\rho(\mathcal{A})|, V^{\rho(\mathcal{A})} \rangle.$$

Por la definición de φ_0,

$$|\rho(\mathcal{A})| = |A|^6,$$

es decir, $\rho(\mathcal{A})$ tiene asociado un triple $(A, O, \mathcal{R})$ con n^6 agentes y con el mismo número de bienes. Por otro lado, por la definición de $\varphi_1(\bar{x}, \bar{y}, \bar{z})$, la relación $V^{\rho(\mathcal{A})}$ consiste de los triples, y solo de los triples,

$$\begin{aligned}
&(0,0,1,i,j,k, \quad 1,1,0,i,j,k, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(0,0,1,i,j,k, \quad 1,0,1,i,j,k, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(0,0,1,i,j,k, \quad 0,1,1,i,j,k, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(0,0,1,i,j,k, \quad 0,0,1,0,0,k, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(0,0,1,i,j,k, \quad 0,1,0,0,j,0, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(0,0,1,i,j,k, \quad 1,0,0,i,0,0, \quad n-1,n-1,n-1,n-1,n-1,n-1),
\end{aligned}$$

$$\begin{aligned}
&(0,1,0,i,j,k, \quad 1,1,0,i,j,k, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(0,1,0,i,j,k, \quad 1,0,1,i,j,k, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(0,1,0,i,j,k, \quad 0,1,1,i,j,k, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(0,1,0,i,j,k, \quad 0,0,1,0,0,k, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(0,1,0,i,j,k, \quad 0,1,0,0,j,0, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(0,1,0,i,j,k, \quad 1,0,0,i,0,0, \quad n-1,n-1,n-1,n-1,n-1,n-1),
\end{aligned}$$

$$
\begin{aligned}
&(1,0,0,i,j,k, \quad 1,1,0,i,j,k, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(1,0,0,i,j,k, \quad 1,0,1,i,j,k, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(1,0,0,i,j,k, \quad 0,1,1,i,j,k, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(1,0,0,i,j,k, \quad 0,0,1,0,0,k, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(1,0,0,i,j,k, \quad 0,1,0,0,j,0, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(1,0,0,i,j,k, \quad 1,0,0,i,0,0, \quad n-1,n-1,n-1,n-1,n-1,n-1),
\end{aligned}
$$

con $(i,j,k) \in D^{\mathcal{A}}$.

A partir de aquí, en esta subsección adoptaremos la siguiente convención: no notaremos a los bienes con los símbolos x_s; en vez, los notaremos directamente con s, mencionando que son bienes. Es posible hacer esto ya que hay una biyección entre el conjunto O asociado a una σ-estructura y el universo de esta.

Ahora, notemos que si $\mathcal{A}$ es una ω-estructura de tamaño n, en vista de que corresponde a una instancia de 3DMx, entonces $D^{\mathcal{A}}$ es no vacía. Luego, por cada $(i,j,k) \in D^{\mathcal{A}}$ se crean, en $\rho(\mathcal{A})$, tres agentes no indiferentes:

$$(0,0,1,i,j,k), \quad (0,1,0,i,j,k), \quad (1,0,0,i,j,k).$$

Todos estos tres agentes declaran un valor de 1, en el sistema decimal, por los siguientes bienes:

$$
\begin{aligned}
&(1,1,0,i,j,k), \quad (1,0,1,i,j,k), \quad (0,1,1,i,j,k),\\
&(0,0,1,0,0,k), \quad (0,1,0,0,j,0), \quad (1,0,0,i,0,0).
\end{aligned}
$$

Declaran un valor de 1, pues la tercera compononente de todos los elementos de $V^{\rho(\mathcal{A})}$ es

$$(n-1,n-1,n-1,n-1,n-1,n-1),$$

correspondiente al valor máximo en $|\rho(\mathcal{A})|$, es decir, al bit menos significativo. Por el resto de bienes, los agentes mencionados declaran un valor de 0. En conclusión, los tres agentes considerados tienen las mismas preferencias.

Ahora, sea I el conjunto conformado por el resto de agentes, es decir, por aquellos distintos de

$$(0,0,1,i,j,k), \quad (0,1,0,i,j,k), \quad (1,0,0,i,j,k),$$

para todo $(i,j,k) \in D^{\mathcal{A}}$. Entonces los agentes en I son indiferentes, pues declaran un valor de 0 por cada bien (ver la sección 3.2). De esta manera, el triple $(A,O,\mathcal{R})$ asociado a $\rho(\mathcal{A})$, es tal que la función v toma valores en el conjunto $\{0,1\}$.

Los bienes deseados, es decir, aquellos por los cuales al menos un agente declara un valor de 1, pueden ser divididos en dos categorías. La primera categoría consiste de los bienes

$$(1,1,0,i,j,k), \quad (1,0,1,i,j,k), \quad (0,1,1,i,j,k),$$

con $(i,j,k) \in D^{\mathcal{A}}$; es decir, cada $(i,j,k) \in D^{\mathcal{A}}$ tiene asociados tres bienes en esta categoría.

La otra categoría consiste de los objetos

$$(0,0,1,0,0,i), \quad (0,1,0,0,i,0), \quad (1,0,0,i,0,0),$$

con $i \in |\mathcal{A}|$. Estos son objetos deseados pues, dado $i \in |\mathcal{A}|$, existen $j_l, k_l \in |\mathcal{A}|$ con $l = 1,2,3$, tales que

$$(i,j_1,k_1), (j_2,i,k_2), (j_3,k_3,i) \in D^{\mathcal{A}}.$$

En efecto, $\mathcal{A}$ es una instancia de 3DMx. Luego, los triples

$$\begin{aligned}
&(1,0,0,j_3,k_3,i, \quad 0,0,1,0,0,i, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(1,0,0,j_2,i,k_2, \quad 0,1,0,0,i,0, \quad n-1,n-1,n-1,n-1,n-1,n-1),\\
&(1,0,0,i,j_1,k_1, \quad 1,0,0,i,0,0, \quad n-1,n-1,n-1,n-1,n-1,n-1),
\end{aligned}$$

están en $V^{\rho(\mathcal{A})}$. En conclusión, por cada $i \in |\mathcal{A}|$, hay tres bienes en la segunda categoría.

El resto de bienes, es decir, los no deseados, son de valor nulo (ver la sección 3.2).

En vista de que la función v asociada a $\rho(\mathcal{A})$ toma valores en $\{0,1\}$, para que el triple $(A,O,\mathcal{R})$ asociado a $\rho(\mathcal{A})$ sea una instancia positiva del PROBLEMA 2, es necesario y basta que exista asignación PELE para $(A,O,\mathcal{R})$.

Recordemos que nuestro objetivo es probar que $\mathcal{A}$ es una instancia positiva de 3DMx si y solo si $\rho(\mathcal{A})$ es una instancia positiva del PROBLEMA 2. Por lo expuesto en el anterior párrafo, $\rho(\mathcal{A})$ es una instancia positiva del PROBLEMA 2 si y solo si existe asignación PELE para su triple asociado $(A,O,\mathcal{R})$. Pero, en vista de que las preferencias son monótonas (ver la sección 3.2), por la proposición 3.1, existe asignación PELE para $(A,O,\mathcal{R})$ si y solo si existe asignación PELE para el problema restringido que no considera los agentes en I. Notar que $|I| < n^6 - 2$. Haciendo $\widehat{A} = A \setminus I$, podemos notar al problema restringido con $\left(\widehat{A}, O, v|_{\widehat{A}\times O}\right)$; nos concentraremos en este problema, es decir, vamos a probar que $\mathcal{A}$ es una instancia positiva de 3DMx si y solo si existe asignación PELE para $\left(\widehat{A}, O, v|_{\widehat{A}\times O}\right)$. La consecuencia será que $\mathcal{A}$ es una instancia positiva de 3DMx si y solo si $\rho(\mathcal{A})$ es una instancia positiva del

PROBLEMA 2.

Lema 4.12. *Sean $\mathcal{A}$ una instancia positiva de 3DMx, $(A, O, \mathcal{R})$ el triple asociado a $\rho(\mathcal{A})$, y $\left(\widehat{A}, O, v|_{\widehat{A}\times O}\right)$ el triple restringido que no considera los agentes indiferentes en A. Entonces existe asignación Pareto-eficiente π' para $\left(\widehat{A}, O, v|_{\widehat{A}\times O}\right)$.*

Demostración. Por hipótesis, existe $M' \subseteq D^{\mathcal{A}}$ tal que, para todo $i \in |\mathcal{A}|$, existen triples

$$(i, a_1, b_1), (a_2, i, b_2), (a_3, b_3, i) \in M',$$

y tal que para todo par de triples diferentes

$$(i_1, j_1, k_1), (i_2, j_2, k_2) \in M' \implies i_1 \neq i_2, j_1 \neq j_2 \text{ y } k_1 \neq k_2.$$

Ahora, sea π' la asignación para $\left(\widehat{A}, O, v|_{\widehat{A}\times O}\right)$ tomada de la siguiente manera. La notación $c \to a$ indica que el bien c es asignado al agente a.

$$\begin{aligned}(1,1,0,i,j,k) &\to (0,0,1,i,j,k),\\ (1,0,1,i,j,k) &\to (0,1,0,i,j,k),\\ (0,1,1,i,j,k) &\to (1,0,0,i,j,k),\end{aligned}$$

para todo $(i, j, k) \in D^{\mathcal{A}}$, y

$$\begin{aligned}(0,0,1,0,0,k) &\to (0,0,1,i,j,k),\\ (0,1,0,0,j,0) &\to (0,1,0,i,j,k),\\ (1,0,0,i,0,0) &\to (1,0,0,i,j,k),\end{aligned}$$

si $(i, j, k) \in M'$.

Vamos a probar que π' es una asignación Pareto-eficiente.

Afirmamos que π' es asignación. Sean $a, b \in \widehat{A}$ agentes distintos. Demostraremos que $\pi'(a) \cap \pi'(b) = \emptyset$.

Tenemos dos casos: en el primero, a y b están asociados al mismo $(i, j, k) \in D^{\mathcal{A}}$; en el segundo, a está asociado a $(i_1, j_1, k_1) \in D^{\mathcal{A}}$ y b está asociado a $(i_2, j_2, k_2) \in D^{\mathcal{A}}$, pero

$$(i_1, j_1, k_1) \neq (i_2, j_2, k_2).$$

Tratemos el primer caso. Los agentes a y b están entre

$$(0,0,1,i,j,k), \quad (0,1,0,i,j,k), \quad (1,0,0,i,j,k).$$

Por la manera en la que se ha tomado π', a y b, o bien reciben paquetes unitarios, o bien reciben paquetes con exactamente dos bienes (si $(i,j,k) \in M'$), pero en ambos casos, los paquetes recibidos por los agentes son disjuntos.

En cuanto al segundo caso, los bienes asignados a a están entre los siguientes:

$$(1,1,0,i_1,j_1,k_1), \quad (1,0,1,i_1,j_1,k_1), \quad (0,1,1,i_1,j_1,k_1),$$
$$(0,0,1,0,0,k_1), \quad (0,1,0,0,j_1,0), \quad (1,0,0,i_1,0,0).$$

Y los objetos asignados a b están entre

$$(1,1,0,i_2,j_2,k_2), \quad (1,0,1,i_2,j_2,k_2), \quad (0,1,1,i_2,j_2,k_2),$$
$$(0,0,1,0,0,k_2), \quad (0,1,0,0,j_2,0), \quad (1,0,0,i_2,0,0).$$

Supongamos, por contradicción, que a y b comparten al menos un bien. Notemos, entonces, que estos agentes deben compartir exactamente un bien de entre

$$(0,0,1,0,0,k_1), \quad (0,1,0,0,j_1,0), \quad (1,0,0,i_1,0,0).$$

Entonces (i_1,j_1,k_1) y (i_2,j_2,k_2) son miembros de M' por la manera en la que se tomó π'. Es más, $k_1 = k_2$, $j_1 = j_2$ o $i_1 = i_2$, pero esto contradice el que hecho de que dos elementos distintos de M' tienen todas sus componentes diferentes.

Vemos así que en todos los casos a y b no comparten bienes. Luego π' es en efecto una asignación.

Probemos ahora que π' es Pareto-eficiente. Ya que la función $v|_{\widehat{A}\times O}$ toma valores en $\{0,1\}$, por la proposición 3.6, basta probar que todo bien deseado es asignado a un agente que lo desea. Notemos que para todo $(i,j,k) \in D^{\mathcal{A}}$, los bienes

$$(1,1,0,i,j,k), \quad (1,0,1,i,j,k), \quad (0,1,1,i,j,k),$$

son asignados a agentes que los desean.

Resta probar que, para todo $i \in |\mathcal{A}|$, los bienes

$$(0,0,1,0,0,i), \quad (0,1,0,0,i,0), \quad (1,0,0,i,0,0),$$

son asignados a agentes que los desean.

Sea $i \in |\mathcal{A}|$. En vista de que $\mathcal{A}$ es una instancia positiva de 3DMx, existen triples

$$(i,a_1,b_1), (a_2,i,b_2), (a_3,b_3,i) \in M'.$$

Entonces,

$$(0,0,1,0,0,i) \to (0,0,1,a_3,b_3,i),$$
$$(0,1,0,0,i,0) \to (0,1,0,a_2,i,b_2),$$
$$(1,0,0,i,0,0) \to (1,0,0,i,a_1,b_1).$$

Así, los tres bienes son asignados a agentes que los desean. Como conclusión, π' es Pareto-eficiente. □

Lema 4.13. *Sean $\mathcal{A}$ una instancia positiva de 3DMx, $(A,O,\mathcal{R})$ el triple asociado a $\rho(\mathcal{A})$, y $\left(\widehat{A},O,v|_{\widehat{A}\times O}\right)$ el triple restringido que no considera los agentes indiferentes en A. Entonces existe asignación PELE π' para $\left(\widehat{A},O,v|_{\widehat{A}\times O}\right)$.*

Demostración. Sea π' como en la demostración del lema 4.12. Luego, π' es una asignación Pareto-eficiente para $\left(\widehat{A},O,v|_{\widehat{A}\times O}\right)$.

Ahora demostraremos que π' está libre de envidia. Sean a,b dos agentes en $\widehat{A}$. Vamos a probar que

$$u_a(\pi'(b)) \leq u_a(\pi'(a)).$$

Tenemos dos casos: a y b están asociados al mismo $(i,j,k) \in D^{\mathcal{A}}$, o a y b no están asociados al mismo $(i,j,k) \in D^{\mathcal{A}}$.

En el primer caso a y b son dos agentes de entre

$$(0,0,1,i,j,k), \quad (0,1,0,i,j,k), \quad (1,0,0,i,j,k).$$

Recordemos que estos tres agentes desean los mismos seis bienes. Así, si $(i,j,k) \in M'$,

$$u_a(\pi'(b)) = 2 = u_a(\pi'(a)).$$

Si $(i,j,k) \notin M'$,

$$u_a(\pi'(b)) = 1 = u_a(\pi'(a)).$$

En el segundo caso, a es uno de

$$(0,0,1,i_1,j_1,k_1), \quad (0,1,0,i_1,j_1,k_1), \quad (1,0,0,i_1,j_1,k_1),$$

para algún $(i_1,j_1,k_1) \in D^{\mathcal{A}}$, y b es alguno de

$$(0,0,1,i_2,j_2,k_2), \quad (0,1,0,i_2,j_2,k_2), \quad (1,0,0,i_2,j_2,k_2)$$

para algún $(i_2,j_2,k_2) \in D^{\mathcal{A}}$ distinto de (i_1,j_1,k_1). Aquí tenemos dos subcasos: a b se

le asigna al menos un bien deseado por a, o a b no se le asigna bien alguno deseado por a.

En el primer subcaso, recordemos que los bienes asignados a b están entre

$$\begin{gathered}(1,1,0,i_2,j_2,k_2), \quad (1,0,1,i_2,j_2,k_2), \quad (0,1,1,i_2,j_2,k_2),\\ (0,0,1,0,0,k_2), \quad (0,1,0,0,j_2,0), \quad (1,0,0,i_2,0,0),\end{gathered}$$

con a lo mucho uno de la segunda fila y exactamente uno de la primera. Mientras los bienes deseados por a son

$$\begin{gathered}(1,1,0,i_1,j_1,k_1), \quad (1,0,1,i_1,j_1,k_1), \quad (0,1,1,i_1,j_1,k_1),\\ (0,0,1,0,0,k_1), \quad (0,1,0,0,j_1,0), \quad (1,0,0,i_1,0,0).\end{gathered}$$

Además, a a se le asigna exactamente un bien de la primera fila y a lo mucho uno de la segunda. Sigue que, en este subcaso,

$$u_a(\pi'(b)) = 1 \leq u_a(\pi'(a)).$$

En el segundo subcaso, es claro que

$$u_a(\pi'(b)) = 0 < u_a(\pi'(a)).$$

Así, en todos los casos,

$$u_a(\pi'(b)) \leq u_a(\pi'(a)).$$

Ya que $a, b \in \widehat{A}$ fueron tomados arbitrarios, se concluye que π' está libre de envidia. □

Lema 4.14. *Sean $\mathcal{A}$ una ω-estructura, $(A, O, \mathcal{R})$ el triple asociado a $\rho(\mathcal{A})$, y $\left(\widehat{A}, O, v|_{\widehat{A}\times O}\right)$ el triple restringido que no considera los agentes indiferentes en A. Supongamos existe asignación π' PELE para $\left(\widehat{A}, O, v|_{\widehat{A}\times O}\right)$. Entonces, para todo $(i,j,k) \in D^{\mathcal{A}}$, el grupo de agentes*

$$\{(0,0,1,i,j,k), (0,1,0,i,j,k), (1,0,0,i,j,k)\}$$

recibe un paquete que incluye los seis bienes deseados por los tres:

$$\begin{gathered}(1,1,0,i,j,k), \quad (1,0,1,i,j,k), \quad (0,1,1,i,j,k),\\ (0,0,1,0,0,k), \quad (0,1,0,0,j,0), \quad (1,0,0,i,0,0);\end{gathered}$$

o recibe un paquete que incluye solo los siguientes tres bienes de entre los deseados:

$$(1,1,0,i,j,k), \quad (1,0,1,i,j,k), \quad (0,1,1,i,j,k).$$

Demostración. Sea $(i,j,k) \in D^{\mathcal{A}}$. Ya que π' es Pareto-eficiente y la función $v|_{\widehat{A}\times O}$ tiene imagen en $\{0,1\}$, por la proposición 3.6, los seis bienes

$$\begin{gathered}(1,1,0,i,j,k), \quad (1,0,1,i,j,k), \quad (0,1,1,i,j,k),\\ (0,0,1,0,0,k), \quad (0,1,0,0,j,0), \quad (1,0,0,i,0,0),\end{gathered}$$

son asignados a agentes que los desean, pero los únicos agentes que desean los bienes en la primera fila son

$$(0,0,1,i,j,k), \quad (0,1,0,i,j,k), \quad (1,0,0,i,j,k).$$

En vista de que π' está libre de envidia, si a y b están entre estos agentes,

$$u_b(\pi'(b)) = u_a(\pi'(b)) \leq u_a(\pi'(a)),$$

pues a y b tienen las mismas preferencias. De esta manera,

$$u_b(\pi'(b)) = u_a(\pi'(a)).$$

Si notamos a esta cantidad con d, vemos que a cada agente de los tres considerados se le asigna exactamente d objetos deseados. Luego, el grupo de esos agentes es asignado $3d$ objetos de entre los deseados. Notar que $d \geq 1$. Si $d = 1$, los objetos deseados asignados deben ser

$$(1,1,0,i,j,k), \quad (1,0,1,i,j,k), \quad (0,1,1,i,j,k).$$

Si $d = 2$, todos los objetos deseados deben asignados al grupo. Finalmente, d no puede ser mayor que 2. El resultado sigue. □

Lema 4.15. *Sean $\mathcal{A}$ una instancia de 3DMx, $(A,O,\mathcal{R})$ el triple asociado a $\rho(\mathcal{A})$, y $\left(\widehat{A},O,v|_{\widehat{A}\times O}\right)$ el triple restringido que no considera los agentes indiferentes en A. Si existe asignación PELE π' para $\left(\widehat{A},O,v|_{\widehat{A}\times O}\right)$, entonces $\mathcal{A}$ una instancia positiva de 3DMx.*

Demostración. Supongamos que existe asignación PELE π' para $\left(\widehat{A},O,v|_{\widehat{A}\times O}\right)$. Vamos a probar que $\mathcal{A}$ es una instancia positiva de 3DMx.

Considerando el lema 4.14, tomemos M' con los triples $(i,j,k) \in D^{\mathcal{A}}$ tales que los

objetos

$$(0,0,1,0,0,k), \quad (0,1,0,0,j,0), \quad (1,0,0,i,0,0),$$

son asignados al grupo

$$\{(0,0,1,i,j,k),(0,1,0,i,j,k),(1,0,0,i,j,k)\}.$$

Vamos a probar que M' es tal que, para todo $i \in |\mathcal{A}|$, existen triples

$$(i,a_1,b_1),(a_2,i,b_2),(a_3,b_3,i) \in M',$$

y tal que para todo par de triples diferentes

$$(i_1,j_1,k_1),(i_2,j_2,k_2) \in M' \implies i_1 \neq i_2, j_1 \neq j_2 \text{ y } k_1 \neq k_2.$$

Sea $i \in |\mathcal{A}|$. Sabemos que los objetos

$$(0,0,1,0,0,i), \quad (0,1,0,0,i,0), \quad (1,0,0,i,0,0),$$

son bienes deseados. En vista de que π' es Pareto-eficiente y la función $v|_{\widehat{A}\times O}$ tiene imagen en $\{0,1\}$, por la proposición 3.6, estos tres bienes son asignados a agentes que los desean. Luego, existen triples

$$(i,a_1,b_1),(a_2,i,b_2),(a_3,b_3,i) \in D^{\mathcal{A}}$$

tales que

$$\begin{aligned}
(1,0,0,i,0,0) &\to \{(0,0,1,i,a_1,b_1),(0,1,0,i,a_1,b_1),(1,0,0,i,a_1,b_1)\},\\
(0,1,0,0,i,0) &\to \{(0,0,1,a_2,i,b_2),(0,1,0,a_2,i,b_2),(1,0,0,a_2,i,b_2)\},\\
(0,0,1,0,0,i) &\to \{(0,0,1,a_3,b_3,i),(0,1,0,a_3,b_3,i),(1,0,0,a_3,b_3,i)\}.
\end{aligned}$$

La notación $c \to G$ indica que el bien c es otorgado a alguno de los agentes en G.

Así, por la manera en que se tomó M',

$$(i,a_1,b_1),(a_2,i,b_2),(a_3,b_3,i) \in M'.$$

Por ejemplo, el bien $(0,0,1,0,0,i)$ es asignado al grupo

$$\{(0,0,1,a_3,b_3,i),(0,1,0,a_3,b_3,i),(1,0,0,a_3,b_3,i)\}.$$

Por el lema 4.14, los bienes deseados por estos agentes que son asignados al grupo,

son todos los seis posibles, en particular,

$$(0,0,1,0,0,i), \quad (0,1,0,0,b_3,0), \quad (1,0,0,a_3,0,0)$$

Luego, por la manera en la que se tomó M', $(a_3, b_3, i) \in M'$.

Resta probar que para todo par de triples diferentes

$$(i_1,j_1,k_1),(i_2,j_2,k_2) \in M' \implies i_1 \neq i_2, j_1 \neq j_2 \text{ y } k_1 \neq k_2.$$

Sean

$$(i_1,j_1,k_1),(i_2,j_2,k_2) \in M'$$

triples diferentes y supongamos, por contradicción, que coinciden en alguna coordenada. Sabemos que los objetos

$$(0,0,1,0,0,k_1), \quad (0,1,0,0,j_1,0), \quad (1,0,0,i_1,0,0),$$

son asignados al grupo

$$\{(0,0,1,i_1,j_1,k_1),(0,1,0,i_1,j_1,k_1),(1,0,0,i_1,j_1,k_1)\};$$

y que los objetos

$$(0,0,1,0,0,k_2), \quad (0,1,0,0,j_2,0), \quad (1,0,0,i_2,0,0),$$

son asignados al grupo

$$\{(0,0,1,i_2,j_2,k_2),(0,1,0,i_2,j_2,k_2),(1,0,0,i_2,j_2,k_2)\}.$$

De esta manera, alguno de los bienes considerados es compartido por agentes distintos, lo que contradice el hecho de π' es asignación.

Concluimos que $\mathcal{A}$ es una instancia positiva de 3DMx. □

El siguiente teorema es consecuencia de que 3DMx es un problema NP-completo vía fops, y de que ρ es una fop que reduce 3DMx al PROBLEMA 2, esto gracias a los lemas 4.13 y 4.15.

Teorema 4.3. *El PROBLEMA 2 es NP-hard vía fops.*

El siguiente corolario es consecuencia de los teoremas 4.2 y 4.3.

Corolario 4.1. *El PROBLEMA 2 es NP-completo vía fops.*

4.3. Problema 3: preferencias aditivas no restringidas

En este problema, $v(i, x_j) \in \mathbb{Z}_{\geq 0}$ para todo agente i y para todo bien x_j. Cuando $v(i, x_j) \in \mathbb{Z}$, la proposición 20 en [4] implica que el problema respectivo es NP-hard vía reducciones polinomiales; además este problema está en Σ_2^p. Aquí demostraremos, usando técnicas sintácticas, que este problema está en Σ_2^p y, usando el corolario 2.2, probaremos que es NP-hard vía fops.

El vocabulario usado para este problema será el mismo que aquel del PROBLEMA 2. Además, la interpretación del vocabulario será la misma.

PROBLEMA 3:

Instancia: triple $(A, O, \mathcal{R})$ con

- $A = \{0, 1, \ldots, n-1\}$,
- $O = \{x_0, \ldots, x_{n-1}\}$ y
- $\mathcal{R}$ el perfil de preferencias generado a partir de v.

Pregunta: ¿Existe asignación PELE para $(A, O, \mathcal{R})$?

Notemos que las instancias del PROBLEMA 3 tienen un número de agentes igual al número de bienes. Esto sucede porque las instancias del PROBLEMA 3 son generadas a partir de σ-estructuras. Así, también es cierto que si una σ-estructura tiene tamaño n, el máximo valor que puede declarar cualquier agente por cualquier bien es $2^n - 1$ en decimal. ¿Podemos, por ejemplo, para cada triple $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$ con número de agentes distinto del número de bienes, crear una σ-estructura con triple asociado $(A, O, \mathcal{R})$, que sea una instancia positiva del PROBLEMA 3 si y solo si existe asignación PELE para $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$? La respuesta es afirmativa. La demostración de la proposición B.2 ayuda en esta y otras situaciones.

Ahora, en el contexto de preferencias aditivas, si π y π' son asignaciones, π' no Pareto-domina π si y solo si existe un agente i tal que $u_i(\pi'(i)) < u_i(\pi(i))$ o, para todo agente k, $u_k(\pi'(k)) = u_k(\pi(k))$. La fórmula en el siguiente lema ayuda a expresar esta propiedad.

Lema 4.16. *Sean $\mathcal{B}$ una σ-estructura, y P, Q, S, T relaciones tales que*

$$\mathcal{B} \models \Phi_{asignación}(P),$$
$$\mathcal{B} \models \Phi_{asignación}(Q),$$
$$\mathcal{B} \models \Phi_{sum-asociada}(S, P) \text{ y}$$
$$\mathcal{B} \models \Phi_{sum-asociada}(T, Q).$$

Definamos, para todo $s \in |\mathcal{B}|$,

$$\pi(s) = \{x_t \in O |\, (s,t) \in P\}\ y$$
$$\pi'(s) = \{x_t \in O |\, (s,t) \in Q\}.$$

Consideremos la fórmula

$$\Phi_{noPdomina}(\mathbf{T^5}, \mathbf{S^5}) = ((\exists x)\Phi_{compara_{menor}}(\mathbf{T^5}, \mathbf{S^5}, x, x)) \vee ((\forall x)\Phi_{compara_{=}}(\mathbf{T^5}, \mathbf{S^5}, x, x)).$$

Luego,

$$\mathcal{B} \models \Phi_{noPdomina}(T, S) \tag{4.25}$$

si y solo si π' no Pareto-domina π.

Demostración. Primero, notemos que 4.25 es cierta si y solo si existe $i \in |\mathcal{B}|$ tal que

$$\mathcal{B} \models \Phi_{compara_{menor}}(T, S, i, i),$$

o, para todo $k \in |\mathcal{B}|$,

$$\mathcal{B} \models \Phi_{compara_{=}}(T, S, k, k).$$

Sea $(A, O, \mathcal{R})$ el triple asociado a $\mathcal{B}$. Así, por el lema 4.10, 4.25 equivale a afirmar que existe $i \in A$ tal que

$$u_i(\pi'(i)) < u_i(\pi(i)),$$

o, para todo $k \in A$,

$$u_k(\pi'(k)) = u_k(\pi(k));$$

es decir, equivale a aseverar que π' no Pareto-domina π. □

Teorema 4.4. *El PROBLEMA 3 está en Σ_2^p.*

Demostración. Una sentencia que define el problema es

$$\begin{aligned}\Phi_2 = &(\exists \mathbf{P^2})(\exists \mathbf{S^5})(\forall \mathbf{Q^2})(\forall \mathbf{T^5}) \\ &(\Phi_{\text{asignación}}(\mathbf{P^2}) \wedge \Phi_{sum-asociada}(\mathbf{S^5}, \mathbf{P^2}) \wedge \Phi_{LE}(\mathbf{S^5}) \\ &\wedge ((\Phi_{\text{asignación}}(\mathbf{Q^2}) \wedge \Phi_{sum-asociada}(\mathbf{T^5}, \mathbf{Q^2})) \rightarrow \\ &\Phi_{noPdomina}(\mathbf{T^5}, \mathbf{S^5}))).\end{aligned}$$

En efecto, supongamos que $\mathcal{B}$ es una σ-estructura con $(A, O, \mathcal{R})$ su triple asociado. Vamos a probar que $\mathcal{B} \models \Phi_2$ si y solo si existe asignación PELE π para $(A, O, \mathcal{R})$.

Asumamos $\mathcal{B} \models \Phi_2$. Sigue que existen relaciones P, S sobre $|\mathcal{B}|$ tales que, para cualesquier relaciones Q, T sobre el mismo conjunto,

$$\mathcal{B} \models \Phi_{\text{asignación}}(P), \tag{4.26}$$

$$\mathcal{B} \models \Phi_{sum-asociada}(S, P), \tag{4.27}$$

$$\mathcal{B} \models \Phi_{LE}(S), \tag{4.28}$$

y, si

$$\mathcal{B} \models \Phi_{\text{asignación}}(Q) \text{ y} \tag{4.29}$$

$$\mathcal{B} \models \Phi_{sum-asociada}(T, Q), \tag{4.30}$$

entonces

$$\mathcal{B} \models \Phi_{noPdomina}(T, S). \tag{4.31}$$

Definamos $\pi(s) = \{x_t \in O | (s,t) \in P\}$ para todo $s \in |\mathcal{B}|$. Vamos a probar que π es asignación PELE para $(A, O, \mathcal{R})$.

4.26 y el lema 4.3 garantizan que π es asignación. Además, las condiciones 4.26, 4.27 y 4.28, y el lema 4.11, implican que π está libre de envidia. Luego, basta probar que π es Pareto-eficiente.

Sea π' una asignación para $(A, O, \mathcal{R})$. Demostraremos que π' no Pareto-domina π. Definamos $Q = \{(i,j) \in |\mathcal{B}|^2 | \, x_j \in \pi'(i)\}$. De esta manera, $\pi'(i) = \{x_j \in O | \, (i,j) \in Q\}$ para todo $i \in |\mathcal{B}|$. Por otro lado, notemos que podemos construir una relación T tal que, para todo $h, i, j, k_1, k_2 \in |\mathcal{B}|$, $(h, i, j, k_1, k_2) \in T$ si y solo si, en el orden lexicográfico de pares de números, el (k_1, k_2)-ésimo bit en la expansión binaria de la cantidad

$$\sum_{s=0}^{j} v(i, x_s) \cdot \mathbb{1}_{\pi'(h)}(x_s)$$

es un 1. Luego, por el lema 4.3, tenemos 4.29. Además, por la manera en la que se ha tomado T, y gracias al lema 4.9, se tiene 4.30. Así, se infiere 4.31 y, a través del lema 4.16, que π' no Pareto-domina π. Concluimos que existe asignación PELE, en este caso π, para $(A, O, \mathcal{R})$.

Recíprocamente, supongamos existe asignación PELE π para $(A, O, \mathcal{R})$. Definamos $P = \{(i,j) \in |\mathcal{B}|^2 | \, x_j \in \pi(i)\}$. Así, $\pi(i) = \{x_j \in O | \, (i,j) \in P\}$ para todo $i \in |\mathcal{B}|$. Por otra parte, podemos construir una relación S tal que, para todo $h, i, j, k_1, k_2 \in |\mathcal{B}|$, $(h, i, j, k_1, k_2) \in S$ si y solo si, en el orden lexicográfico de pares de números, el

(k_1, k_2)-ésimo bit en la expansión binaria de la cantidad

$$\sum_{s=0}^{j} v(i, x_s) \cdot \mathbb{1}_{\pi(h)}(x_s)$$

es un 1. Gracias al lema 4.3, la condición 4.26 es verdadera. Por la manera en la que se ha tomado S y por el lema 4.9 se tiene 4.27. Ahora, el lema 4.11 y el hecho de que π está libre de envidia implican 4.28.

Ahora, sean Q, T relaciones sobre $|\mathcal{B}|$ que verifican 4.29 y 4.30. Definamos $\pi'(s) = \{x_t \in O | (s,t) \in Q\}$ para todo $s \in |\mathcal{B}|$. Por el lema 4.3, π' es asignación para $(A, O, \mathcal{R})$. Entonces π' no Pareto-domina π. En vista de que 4.26 y 4.27 son verdaderas, el lema 4.16 nos permite concluir que 4.31 es cierta. Lo expuesto implica que $\mathcal{B} \models \Phi_2$.

Concluimos que $\mathcal{B} \models \Phi_2$ si y solo si $(A, O, \mathcal{R})$ es una instancia positiva del PROBLEMA 3.

Ahora, Φ_2 es equivalente a una sentencia $\Phi_2' \in \text{SO}\exists\forall$, (ver la prueba del teorema 3.5.11 en [5]: solo necesitamos mover los cuantificadores de primer orden al principio, luego de $(\exists \mathbf{P^2})(\exists \mathbf{S^5})(\forall \mathbf{Q^2})(\forall \mathbf{T^5})$). Así, por el teorema 2.2, el PROBLEMA 3 está en Σ_2^p. □

Observación 4.1. Notemos que las expresiones asociadas a la descripción de una asignación Pareto-eficiente, como $u_i(\pi'(i)) < u_i(\pi(i))$, involucran solo un agente, no dos. Así, en la sentencia planteada en la demostración del teorema 4.4, se pudo haber usado una variable $\mathbf{T^4}$ en vez de la variable $\mathbf{T^5}$, es decir, se pudo haber hecho uso de una variable de relación de aridad 4 en vez de una de aridad 5; desde luego, eso habría implicado el desarrollo de otras fórmulas lógicas, análogas a las que ya se ha usado. Hacerlo no aportaría de manera significativa y más bien extendería el número de páginas del trabajo.

Teorema 4.5. *El PROBLEMA 3 es NP-hard vía fops.*

Demostración. Aquí haremos uso de la definición de superfluidad (ver definición 2.29). Tomemos C = NP, $\mathcal{L}^* = \text{SO}\exists$, $\mathcal{L} = \text{SO}\exists\forall$, y $\mathcal{L}' = \text{FO}\forall$. Consideremos la sentencia

$$\Phi_2 \wedge \gamma_{0-1},$$

donde Φ_2 es la sentencia planteada en la demostración del teorema 4.4, y γ_{0-1} es la sentencia en el lema 4.2. Luego, para toda σ-estructura $\mathcal{B}$, $\mathcal{B} \models \Phi_2 \wedge \gamma_{0-1}$ si y

solo si el triple asociado a esta es una instancia positiva del PROBLEMA 2. Pero Φ_2 es equivalente a una sentencia $\Phi_2' \in \mathcal{L}$. Además, $\gamma_{0-1} \in \mathcal{L}'$ es equivalente a una sentencia $\Psi \in \mathcal{L}$ (ver la expresión 2.12). De esta manera, $\mathcal{B} \models \Phi_2' \wedge \Psi$ si y solo si el triple asociado a $\mathcal{B}$ es una instancia positiva del PROBLEMA 2.

Ahora, el PROBLEMA 2 es NP-hard vía fops. Luego, gracias al corolario 2.2, Φ_2' define un problema NP-hard en el mismo sentido, pero el problema que define esta sentencia es el PROBLEMA 3, de manera que el PROBLEMA 3 es NP-hard vía fops. □

Observación 4.2. La demostración del último teorema ilustra una aplicación del corolario 2.2. Sin embargo, en vez se pudo haber usado la fop ρ de la sección anterior, para reducir 3DMx al PROBLEMA 3. En efecto, $\mathcal{A}$ es una instancia positiva de 3DMx si y solo si $\rho(\mathcal{A})$ es una instancia positiva del PROBLEMA 2, es decir, si y solo si existe asignación PELE para el triple $(A, O, \mathcal{R})$ asociado a $\rho(\mathcal{A})$ y la función v tiene imagen en $\{0, 1\}$, pero todas las instancias $\rho(\mathcal{A})$ verifican esta última condición; es decir, cualquier sentencia $\Psi \in \mathcal{L}(\sigma)$ tal que $\text{MOD}(\Psi) = \text{MOD}(\gamma_{0-1})$ es superflua con respecto a ρ. De esta manera, $\mathcal{A}$ es una instancia positiva de 3DMx si y solo si existe asignación PELE para el triple $(A, O, \mathcal{R})$ asociado a $\rho(\mathcal{A})$, es decir, si y solo si $\rho(\mathcal{A})$ es una instancia positiva del PROBLEMA 3.

Así, el corolario 2.2 toma poder en caso de que la sentencia en FO$\forall$ no sea superflua.

Capítulo 5

Conclusiones y trabajo futuro

- Se ha analizado la complejidad computacional de tres problemas de distribución justa de recursos indivisibles desde el punto de vista de la teoría descriptiva, es decir, se consiguió el objetivo general del proyecto.
- A pesar de lo mencionado en el anterior punto, los problemas de decisión analizados no son los que se planteó en los objetivos específicos del plan del proyecto. En efecto, el primer problema que se planteó corresponde al de la proposición 9 del artículo [4], mientras que el primer problema analizado en este trabajo es una modificación del de la proposición 7. Se tomó este otro porque la demostración de la proposición 9 en el artículo tiene un error; concretamente, no hay garantía de que el problema esté en la clase NP. La proposición 7 trata el problema de la existencia de una asignación PELE cuando las preferencias de todos los agentes son monótonas e idénticas, capturadas por una fórmula proposicional φ.

  Ahora, la demostración de la proposición 7 en el artículo también tiene un error pues, bajo sus hipótesis, no es cierto que una asignación π es PELE si y solo si, para todo agente i, $\pi(i) \in Good_i$. Como contraejemplo, supongamos que la fórmula proposicional φ que describe las preferencias de todos los agentes no puede ser satisfecha. Luego, para todo i, $Good_i = \emptyset$. Por la proposición 3.3, todo agente es indiferente. La proposición 3.2 asegura que en este caso la asignación π que asigna $\emptyset$ a todo agente es PELE. Sin embargo, para todo agente i, $\pi(i) \notin Good_i$.

  Por otro lado, la proposición 3.5 asegura que la afirmación de los autores del artículo es cierta cuando ningún agente es indiferente (inclusive si las preferencias no son monótonas); justamente esta condición se incluye en el plan-

teamiento del PROBLEMA 1 tratado aquí. Además, los autores del artículo no imponen restricciones sobre la fórmula φ o el número de agentes, mientras en el PROBLEMA 1, la fórmula ψ asociada es generada por un conjunto de cláusulas que pueden contener solo variables proposicionales, y el número de agentes es dos. Cabe recalcar que nuestros resultados están basados en la demostración de la proposición 7 en el artículo.

- Los otros dos problemas de decisión planteados en los objetivos específicos del plan del proyecto, están en la clase de complejidad Δ_2^p. El autor de este trabajo estaba convencido de que tal clase era la intersección de las clases Σ_2^p y Π_2^p, pero eso no es cierto. Es más, durante el desarrollo de la investigación se buscó en la literatura alguna caracterización descriptiva de la clase Δ_2^p, pero no fue encontrada, de manera que se tuvo que buscar otros dos problemas para su análisis en el contexto descriptivo. En un principio, se pensó trabajar con los problemas del artículo [6], pero no se vio manera de conseguir los resultados en el mismo a través de fops (en el artículo se usa reducciones polinomiales). Así, se decidió trabajar con dos problemas de la sección 5.3 del artículo [4].
- Creemos que los conceptos de agente indiferente y de bien de valor nulo son bien conocidos entre quienes trabajan en el área de distribución justa de recursos, pero no hemos intentado verificarlo. Las proposiciones 3.1, 3.2, A.2, y A.3 están asociadas a esos conceptos. Suponemos, asimismo, que esos resultados son parte del folclore del área. El autor desarrolló estas definiciones en un contexto general y probó los resultados mencionados, luego de haber revisado literatura para desarrollar el proyecto.
- El aporte principal de este trabajo es la caracterización descriptiva de la complejidad computacional de tres problemas del área de distribución justa de recursos. Se probó que todos ellos son NP-hard vía fops, las cuales son un tipo muy débil de reducciones. Para el PROBLEMA 1 se propuso una sentencia existencial de segundo orden que lo define, mostrando que el problema está en la clase NP; además, se probó completitud en la clase planteando una sentencia que define el problema SET-SPLITTING (el cual es NP-completo) y haciendo uso de un operador sintáctico que sustenta completitud. Por otro lado, la caracterización de la complejidad del PROBLEMA 2 básicamente siguió la línea de la demostración de la proposición 21 de [4]; sin embargo, fue necesario probar que el problema 3DMx, una versión modificada de 3-DIMENSIONAL-MATCHING, es NP-completo vía fops (ver proposición B.3). Así, se probó que el PROBLEMA 2 es NP-completo vía fops. Finalmente, se demostró que el

PROBLEMA 3 está en Σ_2^p; para probar NP-hardness, se usó una modificación de la definición de superfluidad elaborada en [2]. Esta modificación fue una sugerencia del director del proyecto.

- Ahora, la comisión que aprobó el plan de este proyecto sugirió la implementación de algoritmos para los problemas analizados. Se hizo esto para los dos primeros problemas (ver el apéndice C). Concretamente, a partir de las sentencias se obtuvo los algoritmos. Aquí cabe mencionar que la sentencia usada para el PROBLEMA 2 puede ser modificada para obtener una menos compleja; de hecho, el algoritmo respectivo refleja las modificaciones necesarias. No se hizo esto en el capítulo 4, pues algunas fórmulas que se usó para el PROBLEMA 2 también fueron usadas para el PROBLEMA 3, y las modificaciones a las que se hace referencia no pueden ser transferidas al PROBLEMA 3. Además, tales modificaciones no afectan los resultados obtenidos.
- Hacemos notar que, en ocasiones, para caracterizar la complejidad computacional de un problema, es más compacto trabajar con fórmulas lógicas que con algoritmos. Considérese, por ejemplo, el PROBLEMA 1. La fórmula lógica es la sentencia en la demostración del teorema 4.1, mientras el algoritmo respectivo puede ser encontrado en el apéndice C.
- Como se ha mencionado, se modificó la definición de superfluidad encontrada en el artículo [2]. Se conocía que, si C es cierta clase de complejidad, ψ es una sentencia universal de primer orden y $\mathrm{MOD}(\Phi \wedge \psi)$ define un problema C-completo vía fops, entonces $\mathrm{MOD}(\Phi)$ define un problema C-completo. La modificación hecha en este trabajo (ver definición 2.29) y el corolario 2.2 nos permiten concluir que, si $\mathrm{MOD}(\Phi \wedge \psi)$ define un problema C-hard vía fops, entonces $\mathrm{MOD}(\Phi)$ define un problema C-hard en el mismo sentido.
- En el contexto de la lógica proposicional, una cláusula básicamente define una disyunción, y una colección de cláusulas una fórmula en FNC. Las preferencias de los agentes del PROBLEMA 1 son expresadas con cláusulas de variables proposicionales. Ahora nos preguntamos qué sucede si esas cláusulas representan conjunciones, y la colección de cláusulas una fórmula en FND. Esta es de hecho una forma más natural de representar las preferencias de los agentes. Como se mencionó en la sección 3.1, si, por ejemplo, $O = \{x_0, \ldots, x_{10}\}$ y $Good_i$ está conformado por todos los superconjuntos de $\{x_1\}$ y los de $\{x_2, x_4\}$, podemos representar estas preferencias de manera compacta con la fórmula $\varphi_i = x_1 \vee (x_2 \wedge x_4)$. Se ha esbozado una demostración que sugiere que, en es-

te caso, el PROBLEMA 1 estaría en una clase complejidad que es subconjunto propio de P (de hecho subconjunto propio de L, la clase de todos los problemas que pueden ser decididos en espacio logarítmico). En un trabajo futuro se puede intentar determinar en qué clase complejidad está el problema de la existencia de una asignación PELE cuando las preferencias son binarias y son representadas de la manera que aquí se sugiere. El corolario 1 en [4] implica que un problema más general es Σ_2^p-completo vía reducciones polinomiales. Tal vez, el problema que sugerimos estudiar posee menor complejidad.

- Otra dirección en la que se podría trabajar es intentar caracterizar la complejidad computacional de más problemas de distribución justa de recursos, probando hardness o completitud en el sentido de proyecciones de primer orden.

Apéndice A

Distribución justa: proposiciones auxiliares

Proposición A.1. *Sea $\preceq$ un preorden total sobre un conjunto finito no vacío D. Entonces existe $b \in D$ tal que $c \preceq b$ para todo $c \in D$.*

Demostración. Usaremos la notación $d \prec e$ en caso de que $d \preceq e$ y $e \npreceq d$. Ahora, observemos que $d \prec e$ y $e \prec f$ implican $d \prec f$. En efecto, $d \prec e$ y $e \prec f$ implican, por transitividad, $d \preceq f$. Además, $f \npreceq d$ pues, en caso contrario, es decir, si $f \preceq d$, el hecho de que $d \preceq e$ implicaría, por transitividad, que $f \preceq e$, y esto contradice $e \prec f$.

Por otro lado, notemos que $d = e$ implica, por reflexividad, $d \preceq e$ y $e \preceq d$. Así, $d \prec e$ tiene como consecuencia $d \neq e$.

Ahora supongamos por contradicción que para todo $b \in D$, existe $c \in D$ con $c \npreceq b$. En vista de que $\preceq$ es un preorden total sobre D, para todo $b \in D$, existe $c \in D$ tal que $b \prec c$. Entonces D tiene un número infinito de elementos como consecuencia de que $d \prec e$ y $e \prec f$ implican $d \prec f$, y gracias a que $d \neq e$ si $d \prec e$. □

Lema A.1. *Sean $(A, O, \mathcal{R})$ un problema de distribución justa en el que las preferencias son monótonas, y $x_k \in O$ un bien de valor nulo. Entonces*

$$\begin{aligned} D \preceq_i E &\implies D \setminus \{x_k\} \preceq_i E \setminus \{x_k\} \quad y \\ D \prec_i E &\implies D \setminus \{x_k\} \prec_i E \setminus \{x_k\}. \end{aligned}$$

para cualesquier $D, E \in 2^O$ e $i \in A$.

Demostración. Sean $i \in A$ un agente cualquiera, y $D, E \in 2^O$ con $D \preceq_i E$. En vista de que las preferencias son monótonas, $D \setminus \{x_k\} \preceq_i D$. Gracias a la transitividad de $\preceq_i$,

tenemos $D \setminus \{x_k\} \preceq_i E$. Además, $E \preceq_i E \cup \{x_k\}$ por monotonía. Y en vista de que x_k es un bien de valor nulo,

$$E \cup \{x_k\} = (E \setminus \{x_k\}) \cup \{x_k\} \preceq_i E \setminus \{x_k\}. \tag{A.1}$$

Luego, por transitividad $D \setminus \{x_k\} \preceq_i E \setminus \{x_k\}$.

Si D, E son tales que $D \prec_i E$, entonces $D \preceq_i E$ y, por lo expuesto, $D \setminus \{x_k\} \preceq_i E \setminus \{x_k\}$. Probaremos $E \setminus \{x_k\} \not\preceq_i D \setminus \{x_k\}$ por contradicción. Asumamos $E \setminus \{x_k\} \preceq_i D \setminus \{x_k\}$. Por monotonía, $D \setminus \{x_k\} \preceq_i D$. Así, gracias a la transitividad de $\preceq_i$, $E \setminus \{x_k\} \preceq_i D$. Ahora, otra vez por monotonía, $E \preceq_i E \cup \{x_k\}$, y por A.1, $E \cup \{x_k\} \preceq_i E \setminus \{x_k\}$. Así, por transitividad, $E \preceq_i E \setminus \{x_k\}$. Una nueva aplicación de transitividad resulta en $E \preceq_i D$, lo que contradice $D \prec_i E$. Concluimos que $D \setminus \{x_k\} \prec_i E \setminus \{x_k\}$. □

Proposición A.2. *Dado un problema de distribución justa, $(A, O, \mathcal{R})$, en el que las preferencias son monótonas y $|O| = m$, si U es un conjunto de bienes de valor nulo con $|U| \leq m - 1$, entonces existe asignación PELE π para $(A, O, \mathcal{R})$ si y solo si existe asignación PELE π' para el problema restringido que no considera los bienes en U.*

Observación A.1. Es necesaria una aclaración. Las relaciones que definen las preferencias de los agentes en el caso restringido son las heredadas de aquellas en $\mathcal{R}$, es decir, son las relaciones

$$\lesssim_i = \{(C, B) \in \preceq_i \mid C, B \subseteq (O \setminus U)\}.$$

Estas necesariamente son reflexivas, transitivas, y completas. De esta manera, usaremos la notación $<_i$ para preferencia estricta.

Demostración. La condición $|U| \leq m - 1$ se impone para asegurar que el conjunto de bienes del problema restringido tenga al menos un miembro. Asumiremos que $|U| > 0$, pues en caso contrario no se necesita hacer nada.

Ahora, es suficiente probar el caso $|U| = 1$, pues una vez hecho esto se puede repetir el proceso hasta descartar todos los bienes en U. En cada iteración del proceso, la monotonía se mantiene y los bienes de valor nulo heredan esa condición. Asumiremos entonces que $U = \{x_k\}$ para algún k.

Lo que haremos en la demostración de $(\Rightarrow)$, básicamente, es tomar $\pi'(i) = \pi(i) \setminus \{x_k\}$. Para la prueba de $(\Leftarrow)$ tomaremos $\pi(i) = \pi'(i)$.

$(\Rightarrow)$ Supongamos existe asignación PELE π para $(A, O, \mathcal{R})$. Definamos $\pi' : A \to 2^{O \setminus U}$ con la fórmula $\pi'(i) = \pi(i) \setminus \{x_k\}$. Asumamos por contradicción que π' no

es Pareto-eficiente. Entonces existe asignación $\widehat{\pi}$ para el problema restringido que domina π'. Luego, $\pi'(i) \lesssim_i \widehat{\pi}(i)$ para todo i y existe un agente j con $\pi'(j) <_j \widehat{\pi}(j)$. Así, $\pi'(i) \preceq_i \widehat{\pi}(i)$ para todo i y $\pi'(j) \prec_j \widehat{\pi}(j)$. Definamos $\widetilde{\pi} : A \to 2^O$ asignación para el problema original por $\widetilde{\pi}(i) = \widehat{\pi}(i)$. Afirmamos que $\widetilde{\pi}$ Pareto-domina π.

Si existe, para el agente l tal que $\pi(l) = \pi'(l) \cup \{x_k\}$, se tiene $\pi(l) \preceq_l \pi'(l)$ pues x_k es de valor nulo. Para el resto de agentes i, tenemos $\pi(i) = \pi'(i)$, con la consecuencia $\pi(i) \preceq_i \pi'(i)$. De esta manera para todo agente i,

$$\pi(i) \preceq_i \pi'(i) \preceq_i \widehat{\pi}(i) = \widetilde{\pi}(i).$$

Además,

$$\pi(j) \preceq_j \pi'(j) \prec_j \widehat{\pi}(j) = \widetilde{\pi}(j),$$

con lo cual $\pi(j) \prec_j \widetilde{\pi}(j)$. Estos hechos contradicen la Pareto-eficiencia de π.

Ahora, π' está libre de envidia pues, por hipótesis, para cualesquier agentes i, l, $\pi(i) \preceq_l \pi(l)$. Luego, gracias al lema A.1,

$$\pi'(i) = \pi(i) \setminus \{x_k\} \preceq_l \pi(l) \setminus \{x_k\} = \pi'(l),$$

con la consecuencia $\pi'(i) \lesssim_l \pi'(l)$.

($\Leftarrow$) Sea π' una asignación PELE para el problema restringido. Tomemos $\pi : A \to 2^O$ asignación para el problema original con $\pi(i) = \pi'(i)$. Supongamos, por contradicción, que π no es Pareto-eficiente. Así, existe $\widetilde{\pi}$ que domina π. Tomemos la asignación $\widehat{\pi} : A \to 2^{O \setminus U}$ para el problema restringido definida por $\widehat{\pi}(i) = \widetilde{\pi}(i) \setminus \{x_k\}$. Afirmamos que $\widehat{\pi}$ Pareto-domina π'.

Sabemos que para todo agente l, $\pi'(l) = \pi(l) \preceq_l \widetilde{\pi}(l)$. Por el lema A.1 sigue que, para todo l,

$$\pi'(l) = \pi(l) \setminus \{x_k\} \preceq_l \widetilde{\pi}(l) \setminus \{x_k\} = \widehat{\pi}(l).$$

Además, existe j tal que $\pi'(j) = \pi(j) \prec_j \widetilde{\pi}(j)$. Tenemos entonces, gracias al lema A.1,

$$\pi'(j) = \pi(j) \setminus \{x_k\} \prec_j \widetilde{\pi}(j) \setminus \{x_k\} = \widehat{\pi}(j).$$

Así, $\pi'(l) \lesssim_l \widehat{\pi}(l)$ para todo l y $\pi'(j) <_j \widehat{\pi}(j)$. Luego existe contradicción, pues π' no sería Pareto-eficiente.

Finalmente, π está libre de envidia porque para cualesquier i, l,

$$\pi(l) = \pi'(l) \lesssim_i \pi'(i) = \pi(i),$$

con lo cual $\pi(l) \preceq_i \pi(i)$. □

Proposición A.3. *Sea $(A, O, \mathcal{R})$ un problema de distribución justa en el que las preferencias son binarias, no necesariamente monótonas, y en el que, para cada $i \in A$, las preferencias de i son capturadas por una fórmula φ_i. Supongamos que $2 \leq |O|$. Luego, una condición suficiente para que $x_k \in O$ tenga valor nulo es que x_k no aparezca en φ_i para i alguno.*

Demostración. Supongamos x_k satisface la condición. Sean i un agente y B un miembro de 2^O. Probaremos que $B \cup \{x_k\} \sim_i B$. Si φ_i no está en FNC, la transformamos en una fórmula lógicamente equivalente que lo esté y no contenga x_k; así, asumiremos que φ_i está en FNC. Aquí hay dos casos: $B \in Good_i$ o $B \notin Good_i$.

Trabajemos el primer caso. Tenemos $B \models \varphi_i$, es decir, toda disyunción c en φ_i contiene un literal l_c tal que $B \models l_c$. Si l_c es una variable proposicional x_j, se tiene $x_j \in B$. Luego, $x_j \in B \cup \{x_k\}$ y $B \cup \{x_k\} \models l_c$. Si $l_c = \neg x_j$, entonces $x_j \notin B$. Es más, $x_j \notin B \cup \{x_k\}$ pues x_k no aparece en φ_i. En este caso también $B \cup \{x_k\} \models l_c$. Se concluye que toda disyunción c en φ_i contiene un literal l_c tal que $B \cup \{x_k\} \models l_c$. Sigue que $B \cup \{x_k\} \models \varphi_i$ y $B \cup \{x_k\} \in Good_i$. De esta forma $B \preceq_i B \cup \{x_k\}$. Por otro lado, claramente tenemos $B \cup \{x_k\} \preceq_i B$, con lo cual $B \cup \{x_k\} \sim_i B$.

Ahora, asumamos $B \notin Good_i$. Así, existe una disyunción c en φ_i tal que ningún literal en ella es satisfecho por B. Sea l un literal en c. Si $l = x_j$, entonces $x_j \notin B$. Además, $x_j \notin B \cup \{x_k\}$. Luego $B \cup \{x_k\} \not\models l$. Si $l = \neg x_j$, $x_j \in B$ con las consecuencias $x_j \in B \cup \{x_k\}$ y $B \cup \{x_k\} \not\models l$. En todos los casos $B \cup \{x_k\} \not\models l$. Así, $B \cup \{x_k\} \not\models \varphi_i$, con lo cual $B \cup \{x_k\} \notin Good_i$ y $B \cup \{x_k\} \preceq_i B$. Por otro lado es claro que $B \preceq_i B \cup \{x_k\}$. $B \cup \{x_k\} \sim_i B$ sigue.

En vista de que $i \in A$ y $B \in 2^O$ fueron tomados arbitrarios, x_k tiene valor nulo. □

Apéndice B

Caracterización descriptiva de la complejidad computacional de algunos problemas de distribución justa: proposiciones auxiliares

B.1. Problema 1

Lema B.1. *Sean*

- $\widehat{A} = \{0, 1\}$,
- $\widehat{O} = \{x_0, \ldots, x_{m-1}\}$, *y*
- $\widehat{\mathcal{R}} = (\lesssim_0, \lesssim_1)$ *un perfil de preferencias binario generado a partir de un conjunto*

$$\widehat{Good} = \widehat{Good}_0 = \widehat{Good}_1 = \left\{ M \subseteq \widehat{O} |\, M \models \widehat{\psi} \right\},$$

con $\widehat{\psi}$ una fórmula proposicional positiva en FNC, con p disyunciones y con variables en $\widehat{O}$.

Luego, existe $\mathcal{B} \in STRUC[\nu]$ tal que $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$ es una restricción del triple $(A, O, \mathcal{R})$ asociado a $\mathcal{B}$, en el sentido de haber eliminado bienes.

Demostración. Vamos a construir una ν-estructura $\mathcal{B}$ con la propiedad deseada, es decir, vamos a tomar $\mathcal{B}$ tal que su triple asociado $(A, O, \mathcal{R})$ verifica: $A = \widehat{A}, \widehat{O} \subseteq O$, y el perfil de preferencias $\widehat{\mathcal{R}}$ es el heredado de $\mathcal{R}$ (ver la observación A.1). Recordemos que $\mathcal{R} = (\preceq_0, \preceq_1)$ es el perfil de preferencias generado por la fórmula ψ asociada a la estructura $\mathcal{B}$.

Tomemos

$$n = \text{máx}\{2, m, p\},$$

donde $n = \|\mathcal{B}\|$, $m = |\widehat{O}|$ y p es el número de disyunciones de $\widehat{\psi}$. Así, $A = \widehat{A}$ y

$$\widehat{O} \subseteq O = \{x_0, \ldots, x_{n-1}\}.$$

Además, ψ es generada por $n \geq p$ cláusulas. Ahora, por cada disyunción en $\widehat{\psi}$ construimos una cláusula con las mismas variables que la disyunción. Si $n > p$, tomamos las cláusulas restantes repitiendo alguna previa. Luego, todas las variables que ocurren en ψ están en $\widehat{O}$. Además, $\psi \equiv \widehat{\psi}$. A continuación definimos

$$Good_0 = Good_1 = Good = \{M \subseteq O |\, M \models \psi\}.$$

Ahora probaremos que $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$ constituye una restricción de $(A, O, \mathcal{R})$.

Si $\widehat{O} = O$, las instancias consideradas son las mismas, pues $\psi \equiv \widehat{\psi}$. Supongamos entonces que $\widehat{O} \subsetneq O$. Sea $i \in A$. Demostraremos que

$$\lesssim_i = \left\{(C, B) \in \preceq_i \mid C, B \subseteq \widehat{O}\right\},$$

Notemos que, por construcción, las variables presentes en $\widehat{\psi}$ y ψ están todas en $\widehat{O}$, es decir, ninguna es miembro de $O \setminus \widehat{O}$. Así, dado $M \subseteq \widehat{O}$, tratado como asignación de valores de verdad para los miembros de $\widehat{O}$, todas las variables en esas fórmulas son asignadas un valor de verdad bajo M. Entonces, si M es tratado como asignación de valores de verdad para los miembros de O, se está extendiendo la asignación en cuestión; concretamente, a las variables en $O \setminus \widehat{O}$ se les está asignando el valor false, pero esta asignación extendida no cambia el valor que toman ψ o $\widehat{\psi}$ bajo M visto como asignación de valores de verdad para los elementos de $\widehat{O}$ pues, como se ha mencionado, ninguna variable en $O \setminus \widehat{O}$ está presente en esas fórmulas.

Supongamos $C \lesssim_i B$. Por definición, C y B son subconjuntos de $\widehat{O}$ tales que $C \notin \widehat{Good}$ o $B \in \widehat{Good}$. Vamos a probar que $C \preceq_i B$. Naturalmente, C y B están contenidos en O. Ahora, si $C \notin \widehat{Good}$, en vista de que

$$\widehat{Good} = \left\{M \subseteq \widehat{O} |\, M \models \widehat{\psi}\right\},$$

$C \not\models \widehat{\psi}$. Por equivalencia lógica, $C \not\models \psi$. Así, cuando C es tratado como asignación de valores de verdad para los miembros de O, es tal que $C \not\models \psi$, es decir, $C \notin Good$. Por otro lado, si $B \in \widehat{Good}$, necesariamente $B \models \widehat{\psi}$. Luego, por equivalencia lógica, $B \models \psi$, y como consecuencia $B \in Good$. En ambos casos se tiene $C \preceq_i B$.

Recíprocamente, asumamos $C, B \subseteq \widehat{O}$ y $C \preceq_i B$. Luego $C \notin Good$ o $B \in Good$. Si $C \notin Good$, $C \not\models \psi$; aquí C es una asignación de valores de verdad para las variables en O. Por equivalencia lógica, $C \not\models \widehat{\psi}$. Entonces C, tratado como asignación de valores de verdad para las variables en $\widehat{O}$, es tal que $C \not\models \widehat{\psi}$, con lo cual $C \notin \widehat{Good}$. Por otra parte, si $B \in Good$, un razonamiento análogo permite deducir $B \in \widehat{Good}$. De esta manera, $C \lesssim_i B$, y la prueba concluye. □

Proposición B.1. *Sea $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$ como en el lema B.1, y tal que la condición $\widehat{Good} = \emptyset$ podría ser cierta. Entonces, existe $\mathcal{B} \in STRUC[\nu]$ tal que su triple asociado, $(A, O, \mathcal{R})$, es una instancia positiva del PROBLEMA 1 si y solo si $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$ no posee agentes indiferentes y existe asignación PELE para este triple.*

Demostración. En caso de que exista una asignación PELE para $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$ y sus agentes no sean indiferentes, diremos que ese triple es una instancia positiva del problema más general; en caso contrario, diremos que es una instancia negativa del problema más general.

Asumiremos que $\widehat{\mathcal{R}}$ es presentado, o bien como $\widehat{\psi}$, o bien como $\widehat{Good} = \emptyset$.

En el caso $\widehat{Good} = \emptyset$, por la proposición 3.3, los agentes en $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$ son indiferentes. Luego, podemos asociar a este triple una estructura $\mathcal{B}$ tal que todas las cláusulas correspondientes sean vacías, de manera que los agentes en $(A, O, \mathcal{R})$ sean indiferentes (ver proposición 4.1).

Si $\widehat{Good}$ es generado por $\widehat{\psi}$, tomamos $\mathcal{B}$ como en el lema B.1. Así, $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$ es una restricción de $(A, O, \mathcal{R})$. En este caso, $\widehat{Good} \neq \emptyset$, pues $\widehat{O} \models \widehat{\psi}$. Además, para todo $x_k \in O \setminus \widehat{O}$, x_k no aparece en $\widehat{\psi}$, y por la manera en la que se tomó, tampoco en ψ, de manera que x_k es un bien de valor nulo en la instancia $(A, O, \mathcal{R})$, esto gracias a la proposición A.3. Tomando $U = O \setminus \widehat{O}$, vemos que $|U| \leq n - 1$. Así, apelando a la proposición A.2, existe asignación PELE para $(A, O, \mathcal{R})$ si y solo si existe asignación PELE para $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$.

Ahora, si $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$ es una instancia negativa del problema más general, sus agentes son indiferentes o no existe asignación PELE para este. En el primer caso, por la proposición 3.3, $\widehat{Good} = 2^{\widehat{O}}$ o $\widehat{Good} = \emptyset$. Además, $\widehat{Good}$ es generado por $\widehat{\psi}$ o $\widehat{Good} = \emptyset$; en ambos casos $\widehat{Good} \neq 2^{\widehat{O}}$, pues no es posible que $\emptyset \models \widehat{\psi}$. Sigue que $\widehat{Good} = \emptyset$, con la consecuencia de que $\widehat{Good}$ no es generado por $\widehat{\psi}$. Por la manera en la que se toma $\mathcal{B}$ en este caso, los agentes en $(A, O, \mathcal{R})$ son indiferentes y este triple es una instancia negativa del PROBLEMA 1.

En caso de que no exista asignación PELE para $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$, por la proposición 3.2, no

es posible que los agentes sean indiferentes. Sigue que $\widehat{Good} \neq \emptyset$ y $\widehat{Good}$ es generado por $\widehat{\psi}$. Luego, ya que $\mathcal{B}$ se toma como en el lema B.1, no existe asignación PELE para $(A, O, \mathcal{R})$. De esta manera, este triple es una instancia negativa del PROBLEMA 1.

Ahora, si $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$ es una instancia positiva del problema más general, los agentes no son indiferentes y existe asignación PELE para este. En ese caso, por la proposición 3.3, $\emptyset \neq \widehat{Good} \neq 2^{\widehat{O}}$, y $\mathcal{B}$ se toma como en el lema B.1. Sigue que $\widehat{Good} \subseteq Good$. Además, $Good \neq 2^{O}$ (ver la demostración de la proposición 4.1). De esta manera, por la proposición 3.3, los agentes en $(A, O, \mathcal{R})$ no son indiferentes. Además, existe asignación PELE para $(A, O, \mathcal{R})$. Se concluye que $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$ es una instancia positiva del problema más general si y solo si $(A, O, \mathcal{R})$ es una instancia positiva del PROBLEMA 1. □

B.2. Problema 2

Proposición B.2. *Sea* $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$ *con*

- $\widehat{A} = \{0, 1, \ldots, N-1\}$,
- $\widehat{O} = \{x_0, \ldots, x_{m-1}\}$, *y*
- $\widehat{\mathcal{R}}$ *el perfil de preferencias generado por* $\widehat{v} : \widehat{A} \times \widehat{O} \to \mathbb{Z}_{\geq 0}$.

Entonces, existe una σ-estructura $\mathcal{B}$ *tal que su triple asociado,* (A, O, v)*, es una instancia positiva del PROBLEMA 2 si y solo si para todo agente* i *y para todo bien* x_j *en* $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$*,* $\widehat{v}(i, x_j) \in \{0, 1\}$*, y existe asignación PELE para este triple.*

Demostración. En caso de que para todo agente i y para todo bien x_j en $(\widehat{A}, \widehat{O}, \widehat{\mathcal{R}})$, $\widehat{v}(i, x_j) \in \{0, 1\}$, y exista asignación PELE para este triple, diremos que este es una instancia positiva del problema más general.

Primero, necesitamos tomar el número $\|\mathcal{B}\| = n$ de manera que sea posible expresar los valores en la imagen de $\widehat{v}$ a través de $V^{\mathcal{B}}$. Sea

$$M = \text{máx}\,\{\widehat{v}(i, o) \,|\, i \in \widehat{A}, o \in \widehat{O}\} \cup \{1\}.$$

Ya que $M \geq 1$, existe un entero $c \geq 1$ tal que

$$2^{c-1} \leq M < 2^{c}.$$

Así,

$$2^{c-1} \leq M \leq 2^{c} - 1.$$

Entonces bastan c bits para expresar M. Sea $C = \lceil \log_2 M \rceil + 1$. Luego $c \leq C \leq c + 1$.

Tomemos

$$n = \text{máx}\,\{|\widehat{A}|, |\widehat{O}|, C\}.$$

$\widehat{A} \subseteq A$ y $\widehat{O} \subseteq O$ siguen. Además, tomemos $V^{\mathcal{B}}$ de manera que

$$v(i,o) = \begin{cases} \widehat{v}(i,o) & \text{si } i \in \widehat{A} \text{ y } o \in \widehat{O}, \\ 0 & \text{si } i \notin \widehat{A} \text{ u } o \notin \widehat{O}. \end{cases}$$

Notar que $\widehat{v} = v|_{\widehat{A}\times\widehat{O}}$.

Ahora, el objetivo es probar que (A, O, v) es una instancia positiva del PROBLEMA 2 si y solo si $(\widehat{A}, \widehat{O}, \widehat{v})$ es una instancia positiva del problema más general. Notar que la imagen de v está en $\{0,1\}$ si y solo si la imagen de $\widehat{v}$ lo está. Probaremos entonces que existe π PELE para (A, O, v) si y solo si existe π' PELE para $(\widehat{A}, \widehat{O}, \widehat{v})$; lo haremos sin asumir que la imagen de $\widehat{v}$ está en $\{0,1\}$. La demostración consistirá de dos etapas. En cada una se restringirá el problema respectivo y se hará uso de proposiciones 3.1 y A.2.

Observemos primero que $(\widehat{A}, O, v|_{\widehat{A}\times O})$ es una restricción de (A, O, v) en el sentido de que podría tener menos agentes. Para todo $i \in \widehat{A}$, la relación $\preceq_i$ que define sus preferencias es la misma en ambos casos, pues $v(i,o) = v|_{\widehat{A}\times O}(i,o)$ para todo $o \in O$. Ya que la imagen de v está en $\mathbb{Z}_{\geq 0}$, las preferencias en ambos casos son monótonas. Además, para todo $i \in A \setminus \widehat{A}$ y para todo $x_j \in O$ se tiene $v(i, x_j) = 0$, de manera que i es indiferente (ver la sección 3.2). Definiendo $I = A \setminus \widehat{A}$, vemos que $|I| \leq n - 2$ y, gracias a la proposición 3.1, existe π PELE para (A, O, v) si y solo si existe $\widetilde{\pi}$ PELE para $(\widehat{A}, O, v|_{\widehat{A}\times O})$.

Consideremos ahora $(\widehat{A}, \widehat{O}, \widehat{v})$. Este triple es una restricción de $(\widehat{A}, O, v|_{\widehat{A}\times O})$, pues podría tener menos bienes. Además, las relaciones que definen las preferencias de los agentes en el caso restringido, son las heredadas de aquellas generadas por $v|_{\widehat{A}\times O}$, pues para todo $i \in \widehat{A}$ y para todo $x_j \in \widehat{O}$, $\widehat{v}(i, x_j) = v|_{\widehat{A}\times O}(i, x_j)$. También es cierto que para todo $x_k \in O \setminus \widehat{O}$ y para todo $i \in \widehat{A}$, $v|_{\widehat{A}\times O}(i, x_k) = 0$, de modo que x_k es un bien de valor nulo (ver la sección 3.2). Haciendo $U = O \setminus \widehat{O}$ y notando que $|U| \leq n - 1$ vemos, gracias a la proposición A.2, que existe $\widetilde{\pi}$ PELE para $(\widehat{A}, O, v|_{\widehat{A}\times O})$ si y solo si existe π' PELE para $(\widehat{A}, \widehat{O}, \widehat{v})$. Concluimos que (A, O, v) es una instancia positiva del PROBLEMA 1 si y solo si $(\widehat{A}, \widehat{O}, \widehat{v})$ es una instancia positiva del problema más general. □

Proposición B.3. *El problema 3DMx es NP-completo vía fops.*

Demostración. Para probar hardness, vamos a construir una reducción de primer orden desde 3-DIMENSIONAL-MATCHING (3DM), el cual es un problema NP-completo vía fops (teorema 4.74 de [10]). Recordemos, entonces, los problemas 3DM y 3DMx.

PROBLEMA: 3-DIMENSIONAL-MATCHING (3DM)

Instancia: Un conjunto C con n elementos y un conjunto de triples $M \subseteq C^3$

Pregunta: ¿Existe $M' \subseteq M$ tal que, para todo $i \in C$, existen triples

$$(i, a_1, b_1), (a_2, i, b_2), (a_3, b_3, i) \in M',$$

y tal que, para todo par de triples diferentes

$$(i_1, j_1, k_1), (i_2, j_2, k_2) \in M' \implies i_1 \neq i_2, j_1 \neq j_2 \text{ y } k_1 \neq k_2?$$

PROBLEMA: 3DMx

Instancia: Un conjunto C con n elementos y un conjunto de triples $M \subseteq C^3$ tales que, para todo $i \in C$, existen $j_l, k_l \in C$ con $l = 1, 2, 3$, tales que

$$(i, j_1, k_1), (j_2, i, k_2), (j_3, k_3, i) \in M.$$

Pregunta: ¿Existe $M' \subseteq M$ tal que, para todo $i \in C$, existen triples

$$(i, a_1, b_1), (a_2, i, b_2), (a_3, b_3, i) \in M',$$

y tal que, para todo par de triples diferentes

$$(i_1, j_1, k_1), (i_2, j_2, k_2) \in M' \implies i_1 \neq i_2, j_1 \neq j_2 \text{ y } k_1 \neq k_2?$$

Se puede pensar que toda instancia de 3DMx es una instancia positiva, pero eso no es cierto.

El vocabulario usado para estos problemas es

$$\omega = \langle D^3 \rangle.$$

Si $\mathcal{B} = \langle |\mathcal{B}|, D^{\mathcal{B}} \rangle$ es una ω-estructura, $|\mathcal{B}|$ es C y $D^{\mathcal{B}}$ es M.

Notar que la pregunta en 3DM y en 3DMx es la misma. De esta manera, podemos usar la sentencia existencial de segundo orden que define 3DM (demostración del teorema 4.74 en [10]), para definir 3DMx. Gracias al teorema 2.1, 3DMx es un problema en la clase NP.

Usaremos, además, la pregunta equivalente:

Pregunta: ¿Existe $M' \subseteq M$ tal que $|M'| = n$ y tal que, para todo par de triples diferentes

$$(i_1, j_1, k_1), (i_2, j_2, k_2) \in M' \implies i_1 \neq i_2, j_1 \neq j_2 \text{ y } k_1 \neq k_2?$$

Ahora, consideremos las fórmulas

$$\begin{aligned}
\alpha_{00} = \quad & x_1 = 1 \wedge x_2 = 0 \quad \wedge \quad y_1 = 0 \wedge y_2 = 0 \quad \wedge \quad z_1 = 0 \wedge z_2 = 1 \quad \wedge \\
& x_3 = y_3 \wedge y_3 = z_3, \\
\alpha_{01} = \quad & x_1 = 0 \wedge x_2 = 1 \quad \wedge \quad y_1 = 1 \wedge y_2 = 0 \quad \wedge \quad z_1 = 0 \wedge z_2 = 0 \quad \wedge \\
& x_3 = y_3 \wedge y_3 = z_3, \\
\alpha_{02} = \quad & x_1 = 1 \wedge x_2 = 1 \quad \wedge \quad y_1 = 1 \wedge y_2 = 1 \quad \wedge \quad z_1 = 1 \wedge z_2 = 1 \quad \wedge \\
& x_3 = y_3 \wedge y_3 = z_3, \\
\alpha_{03} = \quad & (1 < x_1 \vee 1 < x_2) \quad \wedge \quad x_1 = y_1 \wedge y_1 = z_1 \quad \wedge \quad x_2 = y_2 \wedge y_2 = z_2 \quad \wedge \\
& x_3 = y_3 \wedge y_3 = z_3, \\
\alpha_0 = \quad & \alpha_{00} \vee \alpha_{01} \vee \alpha_{02} \vee \alpha_{03}, \\
\alpha_1 = \quad & x_1 = 0 \wedge x_2 = 0 \quad \wedge \quad y_1 = 0 \wedge y_2 = 1 \quad \wedge \quad z_1 = 1 \wedge z_2 = 0, \\
\alpha_2 = \quad & x_1 = 0 \wedge x_2 = 0 \quad \wedge \quad y_1 = 0 \wedge y_2 = 0 \quad \wedge \quad z_1 = 0 \wedge z_2 = 0 \quad \wedge \\
& y_3 = 0 \wedge z_3 = 0, \\
\alpha_3 = \quad & x_1 = 0 \wedge x_2 = 1 \quad \wedge \quad y_1 = 0 \wedge y_2 = 1 \quad \wedge \quad z_1 = 0 \wedge z_2 = 1 \quad \wedge \\
& x_3 = 1 \wedge z_3 = 1, \text{ y} \\
\alpha_4 = \quad & x_1 = 1 \wedge x_2 = 0 \quad \wedge \quad y_1 = 1 \wedge y_2 = 0 \quad \wedge \quad z_1 = 1 \wedge z_2 = 0 \quad \wedge \\
& x_3 = 1 \wedge y_3 = 1.
\end{aligned}$$

Notemos que todas estas fórmulas son numéricas. Observemos, además, que las fórmulas α_{00}, α_{01}, α_{02}, y α_{03} son mutuamente excluyentes dos a dos. Esto implica que si una estructura satisface alguna de ellas, necesariamente no satisface las otras. Es también cierto que las fórmulas α_0, α_1, α_2, α_3, y α_4 son mutuamente excluyentes dos a dos. Así, sea

$$\rho : \text{STRUC}[\omega] \to \text{STRUC}[\omega]$$

la fop 3-aria definida por

$$\varphi_0 = \textbf{true}, \text{ y}$$

$$\begin{aligned}\varphi_1(\bar{x},\bar{y},\bar{z}) = \quad & \alpha_0 \quad \vee \\ & \Big(\alpha_1 \wedge D^3(x_3,y_3,z_3)\Big) \quad \vee \\ & \Big(\alpha_2 \wedge \neg D^3(x_3,y_3,z_3)\Big) \quad \vee \\ & \Big(\alpha_3 \wedge \neg D^3(x_3,y_3,z_3)\Big) \quad \vee \\ & \Big(\alpha_4 \wedge \neg D^3(x_3,y_3,z_3)\Big),\end{aligned}$$

con

$$\begin{aligned}&\bar{x} = (x_1,x_2,x_3),\\ &\bar{y} = (y_1,y_2,y_3), \text{ y}\\ &\bar{z} = (z_1,z_2,z_3).\end{aligned}$$

ρ toma una ω-estructura $\mathcal{B} = \langle |\mathcal{B}|, D^{\mathcal{B}}\rangle$, y la transforma en una ω-estructura $\rho(\mathcal{B}) = \langle |\rho(\mathcal{B})|, D^{\rho(\mathcal{B})}\rangle$. Concretamente, $|\rho(\mathcal{B})| = |\mathcal{B}|^3$, y $D^{\rho(\mathcal{B})}$ está conformado por los triples, y solo por los triples:

$$(0,0,i,\quad 0,1,j,\quad 1,0,k),\quad \text{con}\quad (i,j,k) \in D^{\mathcal{B}}, \tag{B.1}$$
$$(0,0,i,\quad 0,0,j,\quad 0,0,k),\quad \text{con}\quad (i,j,k) \notin D^{\mathcal{B}} \text{ y } j = k = 0, \tag{B.2}$$
$$(0,1,i,\quad 0,1,j,\quad 0,1,k),\quad \text{con}\quad (i,j,k) \notin D^{\mathcal{B}} \text{ e } i = k = 1, \tag{B.3}$$
$$(1,0,i,\quad 1,0,j,\quad 1,0,k),\quad \text{con}\quad (i,j,k) \notin D^{\mathcal{B}} \text{ e } i = j = 1, \tag{B.4}$$
$$(1,0,i,\quad 0,0,j,\quad 0,1,k),\quad \text{con}\quad i = j = k, \tag{B.5}$$
$$(0,1,i,\quad 1,0,j,\quad 0,0,k),\quad \text{con}\quad i = j = k, \tag{B.6}$$
$$(1,1,i,\quad 1,1,j,\quad 1,1,k),\quad \text{con}\quad i = j = k, \text{ y} \tag{B.7}$$
$$(l,m,i,\quad l,m,j,\quad l,m,k),\quad \text{con}\quad i = j = k, \text{ y } 1 < l \text{ o } 1 < m. \tag{B.8}$$

En efecto, los triples B.1, B.2, B.3, y B.4 provienen, respectivamente, de las fórmulas

$$\begin{aligned}&\alpha_1 \wedge D^3(x_3,y_3,z_3),\\ &\alpha_2 \wedge \neg D^3(x_3,y_3,z_3),\\ &\alpha_3 \wedge \neg D^3(x_3,y_3,z_3), \text{ y}\\ &\alpha_4 \wedge \neg D^3(x_3,y_3,z_3);\end{aligned}$$

mientras los triples B.5, B.6, B.7, y B.8 provienen, respectivamente, de las fórmulas α_{00}, α_{01}, α_{02}, y α_{03}.

Ahora vamos a verificar que $\rho(\mathcal{B})$ es una instancia de 3DMx para toda ω-estructura

$\mathcal{B}$. Sea $(p,q,r) \in |\rho(\mathcal{B})|$, con $\mathcal{B} \in \text{STRUC}[\omega]$. Comprobaremos que (p,q,r) aparece en las componentes primera, segunda, y tercera de elementos de $D^{\rho(\mathcal{B})}$.

Probaremos que (p,q,r) aparece en la primera componente de algún elemento de $D^{\rho(\mathcal{B})}$ por casos:

- $p = 0, q = 0$:
  1. Existen $j,k \in |\mathcal{B}|$ tales que $(r,j,k) \in D^{\mathcal{B}}$: ver los triples B.1.
  2. Para cualesquier $j,k \in |\mathcal{B}|$, $(r,j,k) \notin D^{\mathcal{B}}$: ver los triples B.2.
- $p = 0, q = 1$: ver los triples B.6.
- $p = 0, q \geq 2$: ver los triples B.8.
- $p = 1, q = 0$: ver los triples B.5.
- $p = 1, q = 1$: ver los triples B.7.
- $p = 1, q \geq 2$: ver los triples B.8.
- $p \geq 2$: ver los triples B.8.

Segunda componente:

- $p = 0, q = 0$: ver los triples B.5.
- $p = 0, q = 1$:
  1. Existen $i,k \in |\mathcal{B}|$ tales que $(i,r,k) \in D^{\mathcal{B}}$: ver los triples B.1.
  2. Para cualesquier $i,k \in |\mathcal{B}|$, $(i,r,k) \notin D^{\mathcal{B}}$: ver los triples B.3.
- $p = 0, q \geq 2$: ver los triples B.8.
- $p = 1, q = 0$: ver los triples B.6.
- $p = 1, q = 1$: ver los triples B.7.
- $p = 1, q \geq 2$: ver los triples B.8.
- $p \geq 2$: ver los triples B.8.

Tercera componente:

- $p = 0, q = 0$: ver los triples B.6.
- $p = 0, q = 1$: ver los triples B.5.
- $p = 0, q \geq 2$: ver los triples B.8.
- $p = 1, q = 0$:
  1. Existen $i,j \in |\mathcal{B}|$ tales que $(i,j,r) \in D^{\mathcal{B}}$: ver los triples B.1.
  2. Para cualesquier $i,j \in |\mathcal{B}|$, $(i,j,r) \notin D^{\mathcal{B}}$: ver los triples B.4.

- $p = 1, q = 1$: ver los triples B.7.
- $p = 1, q \geq 2$: ver los triples B.8.
- $p \geq 2$: ver los triples B.8.

Así, en vista de que $\mathcal{B} \in \text{STRUC}[\omega]$ y $(p, q, r) \in |\rho(\mathcal{B})|$ fueron tomados arbitrarios, concluimos que para toda ω-estructura $\mathcal{B}$, $\rho(\mathcal{B})$ es una instancia de 3DMx.

A continuación demostraremos que, para cualquier ω-estructura $\mathcal{B}$, esta es una instancia positiva de 3DM si y solo si $\rho(\mathcal{B})$ es una instancia positiva de 3DMx. Para lograrlo nos valdremos de la siguiente afirmación, que será demostrada al final.

Afirmación: Sea $\mathcal{B}$ una ω-estructura. Entonces, si $\rho(\mathcal{B})$ es una instancia positiva de 3DMx y $M' \subseteq D^{\rho(\mathcal{B})}$ es cualquiera de los conjuntos que satisface la pregunta de 3DMx, necesariamente ninguno de los triples B.2, B.3, o B.4 es parte de M'.

Sea $\mathcal{B}$ una ω-estructura de tamaño n.

($\Rightarrow$) Supongamos que $\mathcal{B}$ es una instancia positiva de 3DM. Entonces existe $N' \subseteq D^{\mathcal{B}}$ tal que $|N'| = n$ y tal que, para todo par de triples diferentes

$$(i_1, j_1, k_1), (i_2, j_2, k_2) \in N' \implies i_1 \neq i_2, j_1 \neq j_2 \text{ y } k_1 \neq k_2.$$

Tomemos $M' \subseteq D^{\rho(\mathcal{B})}$ con los triples de B.1 tales que $(i, j, k) \in N'$, y con aquellos en B.5, B.6, B.7, y B.8. Los primeros suman n, y el resto

$$3n + (n^3 - 4n) = n^3 - n.$$

Luego, M' consta de n^3 triples. Es más, para cada par de triples diferentes en M', sus primeras componentes son distintas, y lo mismo es cierto para aquellas en las posiciones segunda y tercera. Luego, $\rho(\mathcal{B})$ es una instancia positiva de 3DMx.

($\Leftarrow$) Asumamos, por contradicción, que $\rho(\mathcal{B})$ es una instancia positiva de 3DMx y que $\mathcal{B}$ es una instancia negativa de 3DM. Sea $M' \subseteq D^{\rho(\mathcal{B})}$ cualquiera de los conjuntos que satisface la pregunta de 3DMx. Por la afirmación hecha, ninguno de los triples B.2, B.3, o B.4 es parte de M', es decir, M' debe contener únicamente triples de B.1, B.5, B.6, B.7, o B.8. Es más, ya que todo elemento de $|\rho(\mathcal{B})|$ debe aparecer en las coordenadas primera, segunda y tercera de triples de M', M' necesariamente debe contener todos los triples de B.5, B.6, B.7, y B.8, los cuales suman $n^3 - n$. Así, M' debe contener exactamente n triples de B.1, pues $|M'| = n^3$, pero cualquier conjunto de n triples de B.1 se corresponde con un conjunto de n triples de $D^{\mathcal{B}}$, y ya que $\mathcal{B}$ es una instancia negativa de 3DM, cualquier conjunto de n triples de $D^{\mathcal{B}}$ debe ser tal que contiene elementos diferentes con alguna coordenada igual. Esto implica

que cualquier conjunto de n triples de B.1 debe contener elementos diferentes con alguna coordenada igual, y esto contradice el hecho de que M' satisface la pregunta de 3DMx.

Prueba de la afirmación: Supongamos $\rho(\mathcal{B})$ es una instancia positiva de 3DMx, y que $M' \subseteq D^{\rho(\mathcal{B})}$ es cualquiera de los conjuntos que satisface la pregunta asociada al problema.

Notemos que el triple

$$(1,0,0, \quad 0,0,0, \quad 0,1,0)$$

de B.5 debe pertenecer a M', pues es el único en $D^{\rho(\mathcal{B})}$ que tiene el elemento $(1,0,0) \in |\rho(\mathcal{B})|$ en la primera componente. Además, cualquiera los triples de B.2 tiene la forma

$$(0,0,i, \quad 0,0,0, \quad 0,0,0).$$

En vista de que los triples descritos son diferentes, y de que las segundas componentes de estos son las mismas, ninguno de los triples de B.2 puede ser parte de M'.

Por otro lado, observemos que el triple

$$(0,1,1, \quad 1,0,1, \quad 0,0,1)$$

de B.6 necesariamente es parte M', pues es el único en $D^{\rho(\mathcal{B})}$ que tiene el elemento $(0,0,1) \in |\rho(\mathcal{B})|$ en la tercera componente. Ahora, cualquiera los triples de B.3 tiene la forma

$$(0,1,1, \quad 0,1,j, \quad 0,1,1).$$

Los triples descritos son diferentes, y sus primeras componentes son las mismas. Así, ninguno de los triples de B.3 puede pertenecer a M'.

Finalmente, notemos que el triple

$$(1,0,1, \quad 0,0,1, \quad 0,1,1)$$

de B.5 debe pertenecer a M', pues es el único en $D^{\rho(\mathcal{B})}$ que tiene el elemento $(0,0,1) \in |\rho(\mathcal{B})|$ en la segunda componente. Es cierto también que cualquiera los triples de B.4 tiene la forma

$$(1,0,1, \quad 1,0,1, \quad 1,0,k).$$

Ya que los triples descritos son diferentes, y ya que las primeras componentes de estos son las mismas, ninguno de los triples de B.4 puede ser parte de M'. □

Apéndice C

Algoritmos

Para la implementación de los algoritmos se ha usado el lenguaje de programación Python, versión 3.9.7.

C.1. Problema 1

Se ha obtenido el algoritmo a partir de la sentencia que define el PROBLEMA 1 (ver el teorema 4.1).

```
import numpy as np

# La entrada es una matriz boolena R, nxn, con n >= 2.
# R[x,y] = True si y solo si la cláusula x contiene la variable proposicional
# y. La matriz R viene a representar la relación asociada a una nu-estructura
# del PROBLEMA 1.

# Fijado n, la matirz R se toma al azar. Se puede especificar una matriz
# R particular.
n = 2
R = np.empty((n,n), dtype=bool)

# R = np.zeros((n,n), dtype=bool)
# for x in range(0,n):
#     if x < n-1:
#         R[x,x] = True
#         R[x,x+1] = True
#     else:
#         R[x,x] = True
# R[n-1,0] = True
```

```
# La matriz M permite visualizar de mejor manera los datos de entrada del
# algoritmo. La entrada M[0,0] contiene el número de bienes. La entrada M[0,y],
# para y > 0, despliega el número y-1, correspondiente al bien, o variable
# proposicional, y-1. La entrada M[x,0], para x > 0, despliega el número x-1,
# correspondiente a la cláusula x-1. La submatriz correspondiente al resto de
# entradas es R con 1 reemplazando a True y 0 a False.
M = np.ones((n+1,n+1), dtype=int)
for y in range(0,n+1):
    for z in range(0,n+1):
        if y == 0 and z == 0:
            M[y,z] = n
        elif( y == 0 and z > 0):
            M[y,z] = z-1
        elif(z == 0 and y >0):
            M[y,z] = y-1
        else:
            if R[y-1,z-1] == False:
                M[y,z] = 0

# print("R es:")
# print(R)
# print()
print("M es:")
print(M)
print("Para x,y >= 1, M[x,y] = 1 ssi la cláusula x-1 contiene la variable")
print("y-1.")
print()

# Si alguna cláusula es vacía, es decir, si al menos una fila de R tiene solo
# False, los agentes son indiferentes. Si alguna cláusula tiene solo una
# variable proposicional, y el resto de cláusulas son no vacías, no existe
# asignación PELE (la instancia no verifica la sentencia \Psi_{2} del
# PROBLEMA 1).
x = 0
B1 = True # B1 será False si alguna cláusula es vacía.
B2 = True # B2 será False si alguna cláusula tiene solo una variable
# proposicional.
while x <= n-1 and B1 == True:
    y = 0
    z = 0
    while y <= n-1 and z < 2:
        if R[x,y] == True:
            z = z+1
        y = y+1
```

```
    if z == 2:
        x = x + 1
    elif z == 1:
        B2 = False
        x = x + 1
    else:
        B1 = False

if B1 == False:
    print("Instancia negativa: agentes indiferentes")
else:
    if B2 == False:
        print("Instancia negativa: agentes no indiferentes,")
        print("pero no existe asignación PELE.")
    else:
        # Verificar existe asignación PELE, es
        # decir, verificar la sentencia \Psi_{2} del PROBLEMA 1 es satisfecha.
        # Tomaremos conjuntos P, correspondiente a la variable de segundo orden
        # P en \Psi_{2}, en orden. Para eso crearemos un arreglo binario con n
        # entradas, Lista_binaria, que respresenta un número binario: el bien y
        # está en P si Lista_binaria[y] == True. Si el conjunto P así tomado no
        # verifica \Psi_{2}, tomaremos el sucesor del número binario.
        # No se tomará en cuenta los casos en que el número binario tiene solo
        # 0s o solo 1s, pues en esos casos el conjunto P correspondiente
        # no verifica \Psi_{2}.
        Lista_binaria = np.zeros(n, dtype=bool)
        Lista_binaria[n-1] = True
        P = {n-1}
        B4 = False # B4 será False mientras no se haya considerado todos los
        # conjuntos P mencionados.
        # Dado P, verificaremos toda cláusula tiene una variable proposicional
        # en P y otra que no esté en P, es decir, verificaremos que la
        # sentencia \Psi_{2} es satisfecha.
        B1 = False # B1 será True si el conjunto dado P verifica la sentencia
        # \Psi_{2}.
        while B4 == False and B1 == False:
            x = 0
            B2 = True # B2 será False si alguna cláusula no contiene
            # variable proposicional alguna en P o no contiene variable alguna
            # en el complemento de P.
            while x <= n-1 and B2 == True:
                # Verificamos primero la cláusula actual tiene una variable
                # proposicional en P.
                y = 0
```

```
        B3 = False # B3 será True si la cláusula actual tiene
        # variable proposicional en P.
        while y <= n-1 and B3 == False:
            if R[x,y] == True and y in P:
                B3 = True
            else:
                y = y + 1
        if B3 == False:
            B2 = B3
        else:
            # Ahora verificamos la cláusula actual tiene variable
            # proposicional en el complemento de P.
            z = 0
            B3 = False # B3 será True si la claúsula actual tiene
            # variable proposicional en el complemento de P.
            while z <= n-1 and B3 == False:
                if z == y:
                    z = z + 1
                else:
                    if R[x,z] == True and not(z in P):
                        B3 = True
                    else:
                        z = z + 1
            if B3 == False:
                B2 = B3
            else:
                x = x + 1
    B1 = B2
    # Si P no verifica \Psi_{2}, intentamos con otro conjunto P.
    if B2 == False:
        # Tomamos el sucesor del número binario asociado a
        # Lista_binaria.
        y = n-1
        B3 = Lista_binaria[y]
        while B3 == True and y > 0:
            y = y-1
            B3 = Lista_binaria[y]
        Lista_binaria[y] = True
        for z in range(y+1,n):
            Lista_binaria[z] = False
        # Si Lista_binaria tiene solo True, cambiar B4 a True.
        y = 0
        B3 = False # B3 será True si algún dígito del número binario es
        # False.
```

```
            while y <= n-1 and B3 == False:
                if Lista_binaria[y] == False:
                    B3 = True
                else:
                    y = y + 1
            B4 = not(B3)
            # Si el número binario tiene algún 0, tomamos el conjunto P
            # asociado.
            if B4 == False:
                P = set()
                for z in range(0,n):
                    if Lista_binaria[z] == True:
                        P.add(z)
if B1 == False:
    print("Instancia negativa: agentes no indiferentes,")
    print("pero no existe asignación PELE.")
else:
    print("Existe asignación PELE y completa pi con:")
    print("pi(0)=")
    print(P)
    Q = set()
    for z in range(0,n):
        if not(z in P):
            Q.add(z)
    print("pi(1)=")
    print(Q)
```

C.2. Problema 2

Se puede probar que en el caso del PROBLEMA 2, existe asignación PELE π si y solo si existe asignación PELE y completa π'. De esta manera, reducimos el espacio de búsqueda de asignaciones considerando solo las asignaciones completas. Además, como se mencionó en el capítulo 5, la sentencia que define el PROBLEMA 2 puede ser reemplazada por una menos compleja. Concretamente, no es indispensable una variable de relación $\mathbf{S^5}$ de aridad 5; basta con $\mathbf{S^4}$. Las siguientes fórmulas reflejan los cambios respectivos; estas son usadas en el algoritmo. El algoritmo fue obtenido a partir de la sentencia que define el PROBLEMA 2 (ver teorema 4.2).

$$\Phi_{sum_{inic}}(\mathbf{S^4}, \mathbf{P^2}) = (\forall u)(\forall x)(\forall z)(\mathbf{S^4}(u, x, 0, z) \leftrightarrow (\mathbf{P^2}(u, 0) \wedge z = max \wedge V^3(x, 0, max)))$$

$$\Phi_{carry}(\mathbf{S^4}, u, x, w, y, z) = \mathbf{S^4}(u, x, w, max) \wedge V^3(x, y, max) \wedge (\forall r)((z < r \wedge r < max) \rightarrow \mathbf{S^4}(u, x, w, r))$$

$$\Phi_{sum_{\neq max}}(\mathbf{S^4}, u, x, w, y, z) = \mathbf{S^4}(u, x, w, z) \oplus \Phi_{carry}(\mathbf{S^4}, u, x, w, y, z)$$

$$\Phi_{sum_{max}}(\mathbf{S^4}, u, x, w, y) = \mathbf{S^4}(u, x, w, max) \oplus V^3(x, y, max)$$

$$\begin{aligned}\Phi_{sum-operar}(\mathbf{S^4}, \mathbf{P^2}) =& (\forall u)(\forall x)(\forall w)(\forall y) \\ & (\mathrm{SUC}(w, y) \rightarrow \\ & ((\mathbf{S^4}(u, x, y, max) \leftrightarrow ((\neg \mathbf{P^2}(u, y) \wedge \mathbf{S^4}(u, x, w, max)) \\ & \vee (\mathbf{P^2}(u, y) \wedge \Phi_{sum_{max}}(\mathbf{S^4}, u, x, w, y)))) \\ & \wedge (\forall z)(z \neq max \rightarrow \\ & (\mathbf{S^4}(u, x, y, z) \leftrightarrow ((\neg \mathbf{P^2}(u, y) \wedge \mathbf{S^4}(u, x, w, z)) \\ & \vee (\mathbf{P^2}(u, y) \wedge \Phi_{sum_{\neq max}}(\mathbf{S^4}, u, x, w, y, z)))))))\end{aligned}$$

$$\begin{aligned}\Phi_{compara_{menor}}(\mathbf{T^4}, \mathbf{S^4}, u, x) =& (\exists z)(\neg \mathbf{T^4}(u, x, max, z) \wedge \mathbf{S^4}(x, x, max, z) \\ & \wedge (\forall r)(r < z \rightarrow (\mathbf{T^4}(u, x, max, r) \rightarrow \mathbf{S^4}(x, x, max, r))))\end{aligned}$$

$$\Phi_{compara_{=}}(\mathbf{T^4}, \mathbf{S^4}, u, x) = (\forall z)(\mathbf{T^4}(u, x, max, z) \leftrightarrow \mathbf{S^4}(x, x, max, z))$$

$$\Phi_{compara_{\leq}}(\mathbf{S^4}, u, x) = \Phi_{compara_{menor}}(\mathbf{S^4}, \mathbf{S^4}, u, x) \vee \Phi_{compara_{=}}(\mathbf{S^4}, \mathbf{S^4}, u, x)$$

$$\Phi_{LE}(\mathbf{S^4}) = (\forall u)(\forall x)\Phi_{compara_{\leq}}(\mathbf{S^4}, u, x)$$

```
import random
import numpy as np

# La entrada es una matriz booleana V, nxnxn, con n >= 2. La matriz V viene a
# representar la relación asociada a una sigma-estructura del PROBLEMA 2. n
# es el tamaño de la sigma-estructura.

# Las asignaciones serán representadas por matrices binarias P, con
# P[x,y] = True ssi el bien y es asignado al agente x.

# En lo que sigue:
# S : captura las sumas asociadas al cálculo de la utilidad que percibe un
#     agente por el paquete asignado a otro, o al mismo, agente.
# u : representa un agente.
# x : representa un agente.
# w : representa un bien.
# y : representa un bien.
# z : representa un bit.

# Definimos funciones basadas en las fórmulas que se desarrolló para el
# PROBLEMA 2.


# La siguiente función está basada en la fórmula \gamma_{0-1}.
def gamma_01(n,V):
    """
    Returns
    -------
    B : True ssi los agentes declaran por los bienes utilidades en {0,1}.
    """
    x = 0
    B = True
    # B será False si existen un agente x y un bien y tales que x declara por y
    # un valor mayor a 1.
    while(x <= n-1 and B == True):
        y = 0
        while(y <= n-1 and B == True):
            z = 0
            while(z < n-1 and B == True):
                if V[x,y,z] == True:
                    B = False
```

```
                else:
                    z = z+1
            if B == True:
                y = y+1
        if B == True:
            x = x+1
    return B


# La siguiente función está basada en la fórmula \Phi_{PE}.
def Phi_PE(n,V,P):
    """
    Returns
    -------
    D : True ssi la asignación representada por P es Pareto-eficiente.
    """
    D = True
    y = 0
    while(y <= n-1 and D == True):
        E = False
        # E será True si el bien y es deseado.
        x = 0
        while(x <= n-1 and E == False):
            if V[x,y,n-1] == True:
                E = True
            else:
                x = x+1
        if E == True:
            # E será False si el bien y es asignado a un agente que
            # lo desea.
            x = 0
            while(x <= n-1 and E == True):
                if V[x,y,n-1] == True and P[x,y] == True:
                    E = False
                else:
                    x = x+1
            if E == True:
                D = False
            else:
                y = y+1
        else:
            y = y+1
    return D
```

```
# La siguiente función está basada en la fórmula \Phi_{carry}.
def Phi_carry(n,V,S,u,x,w,y,z):
    """
    Returns
    -------
    True ssi el carry correspondiente es un 1.
    """
    B = S[u,x,w,n-1] and V[x,y,n-1]
    if B == True:
        r = z+1
        while(r < n-1 and B == True):
            if not(S[u,x,w,r]):
                B = False
            else:
                r = r+1
    return B


# La siguiente función está basada en la fórmula \Phi_{sum_{\neq max}}.
def Phi_sumneqmax(n,V,S,u,x,w,y,z):
    """
    Returns
    -------
    True ssi el bit respectivo es un 1.
    """
    return S[u,x,w,z] ^ Phi_carry(n,V,S,u,x,w,y,z)


# La siguiente función está basada en la fórmula \Phi_{sum_{max}}.
def Phi_summax(n,V,S,u,x,w,y):
    """
    Returns
    -------
    True ssi el bit respectivo es un 1.
    """
    return S[u,x,w,n-1] ^ V[x,y,n-1]


# La siguiente función está basada en la fórmula \Phi_{compara_{menor}}.
def Phi_comparamenor(n,S,u,x):
    """
    Returns
    -------
```

```
    True ssi la utilidad que percibe x por el paquete asignado a u es menor
    que la utilidad que x percibe por el paquete asignado a x.
    """
    B = False
    z = n-1
    while(z >= 0 and B == False):
        if S[u,x,n-1,z] == False and S[x,x,n-1,z] == True:
            B = True
            r = z-1
            while(r >= 0 and B == True):
                if S[u,x,n-1,r] == True and S[x,x,n-1,r] == False:
                    B = False
                    z = z-1
                else:
                    r = r-1
        else:
            z = z-1
    return B


# La siguiente función está basada en la fórmula \Phi_{compara_{=}}.
def Phi_comparaigual(n,S,u,x):
    """
    Returns
    -------
    True ssi la utilidad que percibe x por el paquete asignado a u es igual
    que la utilidad que x percibe por el paquete asignado a x.
    """
    B = True
    z = 0
    while(z <= n-1 and B == True):
        if S[u,x,n-1,z] != S[x,x,n-1,z]:
            B = False
        else:
            z = z+1
    return B


# La siguiente función está basada en la fórmula \Phi_{LE}.
def Phi_LE(n,S):
    """
    Returns
    -------
    True ssi la asignación representada por P está libre de envidia (S contiene
```

```
    datos asociados a P).
    """
    B = True
    u = 0
    while(u <= n-1 and B == True):
        x = 0
        while(x <= n-1 and B == True):
            if u != x:
                if (not(Phi_comparamenor(n,S,u,x)) and
                    not(Phi_comparaigual(n,S,u,x))):
                    B = False
                else:
                    x = x+1
            else:
                x = x+1
        if B == True:
            u = u+1
    return B


# La entrada es una matriz booleana V, nxnxn, con n >= 2.
# Fijado n, la matirz V se puede toma al azar:
n = 5
V = np.empty((n,n,n), dtype=bool)

# V = np.zeros((n,n,n), dtype=bool)
# for x in range(0,n):
#     for y in range(0,n):
#         if random.random() > 0.5:
#             V[x,y,n-1] = True

# Se puede especificar una matriz V particular.

# La matriz M, de tamaño (n+1)x(n+1), permite visualizar de mejor manera los
# datos de entrada del algoritmo. La entrada 00 contiene el número de bienes.
# La entrada 0y, para y > 0, despliega el número y-1, correspondiente al bien
# y-1. La entrada x0, para x > 0, despliega el número x-1, correspondiente
# al agente x-1. Para la submatriz correspondiente al resto de entradas,
# M(x,y) = k ssi el agente x-1 asigna el valor k al bien y-1.
M = np.zeros((n+1,n+1), dtype=int)
for x in range(0,n+1):
    for y in range(0,n+1):
        if x == 0 and y == 0:
            M[x,y] = n
```

```
        elif( x == 0 and y > 0):
            M[x,y] = y-1
        elif(y == 0 and x >0):
            M[x,y] = x-1
        else:
            for z in range(0,n):
                if V[x-1,y-1,z] == True:
                    M[x,y] = M[x,y] + pow(2,n-(z+1))

# print("V es:")
# print(V)
# print()
print("M es:")
print(M)
print("Para x,y >= 1,")
print("M(x,y) = k ssi el agente x-1 asigna el valor k al bien y-1.")
print()

# Ahora verificaremos que los agentes declaran por los bienes utilidades en
# {0,1}.
B = gamma_01(n,V)

if B == False:
    print("Instancia negativa:")
    print("existe al menos un agente que ha declarado un valor mayor a 1")
    print("por algún bien.")
else:
    # Los agentes declaran por los bienes utilidades en {0,1}. A continuación
    # verificaremos existe asignación PELE.Tomaremos en cuenta solo las
    # asignaciones completas. Comenzaremos intentando con la asignación que
    # asigna todos los bienes al agente 0. La última asignación con la cual
    # intentaremos es aquella que otorga todos los bienes al agente n-1.
    # Para generar las asignaciones utilizaremos una lista L de tamaño n, con
    # L[y] = x ssi el bien y es asignado al agente x.
    L = np.zeros(n, dtype=int)

    # La asignación respectiva es representada por una matriz binaria P, con
    # P[x,y] = True ssi el bien y es asignado al agente x.
    P = np.zeros((n,n), dtype=bool)
    for y in range(0,n):
        x = L[y]
        P[x,y] = True

    B = False
```

```
# B será False mientras no se haya considerado todas las asignaciones
# mencionadas.

C = False
# C será True si la asignación correspondiente es PELE.

while(B == False and C == False):
    # Verificaremos primero la asignación es Pareto-eficiente. Si no es
    # Pareto-eficiente, intentaremos con otra asignación.
    D = Phi_PE(n,V,P)

    if D == True:
        # Si la asignación es Pareto-eficiente, verificamos a continuación
        # está libre de envidia. Para lograrlo, construimos un arreglo
        # binario S, de tamaño nxnxnxn. Este juega el rol de la relación
        # capturada por la variable relacional S en la sentencia que
        # define el PROBLEMA 2, es decir, a través de esta relación vamos
        # a sumar.

        # Incializamos la suma basándonos en la fórmula \Phi_{sum_{inic}}.
        S = np.zeros((n,n,n,n), dtype=bool)
        for u in range(0,n):
            for x in range(0,n):
                S[u,x,0,n-1] = P[u,0] and V[x,0,n-1]

        # A continuación calculamos el resto de sumas parciales basándonos
        # en la fórmula \Phi_{sum-operar}.
        for u in range(0,n):
            for x in range(0,n):
                for w in range(0,n-1):
                    y = w+1
                    if ((P[u,y] == False and S[u,x,w,n-1] == True) |
                        (P[u,y] == True and Phi_summax(n,V,S,u,x,w,y))):
                        S[u,x,y,n-1] = True
                    for z in range(0,n-1):
                        if ((P[u,y] == False and S[u,x,w,z] == True) |
                            (P[u,y] == True and
                             Phi_sumneqmax(n,V,S,u,x,w,y,z))):
                            S[u,x,y,z] = True

        # A continuación verificamos la asignación actual está libre de
        # envidia.
        D = Phi_LE(n,S)
```

```
    if D == True:
        C = True
        print("Instancia positiva:")
        print("Asignación PELE y completa A encontrada:")
        A = np.zeros((n+1,n+1), dtype=int)
        for x in range(0,n+1):
            for y in range(0,n+1):
                if x == 0 and y == 0:
                    A[x,y] = n
                elif( x == 0 and y > 0):
                    A[x,y] = y-1
                elif(y == 0 and x >0):
                    A[x,y] = x-1
                else:
                    if P[x-1,y-1] == True:
                        A[x,y] = 1
        print(A)
        print("Para x,y>=1, A[x,y]=1 ssi y-1 es asignado a x-1.")

        # Se puede verificar que la asignación está libre de envidia
        # visualizando la matriz Util. Util[x,u] es la  utilidad que x
        # asigna al paquete que recibe u. Basta comprobar que los
        # elementos de la diagonal son mayores o iguales que los
        # elementos de la respectiva fila.
        Util = np.matmul(M[1:n+1,1:n+1], A[1:n+1,1:n+1].transpose())
        print()
        print("Util:")
        print(Util)
        print("Util[x,u] es la  utilidad que x asigna al paquete que")
        print("recibe u.")

if D == False:
    # Si la asignación no es PELE, intentamos con otra
    # asignación, siempre que no hayamos considerado todas.
    y = n-1
    x = L[y]
    while(x == n-1 and y > 0):
        y = y-1
        x = L[y]
    if y == 0 and x == n-1:
        B = True
    else:
        L[y] = L[y]+1
        for z in range(y+1,n):
```

```
                L[z] = 0
            P = np.zeros((n,n), dtype=bool)
            for y in range(0,n):
                x = L[y]
                P[x,y] = True

if C == False:
    print("Instancia negativa: no existe asignación PELE.")
```

Bibliografía

[1] Borges, N. *Técnicas sintácticas y combinatorias en el estudio de la complejidad computacional*. PhD thesis, Universidad Simón Bolívar, 2011.

[2] Borges, N. y Bonet, B. Universal First-Order Logic is Superfluous for NL, P, NP and coNP. *Logical Methods in Computer Science*, Volume 10, Issue 1, March 2014.

[3] Borges, N. y Pin, E. Universal first-order logic is superfluous in the second level of the polynomial-time hierarchy. *Log. J. IGPL*, 27(6):895–909, 2019.

[4] Bouveret, S. y Lang, J. Efficiency and Envy-freeness in Fair Division of Indivisible Goods: Logical Representation and Complexity. *Journal of Artificial Intelligence Research*, Volume 32, 2008.

[5] Dalen, D. *Logic and Structure*. Springer London, 2013.

[6] de Keijzer, B., Bouveret, S., Klos, T., y Zhang, Y. On the Complexity of Efficiency and Envy-Freeness in Fair Division of Indivisible Goods with Additive Preferences. En Rossi, F. y Tsoukias, A., editors, *Algorithmic Decision Theory*, págs. 98–110, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[7] Ebbinghaus, H. y Flum, J. *Finite Model Theory*. Springer, 1999.

[8] Garey, M. R. y Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1979.

[9] Immerman, N. *Descriptive Complexity*. Springer New York, NY, 1999.

[10] Medina-Peralta, J. *A descriptive approach to the class NP*. PhD thesis, University of Massachusetts Amherst, 1997.

[11] Mendelson, E. *Introduction to Mathematical Logic*. Routledge, 2015.