



FACULTAD DE MATEMÁTICA, ASTRONOMÍA,
FÍSICA Y COMPUTACIÓN

TRABAJO ESPECIAL DE LA LICENCIATURA EN
CIENCIAS DE LA COMPUTACIÓN

Algoritmos para decidir definibilidad de relaciones en fragmentos de primer orden

Autor:
Pablo VENTURA

Director:
Dr. Miguel CAMPERCHOLI

Febrero de 2016

Abstract

The definability of a relation for a class of structures, for some first order fragment, holds a semantic notion associated with its bases in preservation properties of different kinds of morphisms. In this work, we propose algorithms to decide definability in an automatic way; also, algorithms to generate the algebra of definable relations for a class of structures are provided. Beside presenting the algorithms, we'll introduce the package *Definability*, used to implement the algorithms. Documentations of its use will be provided with some examples. Lastly, a theorem about the characterization of binary definable relations by positive existentials on distributive lattices will be proved. The theorem was inspired with the use of the package.

Resumen

La definibilidad de una relación en una clase de estructuras para algún fragmento de primer orden tiene una noción semántica asociada fundamentada en propiedades de preservación de diferentes tipos de morfismos. Este trabajo propone algoritmos para decidir definibilidad automáticamente como así también algoritmos para generar el álgebra de relaciones definibles para una clase de estructuras. Además de presentar los algoritmos, se presenta *Definability*, un paquete que implementa los algoritmos y se documenta su uso con algunos ejemplos. Luego se prueba un teorema sobre la caracterización de las relaciones binarias definibles por existenciales positivas en reticulados distributivos, que fue inspirado con la utilización de nuestro paquete.

Agradecimientos

A Inés, por ayudarme tanto y soportar los fines de semana de estudio,
a mi familia, por el apoyo y el incentivo,
a mis amigos, dentro y fuera de la facultad,
a Camper, por el entusiasmo y la confianza,
y a todo FaMAF, por la emoción de cada clase.

Índice general

1. Introducción	9
1.1. El problema	9
1.2. La solución teórica	9
1.2.1. Ejemplo	10
1.3. Objetivos	10
1.4. Estructura del trabajo	11
2. Teoremas de definibilidad	13
2.1. Preliminares	13
2.2. Definibilidad de primer orden	19
2.3. Definibilidad por fórmulas abiertas	20
2.4. Definibilidad por fórmulas abiertas positivas	21
2.5. Definibilidad por fórmulas existenciales	22
2.6. Definibilidad por fórmulas existenciales positivas	24
2.7. Definibilidad por conjunción de atómicas y primitivas positivas	25
3. Algoritmos de chequeo de definibilidad	31
3.1. Definiciones e ideas preliminares	31
3.2. Preprocesamiento	32
3.3. Definibilidad abierta	33
3.4. Definibilidad abierta-positiva	35
3.5. Definibilidad de primer orden	39
3.6. Definibilidad existencial	39
3.7. Definibilidad existencial-positiva	39
3.8. Detección de homomorfismos	40
3.8.1. CSP	40
3.8.2. CSP para calcular homomorfismos	40
3.9. Generación de subestructuras	41
4. Álgebras de Lindenbaum	43
4.1. Definiciones e ideas básicas	43
4.2. Algoritmos	44
4.3. Ejemplos y aplicación	47
4.3.1. Utilizando el paquete Definability para SageMath	47
4.3.2. Caracterización de las relaciones binarias definibles por existenciales positivas en reticulados distributivos	50

5. Documentación de Definability	53
5.1. Introducción	53
5.2. Arquitectura del paquete	53
5.2.1. Programas externos necesarios	53
5.2.1.1. SageMath	54
5.2.1.2. Minion	54
5.2.1.3. LADR	54
5.2.2. Módulos y organización del código fuente	54
5.3. Instalación	55
5.3.1. Instalación del paquete en SageMath	55
5.4. Uso del paquete	55
5.5. Generación y entrada de estructuras	55
5.5.1. Generación de estructuras de una teoría de primer orden	56
5.5.2. Entrada manual de una estructura	56
5.6. Chequeo de definibilidad	57
5.7. Generación de álgebras de Lindenbaum	58
6. Conclusiones y trabajo futuro	61
6.1. Conclusiones	61
6.2. Trabajo futuro	61

Capítulo 1

Introducción

En nuestra experiencia, los buenos algoritmos son descubiertos al hacer un análisis formal y/o matemático del problema a resolver. En este caso nos proponemos el desarrollo de algoritmos para resolver un problema básico de la teoría de modelos finitos: decidir definibilidad de relaciones en el lenguaje de primer orden. La motivación inicial es la explotación de resultados teóricos recientes que caracterizan la definibilidad en términos semánticos presentados en Campercholi y Vaggione 2015.

1.1. El problema

Sean \mathcal{L} un lenguaje de primer orden y \mathcal{K} una clase de \mathcal{L} -estructuras. Dado un símbolo de relación n -ario $R \in \mathcal{L}$ diremos que la fórmula $\varphi(x_1, \dots, x_n)$ *define* a R en \mathcal{K} si

$$\mathcal{K} \models \forall \bar{x} \, R(\bar{x}) \leftrightarrow \varphi(\bar{x}).$$

Problema 1. Dados:

- \mathcal{L} un lenguaje de primer orden finito,
- \mathcal{K} un conjunto finito de estructuras finitas de \mathcal{L} ,
- $R \in \mathcal{L}$ un símbolo de relación, y
- Σ un conjunto recursivo de fórmulas de $\mathcal{L} - \{R\}$,

decidir si hay una fórmula de Σ que define a R en \mathcal{K} .

1.2. La solución teórica

La tipología del problema sugiere que su complejidad computacional en el caso general será muy elevada. Una solución por «fuerza bruta» requiere la construcción de todas las relaciones Σ -definibles en \mathcal{K} para poder dar una respuesta negativa. Nuestro enfoque es evitar este costo explotando propiedades de preservación que pueda tener el conjunto Σ .

En el trabajo Campercholi y Vaggione 2015 se presentan condiciones semánticas equivalentes a la existencia de una \mathcal{L} -fórmula $\varphi(\bar{x})$ que define a R en \mathcal{K} ,

donde φ pertenece a un fragmento de primer orden específico, como las fórmulas abiertas, las abiertas positivas, las existenciales, etc.

Por ejemplo, en el caso de la definibilidad por una fórmula abierta (i.e. fórmulas sin cuantificadores) el trabajo establece que (cf. 15) dado \mathcal{L} un lenguaje de primer orden, $R \in \mathcal{L}$ un símbolo de relación n -ario y \mathbf{A} una \mathcal{L} -estructura, los siguientes son equivalentes:

1. Hay una fórmula abierta que define R en \mathbf{A} .
2. Todo isomorfismo $\sigma : \mathbf{A}_0 \rightarrow \mathbf{B}_0$ donde \mathbf{A}_0 y \mathbf{B}_0 son subestructuras en $\mathcal{L} - \{R\}$ de \mathbf{A} , preserva R (i.e. si $(a_1, \dots, a_n) \in R$, entonces $(\sigma(a_1), \dots, \sigma(a_n)) \in R$).

Al utilizar este resultado teórico, decidir definibilidad se convierte en un problema de búsqueda y chequeo de flechas entre estructuras, siendo mucho mas manejable computacionalmente.

1.2.1. Ejemplo

Supongamos que se quiere decidir la definibilidad por fórmulas abiertas de la relación unaria P , dada en el siguiente reticulado.



Figura 1.2.1: Reticulado 2×2 con una relación unaria P

A la luz del resultado visto anteriormente es fácil notar que entre el subreticulado formado por el elemento 1 y el formado por el elemento 2 hay un isomorfismo σ que no preserva R , ya que $1 \in R$ pero $\sigma(1) = 2 \notin R$. Esto nos permite asegurar que no existe una fórmula abierta que defina la relación unaria P en el tipo de los reticulados.

1.3. Objetivos

A pesar de la ayuda que representan estas condiciones semánticas a la hora de decidir definibilidad, es claro que con estructuras mas complejas la cantidad de morfismos a chequear por preservación puede volverse enorme, hasta ser imposible de manejar manualmente. Nuestro objetivo es el desarrollo e implementación de algoritmos eficientes para decidir automáticamente definibilidad de relaciones en clases finitas de estructuras finitas.

En la búsqueda de eficiencia a la hora de encontrar morfismos modelizamos el problema como un *Constraint Satisfaction Problem (CSP)*. Este es un problema computacional que ha sido ampliamente estudiado y para el que hay buenas

implementaciones de solucionadores de CSP's (e.g. Minion Gent, Jefferson y Miguel 2006).

Para nuestros algoritmos, probamos algunos resultados teóricos sobre un subconjunto de morfismos generador por composición de todo el conjunto de morfismos. A partir de estos resultados diseñamos algoritmos que realizan los chequeos de preservación solo sobre el conjunto generador.

Como segundo objetivo pretendimos desarrollar algoritmos eficientes a la hora de obtener el álgebra de relaciones definibles para una clase de estructuras. Aquí nos basamos en los teoremas de representación de Stone y Birkhoff, para buscar tan solo los elementos join-irreducibles en estas álgebras.

1.4. Estructura del trabajo

El trabajo se encuentra organizado de la siguiente forma:

En el Capítulo 2 se prueban algunos lemas fundamentales para luego probar los Teoremas de definibilidad de relaciones con fórmulas de primer orden, abiertas, abiertas positivas, existenciales, existenciales positivas, conjunción de atómicas y fórmulas primitivas positivas.

En el Capítulo 3, presentamos los algoritmos para chequeo de definibilidad en fórmulas de primer orden, abiertas, abiertas positivas, existenciales y existenciales positivas. Antes de cada algoritmo presentamos los resultados teóricos sobre el subconjunto de morfismos que genera por composición a todo el conjunto de morfismos a chequear. Luego de los algoritmos mostramos como modelizar la búsqueda de homomorfismos mediante un CSP, y finalizamos el capítulo con la generación de subestructuras.

En el Capítulo 4, presentamos los algoritmos para generar el álgebra de relaciones definibles de una estructura. Luego mostramos algunos ejemplos mediante el uso de nuestro paquete *Definability* sobre reticulados. Finalizamos el capítulo con un teorema que caracteriza las relaciones binarias definibles por existenciales positivas en los reticulados distributivos, inspirado en los ejemplos probados sobre el paquete.

En el Capítulo 5, presentamos nuestro paquete *Definability* para el software matemático SageMath, documentando su instalación y su uso.

En el Capítulo 6, se presentan nuestras conclusiones y las posibles continuaciones a este trabajo.

Capítulo 2

Teoremas de definibilidad

En esta sección presentamos resultados que caracterizan la definibilidad, en diferentes fragmentos de primer orden, de relaciones sobre familias de estructuras. Estas caracterizaciones reducen la pregunta de si una relación es definible a la pregunta de si dicha relación es preservada por una colección de morfismos que depende del fragmento de primer orden considerado. Por ejemplo, (cf. 15) dados \mathcal{L} un lenguaje de primer orden, $R \in \mathcal{L}$ un símbolo de relación n -ario y \mathbf{A} una \mathcal{L} -estructura, entonces hay una fórmula abierta que define R en \mathbf{A} , sii todo isomorfismo $\sigma : \mathbf{A}_0 \rightarrow \mathbf{B}_0$ donde \mathbf{A}_0 y \mathbf{B}_0 son subestructuras en $\mathcal{L} - \{R\}$ de \mathbf{A} , preserva R (i.e. si $(a_1, \dots, a_n) \in R$, entonces $(\sigma(a_1), \dots, \sigma(a_n)) \in R$).

Estos resultados, expuestos originalmente en Campercholi y Vaggione 2015, constituyen el punto de partida para nuestro estudio del chequeo computacional de la definibilidad. Los algoritmos desarrollados pueden encontrarse en los capítulos siguientes.

2.1. Preliminares

Lo primero es definir los fragmentos de primer orden que nos interesan. Definimos:

$$\begin{aligned}\text{Fo}(\mathcal{L}) &= \{\varphi : \varphi \text{ es una } \mathcal{L}\text{-fórmula}\}, \\ \text{Op}(\mathcal{L}) &= \{\varphi : \varphi \text{ es una } \mathcal{L}\text{-fórmula abierta}\}, \\ \text{Op}^+(\mathcal{L}) &= \{\varphi : \varphi \text{ es una } \mathcal{L}\text{-fórmula abierta positiva}\}, \\ \text{E}(\mathcal{L}) &= \{\varphi : \varphi \text{ es una } \mathcal{L}\text{-fórmula existencial}\}, \\ \text{E}^+(\mathcal{L}) &= \{\varphi : \varphi \text{ es una } \mathcal{L}\text{-fórmula existencial positiva}\}, \\ \pm\text{At}(\mathcal{L}) &= \text{At}(\mathcal{L}) \cup \{\neg\alpha : \alpha \in \text{At}(\mathcal{L})\}, \\ \text{At}(\mathcal{L}) &= \{\mathcal{L}\text{-fórmulas atómicas}\}.\end{aligned}$$

Una fórmula es abierta si no tiene ocurrencias de \forall, \exists . Es positiva si no tiene ocurrencias de $\neg, \rightarrow, \leftrightarrow$. Una fórmula es existencial si es de la forma $\exists \bar{x} \varphi(\bar{x}, \bar{y})$ con $\varphi(\bar{x}, \bar{y})$ una fórmula abierta, mientras que es existencial positiva si $\varphi(\bar{x}, \bar{y})$ es abierta positiva. Una fórmula es conjunción de atómicas si es de la forma

$\varphi_1(\bar{x}) \wedge \cdots \wedge \varphi_n(\bar{x})$, donde cada $\varphi_i \in \text{At}(\mathcal{L})$. Una fórmula es primitiva positiva si es de la forma $\exists \bar{y} \varphi(\bar{x}, \bar{y})$ con $\varphi(\bar{x}, \bar{y})$ conjunción de atómicas.

Ahora definiremos el concepto de que una fórmula defina a una relación para una clase de estructuras.

Definición 2. Sean $\mathcal{L} \subseteq \mathcal{L}'$ lenguajes de primer orden y $R \in \mathcal{L}' - \mathcal{L}$ un símbolo de relación n -ario y $\Sigma(\mathcal{L}) \subseteq \text{Fo}(\mathcal{L})$. Diremos que R es *definible* en $\Sigma(\mathcal{L})$ sobre una familia \mathcal{K} de \mathcal{L}' -estructuras cuando exista una \mathcal{L} -fórmula $\varphi(x_1, \dots, x_n) \in \Sigma(\mathcal{L})$ tal que para todo $\mathbf{A} \in \mathcal{K}$ y todas $a_1, \dots, a_n \in A$

$$(a_1, \dots, a_n) \in R^{\mathbf{A}} \iff \varphi(a_1, \dots, a_n)$$

Luego definimos la preservación de una relación por parte de una función.

Definición 3. Dados A, B conjuntos, $R^A \subseteq A^n$, $R^B \subseteq B^n$, diremos que una función $\gamma : A \rightarrow B$ preserva R si para toda tupla $(a_1, \dots, a_n) \in R^A$ tenemos que si $(a_1, \dots, a_n) \in R^A$ implica que $(\sigma(a_1), \dots, \sigma(a_n)) \in R^B$.

Durante las siguientes pruebas, muchas veces hablaremos de conjuntos de formulas, por lo que dado un conjunto de fórmulas Δ , escribiremos $\Delta(\bar{x})$ para anunciar que cada una de las fórmulas en Δ tiene sus variables libres contenidas en la tupla \bar{x} , y que consideramos cada $\delta \in \Delta(\bar{x})$ declarada $\delta = \delta(\bar{x})$. Si \mathbf{A} es una estructura y \bar{a} es una tupla de elementos de A , escribiremos $\mathbf{A} \models \Delta[\bar{a}]$ cuando $\mathbf{A} \models \delta[\bar{a}]$ para cada $\delta \in \Delta(\bar{x})$.

Además, dado que muchas veces habrá dos lenguajes de primer orden involucrados, necesitamos definir el *reducto*. Dados $\mathcal{L} \subseteq \mathcal{L}'$ lenguajes de primer orden y \mathbf{A} una \mathcal{L}' -estructura, usaremos $\mathbf{A}_{\mathcal{L}}$ para indicar el reducto de \mathbf{A} al lenguaje \mathcal{L} . Si \mathbf{A}, \mathbf{B} son \mathcal{L} -estructuras, usaremos $\mathbf{A} \leq \mathbf{B}$ para expresar que \mathbf{A} es subestructura de \mathbf{B} . Dada una estructura \mathbf{A} y $\bar{a} = a_1, \dots, a_n \in A$ utilizaremos $\langle \bar{a} \rangle^{\mathbf{A}}$ para denotar la subestructura generada por \bar{a} en \mathbf{A} .

Definición 4. Dos fórmulas $\alpha(\bar{x})$ y $\beta(\bar{x})$ se dicen *equivalentes* sobre una familia de estructuras \mathcal{K} si para cada $\mathbf{A} \in \mathcal{K}$ y cada \bar{a} de A vale que

$$\mathbf{A} \models \alpha[\bar{a}] \iff \mathbf{A} \models \beta[\bar{a}].$$

La equivalencia de formulas es un concepto fundamental para nuestro desarrollo ya que permite probar el siguiente teorema. Este teorema permite convertir conjuntos infinitos de formulas, en conjuntos finitos de representantes de cada clase de equivalencia entre ellas. Esto será fundamental a la hora de construir formulas que definan relaciones durante nuestro trabajo.

Teorema 5 (Modulo equivalencia sobre una estructura finita, la cantidad de fórmulas en x_1, \dots, x_n es finita). *Sea \mathcal{K} una clase finita de estructuras finitas, y sean x_1, \dots, x_n variables.*

Hay un conjunto finito de fórmulas $\Sigma(\bar{x})$ tal que para toda fórmula $\varphi(\bar{x})$ hay $\sigma(\bar{x}) \in \Sigma(\bar{x})$ tal que $\varphi(\bar{x})$ y $\sigma(\bar{x})$ son equivalentes sobre \mathcal{K} .

Demostración. Sea $\mathcal{K} = \{\mathbf{A}_1, \dots, \mathbf{A}_m\}$. Veamos primero que las fórmulas son finitas modulo equivalencia. Sea $\varphi(\bar{x})$ defino

$$T_{\varphi\mathcal{K}} = \{\bar{a} \in A_1^n \mid \mathbf{A}_1 \models \varphi[\bar{a}]\} \times \cdots \times \{\bar{a} \in A_m^n \mid \mathbf{A}_m \models \varphi[\bar{a}]\}.$$

Ahora supongamos φ equivalente ψ en \mathcal{K} , entonces

$$(\bar{a}_1, \dots, \bar{a}_m) \in T_{\varphi\mathcal{K}} \Leftrightarrow (\bar{a}_1, \dots, \bar{a}_m) \in T_{\psi\mathcal{K}}$$

$$\text{para cada } (\bar{a}_1, \dots, \bar{a}_m) \in A_1^n \times \dots \times A_m^n$$

lo que significa que $T_{\varphi\mathcal{K}} = T_{\psi\mathcal{K}}$. Entonces basta con contar los subconjuntos de $A_1^n \times \dots \times A_m^n$. Como $|A_1^n \times \dots \times A_m^n| = |A_1|^n \dots |A_m|^n$ y cada A_i era finito, es claro que $A_1^n \times \dots \times A_m^n$ es finito. Por lo tanto $\mathcal{P}(A^n)$ también lo es.

Ahora veamos que existe $\Sigma(\bar{x})$. Sean $T_1, \dots, T_k \subseteq A_1^n \times \dots \times A_m^n$ tales que existe $\varphi_i(\bar{x})$ tal que para cada $\mathbf{A} \in \mathcal{K}$, $\mathbf{A} \models \varphi_i[T_i]$ y sea $\psi(\bar{x})$, tomo $T_{\psi\mathcal{K}}$ como T_1, \dots, T_k es la sucesión de todos los subconjuntos que se definen con una fórmula, hay un j tal que $T_j = T_{\psi\mathcal{K}}$. Entonces, para cada $\mathbf{A} \in \mathcal{K}$, $\mathbf{A} \models \psi[\bar{a}] \Leftrightarrow \mathbf{A} \models \varphi_j[\bar{a}]$. \square

El siguiente lema establece que los embeddings preservan las formulas abiertas.

Lema 6 (Los embeddings preservan fórmulas abiertas). *Sean \mathbf{A}, \mathbf{B} estructuras y $\gamma : \mathbf{A} \rightarrow \mathbf{B}$ una función. Son equivalentes:*

1. γ es un embedding de A en B .
2. Para toda fórmula abierta $\varphi(\bar{x})$ y para cada \bar{a} de A vale que:

$$\mathbf{A} \models \varphi[\bar{a}] \Leftrightarrow \mathbf{B} \models \varphi[\gamma(\bar{a})].$$

Demostración. $1 \Rightarrow 2$)

Sea γ un embedding de A en B , sea $\varphi(x_1, \dots, x_n)$ una fórmula abierta y $\bar{a} \in A^n$, el caso base sale directo ya que los homomorfismos preservan términos. Veamos los casos inductivos:

Sea $\varphi(\bar{x}) = \neg\varphi_1(\bar{x})$:

$$\mathbf{A} \models \neg\varphi_1[\bar{a}] \text{ sii } \mathbf{A} \not\models \varphi_1[\bar{a}] \text{ sii } \mathbf{B} \not\models \varphi_1[\gamma(\bar{a})] \text{ sii } \mathbf{B} \models \neg\varphi_1[\gamma(\bar{a})]$$

Sea $\varphi(\bar{x}) = (\varphi_1 \eta \varphi_2)(\bar{x})$:

$$\begin{aligned} \mathbf{A} \models (\varphi_1 \eta \varphi_2)[\bar{a}] \text{ sii } \mathbf{A} \models \varphi_1[\bar{a}] \text{''} \eta \text{''} \mathbf{A} \models \varphi_2[\bar{a}] \\ \text{sii } \mathbf{A} \models \varphi_1[\gamma(\bar{a})] \text{''} \eta \text{''} \mathbf{B} \models \varphi_2[\gamma(\bar{a})] \text{ sii } \mathbf{B} \models (\varphi_1 \eta \varphi_2)[\gamma(\bar{a})] \end{aligned}$$

$2 \Rightarrow 1$)

Supongamos que para toda fórmula abierta $\varphi = \varphi(\bar{x})$ y cada $\bar{a} \in A^m$ vale que:

$$\mathbf{A} \models \varphi[\bar{a}] \Leftrightarrow \mathbf{B} \models \varphi[\gamma(\bar{a})]$$

Es rutina ver que γ es homomorfismo.

■ Veamos que γ es inyectiva:

$$\begin{aligned} \gamma(a) &= \gamma(a') \\ \mathbf{B} &\models (x_1 \equiv x_2)[\gamma(a), \gamma(a')] \\ \mathbf{A} &\models (x_1 \equiv x_2)[a, a'] \\ a &= a' \end{aligned}$$

- Veamos que γ es embedding:

Sea $r \in \mathcal{R}_n$

$$\begin{aligned} (\gamma(a_1), \dots, \gamma(a_n)) &\in r^{\mathbf{B}} \\ \mathbf{B} &\models r(x_1, \dots, x_n)[\gamma(a_1), \dots, \gamma(a_n)] \\ \mathbf{A} &\models r(x_1, \dots, x_n)[a_1, \dots, a_n] \\ (a_1, \dots, a_n) &\in r^{\mathbf{A}} \end{aligned}$$

□

El siguiente lema establece que las subestructuras preservan formulas abiertas.

Lema 7. Si \mathbf{A} es una subestructura de \mathbf{B} y $\varphi(\bar{x})$ es una fórmula abierta, entonces para cada $\bar{a} \in A^n$ vale que

$$\mathbf{A} \models \varphi[\bar{a}] \Leftrightarrow \mathbf{B} \models \varphi[\bar{a}]$$

Demostración. Sea \mathbf{A} una subestructura de \mathbf{B} y $\varphi(\bar{x})$ una fórmula abierta.

Como \mathbf{A} es subestructura es cerrada sobre r, f y $c^{\mathbf{B}} \in A$

Sea $\gamma : \mathbf{A} \rightarrow \mathbf{B}$ tal que para todo $x \in A$, $\gamma(x) = x$. Como $\mathbf{A} \leq \mathbf{B}$, es directo que γ es un embedding. Entonces por el Lema 6, $\mathbf{A} \models \varphi[\bar{a}] \Leftrightarrow \mathbf{B} \models \varphi[\gamma(\bar{a})]$

Finalmente, por definición de γ :

$$\mathbf{A} \models \varphi[\bar{a}] \Leftrightarrow \mathbf{B} \models \varphi[\bar{a}]$$

□

Ahora definimos el diagrama abierto, que luego sera utilizado en la construcción de formulas abiertas que definan relaciones.

Definición 8. Sea \mathbf{A} una estructura y sean $a_1, \dots, a_n \in A$. Definimos el diagrama abierto para a_1, \dots, a_n en \mathbf{A} como:

$$\Delta_{\mathbf{A}, \bar{a}}(x_1, \dots, x_n) := \{\alpha \mid \alpha \in \text{Op}(\mathcal{L}) \text{ y } \mathbf{A} \models \alpha[\bar{a}]\}$$

El siguiente lema establece una relación muy interesante entre elementos de dos estructuras al mostrar que si dos grupos de elementos tienen las mismas propiedades abiertas (i.e. el diagrama abierto) significa que generan subestructuras isomorfas.

Lema 9. Sea \mathbf{A} una estructura y $b_1, \dots, b_n \in B$, son equivalentes:

1. $\mathbf{B} \models \Delta_{\mathbf{A}, \bar{a}}[\bar{b}]$
2. Hay un isomorfismo γ de $\langle \bar{a} \rangle^{\mathbf{A}}$ en $\langle \bar{b} \rangle^{\mathbf{B}}$ tal que $\gamma(\bar{a}) = \bar{b}$

Demostración. Veamos $1 \Rightarrow 2$:

Supongamos que $\mathbf{B} \models \Delta_{\mathbf{A}, \bar{a}}[\bar{b}]$

Si α fórmula abierta y $\mathbf{A} \models \alpha[\bar{a}]$ entonces $\mathbf{B} \models \alpha[\bar{b}]$

Tomo

$$\langle \bar{a} \rangle^{\mathbf{A}} = \{t^{\mathbf{A}}[a_1, \dots, a_n] : t \in T^r, a_1, \dots, a_n \in \bar{a}\}$$

y

$$\langle \bar{b} \rangle^{\mathbf{A}} = \{t^{\mathbf{A}}[b_1, \dots, b_n] : t \in T^\tau, b_1, \dots, b_n \in \bar{b}\}.$$

Defino

$$\begin{aligned} \gamma & : A' \rightarrow B' \\ \gamma & (t^{\mathbf{A}}[a_1, \dots, a_n]) = t^{\mathbf{B}}[b_1, \dots, b_n] \end{aligned}$$

Es claro que γ es un homomorfismo.

- Veamos que γ es inyectivo

Sean $a'_1, a'_2 \in A'$ tales que $a'_1 \neq a'_2$

$$\begin{aligned} a'_1 & \neq a'_2 \\ t_1^{\mathbf{A}}[\bar{a}] & \neq t_2^{\mathbf{A}}[\bar{a}] \\ \mathbf{A} & \models \neg(t_1 \equiv t_2)[\bar{a}] \\ \mathbf{B} & \models \neg(t_1 \equiv t_2)[\bar{b}] \\ t_1^{\mathbf{B}}[\bar{b}] & \neq t_2^{\mathbf{B}}[\bar{b}] \\ \gamma(t_1^{\mathbf{A}}[\bar{a}]) & \neq \gamma(t_2^{\mathbf{A}}[\bar{a}]) \end{aligned}$$

- Veamos que γ es sobreyectivo

Sea $b' \in B'$

$$\begin{aligned} b' & \in B' \\ t^{\mathbf{B}}[\bar{b}] & \in B' \end{aligned}$$

Tomo $t^{\mathbf{A}}[\bar{a}] = a'$

$$\gamma(t^{\mathbf{A}}[\bar{a}]) = t^{\mathbf{B}}[\bar{b}]$$

Por lo tanto γ es un isomorfismo y

$$\gamma(a_j) = \gamma(x_j^{\mathbf{A}}[a_1, \dots, a_n]) = x_j^{\mathbf{B}}[b_1, \dots, b_n] = b_j.$$

Ahora veamos $2 \Rightarrow 1$:

Supongamos que hay un isomorfismo γ de $\langle \bar{a} \rangle^{\mathbf{A}}$ en $\langle \bar{b} \rangle^{\mathbf{B}}$ tal que $\gamma(\bar{a}) = \bar{b}$

Entonces, es claro que γ es un embedding de $\langle \bar{a} \rangle^{\mathbf{A}}$ en $\langle \bar{b} \rangle^{\mathbf{B}}$.

Por el Lema 6 para toda fórmula abierta $\varphi(\bar{x})$ y cada $\bar{a} \in A'$

$$\langle \bar{a} \rangle^{\mathbf{A}} \models \varphi[\bar{a}] \Leftrightarrow \langle \bar{b} \rangle^{\mathbf{B}} \models \varphi[\gamma(\bar{a})]$$

Por el Lema 7 como $\langle \bar{a} \rangle^{\mathbf{A}}$ es subestructura de \mathbf{A} , $\langle \bar{b} \rangle^{\mathbf{B}}$ lo es de \mathbf{B} y φ es abierta

$$\mathbf{A} \models \varphi[\bar{a}] \Leftrightarrow \mathbf{B} \models \varphi[\gamma(\bar{a})]$$

$$\mathbf{A} \models \varphi[\bar{a}] \Leftrightarrow \mathbf{B} \models \varphi[\bar{b}]$$

Por lo tanto si α es abierta y $\mathbf{A} \models \alpha[\bar{a}]$ entonces $\mathbf{B} \models \alpha[\bar{b}]$

Finalmente, $\mathbf{B} \models \Delta_{\mathbf{A}\bar{a}}[\bar{b}]$ \square

En el lema que sigue se establece que los homomorfismos preservan formulas atómicas, y también formulas abiertas positivas.

Lema 10. Sean \mathbf{A}, \mathbf{B} estructuras y $h : A \rightarrow B$ una función, son equivalentes:

1. h es un homomorfismo de \mathbf{A} en \mathbf{B} .
2. Para toda fórmula atómica $\varphi = \varphi(\bar{x})$ y para cada \bar{a} de A vale que

$$\mathbf{A} \models \varphi[\bar{a}] \implies \mathbf{B} \models \varphi[h(\bar{a})].$$

3. Para toda fórmula abierta positiva $\varphi = \varphi(\bar{x})$ y para cada \bar{a} de A vale que

$$\mathbf{A} \models \varphi[\bar{a}] \implies \mathbf{B} \models \varphi[h(\bar{a})].$$

Demostración. $1 \Rightarrow 2$) Directo, ya que los homomorfismos preservan términos.

$2 \Rightarrow 1$) Supongamos $\mathbf{A} \models \varphi[\bar{a}] \Rightarrow \mathbf{B} \models \varphi[h(\bar{a})]$ con φ atómica. Veamos que h es homomorfismo.

Sea $c \in \mathcal{C}$

$$\mathbf{A} \models (c \equiv x_1) [c^{\mathbf{A}}] \Rightarrow \mathbf{B} \models (c \equiv x_1) [h(c^{\mathbf{A}})] \Leftrightarrow c^{\mathbf{B}} = h(c^{\mathbf{A}})$$

Sea $f \in \mathcal{F}$

$$\begin{aligned} \mathbf{A} &\models (f(x_1, \dots, x_n) \equiv x_{n+1}) [a_1, \dots, a_n, f^{\mathbf{A}}(\bar{a})] \\ \mathbf{B} &\models (f(x_1, \dots, x_n) \equiv x_{n+1}) [h(a_1), \dots, h(a_n), h(f^{\mathbf{A}}(\bar{a}))] \\ f^{\mathbf{B}}(h(\bar{a})) &= h(f^{\mathbf{A}}(\bar{a})) \end{aligned}$$

Sea $r \in \mathcal{R}$

$$\bar{a} \in r^{\mathbf{A}} \Rightarrow \mathbf{A} \models r(\bar{x}) [\bar{a}] \Rightarrow \mathbf{B} \models r(\bar{x}) [h(\bar{a})] \Rightarrow h(\bar{a}) \in r^{\mathbf{B}}$$

$2 \Rightarrow 3$) Rutina.

$3 \Rightarrow 2$) Directo ya que toda atómica es abierta positiva. \square

Ahora definimos el diagrama atómico positivo. Este se utilizara para la construcción de formulas positivas.

Definición 11. Sea \mathbf{A} una estructura y sean $a_1, \dots, a_n \in A$. Definimos el diagrama atómico positivo de \bar{a} en \mathbf{A} como

$$\Delta_{\mathbf{A}, \bar{a}}^+(x_1, \dots, x_n) := \{\alpha \mid \alpha \in \text{At}(\mathcal{L}) \text{ y } \mathbf{A} \models \alpha[\bar{a}]\}.$$

De manera análoga al Lema 9, se establece que si dos grupos de elementos en dos estructuras satisfacen las mismas propiedades atómicas positivas, hay un homomorfismo entre las subestructuras generadas entre ellos.

Teorema 12. Sean \mathbf{B} una estructura y $b_1, \dots, b_n \in B$, son equivalentes:

$$1. \mathbf{B} \models \Delta_{\mathbf{A}, \bar{a}}^+ [\bar{b}].$$

$$2. \text{ Hay un homomorfismo } h \text{ de } \langle \bar{a} \rangle^{\mathbf{A}} \text{ en } \langle \bar{b} \rangle^{\mathbf{B}} \text{ tal que } h(\bar{a}) = \bar{b}.$$

Demostración. $1 \Rightarrow 2$) Defino $h : \langle \bar{a} \rangle^{\mathbf{A}} \rightarrow \langle \bar{b} \rangle^{\mathbf{B}}$, como

$$h(t^{\mathbf{A}}[\bar{a}]) = t^{\mathbf{B}}[\bar{b}]$$

El cual es se ve fácilmente que es un homomorfismo que cumple $h(\bar{a}) = \bar{b}$.

$2 \Rightarrow 1$) Supongamos que hay un homomorfismo h de $\langle \bar{a} \rangle^{\mathbf{A}}$ en $\langle \bar{b} \rangle^{\mathbf{B}}$ tal que $h(\bar{a}) = \bar{b}$, entonces como $\mathbf{A} \models \Delta_{\mathbf{A}, \bar{a}}^+ [\bar{a}]$, por el Lema 10 $\mathbf{B} \models \Delta_{\mathbf{A}, \bar{a}}^+ [h(\bar{a})]$, entonces $\mathbf{B} \models \Delta_{\mathbf{A}, \bar{a}}^+ [\bar{b}]$ \square

2.2. Definibilidad de primer orden

Este es el tipo de definibilidad mas débil. Los morfismos en cuestión son los isomorfismos entre estructuras de \mathcal{K} .

Teorema 13. *Sean $\mathcal{L} \subseteq \mathcal{L}'$ lenguajes de primer orden, sea $R \in \mathcal{L}' - \mathcal{L}$ un símbolo de relación n -ario. Para una clase finita \mathcal{K} de \mathcal{L}' -estructuras, los siguientes son equivalentes:*

1. Hay una fórmula en $\text{Fo}(\mathcal{L})$ que define R en \mathcal{K} .
2. Para todas $\mathbf{A}, \mathbf{B} \in \mathcal{K}$, todo isomorfismo $\gamma : \mathbf{A}_{\mathcal{L}} \rightarrow \mathbf{B}_{\mathcal{L}}$ preserva R .

Demostración. $1 \Rightarrow 2$) Sea φ una \mathcal{L} -fórmula que define a R en \mathcal{K} , y sea $\gamma : \mathbf{A} \rightarrow \mathbf{B}$ un isomorfismo. Supongamos $\mathbf{A} \models \varphi[\bar{a}]$ entonces $\mathbf{B} \models \varphi[\gamma(\bar{a})]$ y como φ define a R en \mathcal{K} , entonces $\gamma(\bar{a}) \in R^{\mathbf{B}}$.

$2 \Rightarrow 1$) Para cada $\mathbf{A} \in \mathcal{K}$, cada $\bar{a} \in R^{\mathbf{A}}$, tomo \bar{a}' tal que $\langle \bar{a}, \bar{a}' \rangle^{\mathbf{A}} = \mathbf{A}$. Ahora tomo $\delta_{\mathbf{A}, (\bar{a}, \bar{a}')}(\bar{x}, \bar{y}) = \bigwedge_{\alpha \in \Delta_{(\bar{a}, \bar{a}'), \mathbf{A}}} \alpha(\bar{x}, \bar{y})$, que es fórmula por Teorema 5. Ahora tomo

$$\varphi = \bigvee_{\mathbf{A} \in \mathcal{K}} \left(\left(\bigvee_{\bar{a} \in R^{\mathbf{A}}} (\exists \bar{y} \delta_{\mathbf{A}, (\bar{a}, \bar{a}')}(\bar{x}, \bar{y})) \right) \wedge \left(\forall x_1, \dots, x_{|\mathbf{A}|+1} \bigvee_{i \neq j} x_i = x_j \right) \right)$$

Sea $(b_1, \dots, b_n) \in R^{\mathbf{B}}$. Entonces como

$$\varphi = \dots \vee \left((\exists \bar{y} \delta_{\mathbf{B}, (\bar{b}, \bar{b}')}(\bar{x}, \bar{y}) \vee \dots) \wedge \left(\forall x_1, \dots, x_{|\mathbf{B}|+1} \bigvee_{i \neq j} x_i = x_j \right) \right) \vee \dots,$$

y como $\mathbf{B} \models \exists \bar{y} \delta_{\mathbf{B}, (\bar{b}, \bar{b}')}(\bar{x}, \bar{y}) [\bar{b}, \bar{b}']$ y $\mathbf{B} \models \forall x_1, \dots, x_{|\mathbf{B}|+1} \bigvee_{i \neq j} x_i = x_j [\bar{b}, \bar{b}']$ entonces $\mathbf{B} \models \varphi[\bar{b}]$.

Sea $\mathbf{B} \models \varphi[b_1, \dots, b_n]$. Entonces hay algún $\mathbf{A} \in \mathcal{K}$, y algún $\bar{a} \in R^{\mathbf{A}}$ tales que

$$\mathbf{B} \models (\exists \bar{y} \delta_{\mathbf{A}, (\bar{a}, \bar{a}')}(\bar{x}, \bar{y})) \wedge \left(\forall x_1, \dots, x_{|\mathbf{A}|+1} \bigvee_{i \neq j} x_i = x_j \right) [\bar{b}].$$

Entonces hay un \bar{z} en \mathbf{B} tal que

$$\mathbf{B} \models \delta_{\mathbf{A},(\bar{a},\bar{a}')}(\bar{x},\bar{y})[\bar{b},\bar{z}].$$

Como $\mathbf{B} \models \Delta_{(\bar{a},\bar{a}'),\mathbf{A}}[\bar{b},\bar{z}]$, por Lema 9 hay $\gamma : \langle \bar{a}, \bar{a}' \rangle^{\mathbf{A}} \rightarrow \langle \bar{b}, \bar{b}' \rangle^{\mathbf{B}}$ isomorfismo tal que $(\gamma(\bar{a}), \gamma(\bar{a}')) = (\bar{b}, \bar{z})$ con $\langle \bar{a}, \bar{a}' \rangle^{\mathbf{A}} = \mathbf{A}$ y $\langle \bar{b}, \bar{b}' \rangle^{\mathbf{B}} \leq \mathbf{B}$, lo que implica que $|\mathbf{A}| \leq |\mathbf{B}|$. Pero como

$$\mathbf{B} \models \forall x_1, \dots, x_{|\mathbf{A}|+1} \bigvee_{i \neq j} x_i = x_j [b_1, \dots, b_n]$$

entonces $|\mathbf{B}| \leq |\mathbf{A}|$, luego $|\mathbf{B}| = |\mathbf{A}|$, por lo que es claro que $\langle \bar{b}, \bar{b}' \rangle^{\mathbf{B}} = \mathbf{B}$. Como $\gamma : \mathbf{A} \rightarrow \mathbf{B}$ es un isomorfismo tal que $(\gamma(\bar{a}), \gamma(\bar{a}')) = (\bar{b}, \bar{z})$ y sabemos que los isomorfismos entre estructuras de \mathcal{K} , preservan R , como $\bar{a} \in R^{\mathbf{A}}$, $\gamma(\bar{a}) = \bar{b} \in R^{\mathbf{B}}$. \square

2.3. Definibilidad por fórmulas abiertas

Ahora estableceremos la noción semántica para la definibilidad abierta.

En el siguiente lema, construimos una formula equivalente al diagrama abierto para un conjunto de elementos en una estructura, que utilizaremos en el teorema de definibilidad abierta.

Lema 14. Sean \mathbf{A}, \mathbf{B} estructuras finitas y $a_1, \dots, a_n \in A$, entonces existe una fórmula abierta $\varphi(\bar{x})$ tal que para todo $\bar{b} \in B^n$ son equivalentes:

1. $\mathbf{B} \models \varphi[\bar{b}]$
2. Hay un isomorfismo γ de $\langle \bar{a} \rangle^{\mathbf{A}}$ en $\langle \bar{b} \rangle^{\mathbf{B}}$ tal que $\gamma(\bar{a}) = \bar{b}$

Demostración. Tomo

$$\varphi = \bigwedge_{\alpha \in \Delta_{\mathbf{A}\bar{a}}} \alpha(\bar{x})$$

que es una fórmula, por el Teorema 5.

Es claro que $\mathbf{B} \models \bigwedge_{\alpha \in \Delta_{\mathbf{A}\bar{a}}} \alpha(\bar{x})[\bar{b}] \Leftrightarrow \mathbf{B} \models \Delta_{\mathbf{A}\bar{a}}[\bar{b}]$

Luego por Lema 9, hay isomorfismo $\gamma : \langle \bar{a} \rangle^{\mathbf{A}} \rightarrow \langle \bar{b} \rangle^{\mathbf{B}}$ tal que $\gamma(\bar{a}) = \bar{b}$ \square

Como establece el siguiente teorema, la definibilidad abierta depende de la preservación en los isomorfismos entre subestructuras de las estructuras de \mathcal{K} . Notar que esta condición incluye a los isomorfismos entre estructuras de \mathcal{K} , chequeados para el Teorema 13.

Teorema 15. Sean $\mathcal{L} \subseteq \mathcal{L}'$ lenguajes de primer orden, sea $R \in \mathcal{L}' - \mathcal{L}$ un símbolo de relación n -ario. Para una clase finita \mathcal{K} de \mathcal{L}' -estructuras, los siguientes son equivalentes:

1. Hay una fórmula en $\text{Op}(\mathcal{L})$ que define R en \mathcal{K} .
2. Para todas $\mathbf{A}, \mathbf{B} \in \mathcal{K}$, todas $\mathbf{A}_0 \leq \mathbf{A}_{\mathcal{L}}, \mathbf{B}_0 \leq \mathbf{B}_{\mathcal{L}}$, todo isomorfismo $\sigma : \mathbf{A}_0 \rightarrow \mathbf{B}_0$ preserva R .

Demostración. $1 \Rightarrow 2$) Sea $\varphi(\bar{x})$ la fórmula que define R en \mathcal{K} , sean $\mathbf{A}, \mathbf{B} \in \mathcal{K}$ y sean $\mathbf{A}_0 \leq \mathbf{A}_{\mathcal{L}}$ y $\mathbf{B}_0 \leq \mathbf{B}_{\mathcal{L}}$ tales que $\sigma : \mathbf{A}_0 \rightarrow \mathbf{B}_0$ sea un isomorfismo. Sea $\bar{a} \in R^{\mathbf{A}_0} \cap A_0$ veamos que $\sigma(\bar{a}) \in R^{\mathbf{B}_0}$. Como $\bar{a} \in R^{\mathbf{A}_0} \Rightarrow \bar{a} \in R^{\mathbf{A}} \Leftrightarrow \mathbf{A} \models \varphi[\bar{a}]$, por Lema 7, $A_0 \models \varphi[\bar{a}]$ y como \mathbf{A}_0 y \mathbf{B}_0 son isomorfos por σ , $\mathbf{B}_0 \models \varphi[\sigma(\bar{a})]$. Nuevamente por Lema 7 $\mathbf{B} \models \varphi[\sigma(\bar{a})]$ y como por hipótesis φ define a R en \mathcal{K} , $\sigma(\bar{a}) \in R^{\mathbf{B}}$.

$2 \Rightarrow 1$) Sea

$$\varphi = \bigvee_{\mathbf{A} \in \mathcal{K}} \left(\bigvee_{\bar{a} \in R^{\mathbf{A}}} \left(\bigwedge_{\alpha \in \Delta_{\mathbf{A}\bar{a}}} \alpha(\bar{x}) \right) \right),$$

la cual es fórmula por el Teorema 5. Veamos que para cada $\mathbf{B} \in \mathcal{K}$, $\mathbf{B} \models \varphi[\bar{b}]$ sii $\bar{b} \in R^{\mathbf{B}}$. Sea $\mathbf{B} \in \mathcal{K}$:

\Rightarrow) Supongamos $\mathbf{B} \models \varphi[\bar{b}]$, entonces en particular para algún $\mathbf{A} \in \mathcal{K}$ y $\bar{a} \in R^{\mathbf{A}}$,

$$\mathbf{B} \models \bigwedge_{\alpha \in \Delta_{\mathbf{A}\bar{a}}} \alpha(\bar{x})[\bar{b}].$$

Por Lema 14, hay un isomorfismo $\gamma : \langle \bar{a} \rangle^{\mathbf{A}} \rightarrow \langle \bar{b} \rangle^{\mathbf{B}}$ tal que $\gamma(\bar{a}) = \bar{b}$. Entonces por hipótesis γ preserva R . Finalmente como $\bar{a} \in R^{\mathbf{A}}$, entonces $\gamma(\bar{a}) \in R^{\mathbf{B}}$ o sea que $\bar{b} \in R^{\mathbf{B}}$.

\Leftarrow) Supongamos $\bar{b} \in R^{\mathbf{B}}$. Entonces

$$\varphi = \bigvee_{\mathbf{A} \in \mathcal{K}} \left(\bigvee_{\bar{a} \in R^{\mathbf{A}}} \left(\bigwedge_{\alpha \in \Delta_{\mathbf{A}\bar{a}}} \alpha(\bar{x}) \right) \right) = \dots \vee \dots \vee \left(\bigwedge_{\alpha \in \Delta_{\mathbf{B}\bar{b}}} \alpha(\bar{x}) \right),$$

evidentemente $\mathbf{B} \models \Delta_{\mathbf{B}\bar{b}}[\bar{b}]$, entonces $\mathbf{B} \models \varphi[\bar{b}]$. \square

2.4. Definibilidad por fórmulas abiertas positivas

Ahora estableceremos la noción semántica para la definibilidad abierta positiva.

En el siguiente lema, construimos una formula equivalente al diagrama atómico positivo para un conjunto de elementos en una estructura, que utilizaremos en el teorema de definibilidad abierta positiva.

Lema 16. *Sean \mathbf{A}, \mathbf{B} estructuras finitas y $a_1, \dots, a_n \in A$, entonces existe una fórmula abierta $\varphi(\bar{x})$ tal que para todo $\bar{b} \in B^n$ son equivalentes:*

$$1. \mathbf{B} \models \varphi[\bar{b}]$$

$$2. \text{ Hay un homomorfismo } h \text{ de } \langle \bar{a} \rangle^{\mathbf{A}} \text{ en } \langle \bar{b} \rangle^{\mathbf{B}} \text{ tal que } h(\bar{a}) = \bar{b}$$

Demostración. Tomo $\varphi = \bigwedge_{\alpha \in \Delta_{\mathbf{A}\bar{a}}}^+ \alpha(\bar{x})$ que es una fórmula, por el Teorema 5.

$$\text{Es claro que } \mathbf{B} \models \varphi[\bar{b}] \Leftrightarrow \mathbf{B} \models \Delta_{\mathbf{A}\bar{a}}^+[\bar{b}]$$

Entonces por Lema 12, hay un homomorfismo h de $\langle \bar{a} \rangle^{\mathbf{A}}$ en $\langle \bar{b} \rangle^{\mathbf{B}}$ tal que $h(\bar{a}) = \bar{b}$ \square

El siguiente teorema establece que la definibilidad abierta positiva depende de la preservación en los homomorfismos entre subestructuras de las estructuras de \mathcal{K} . Notar que esta condición incluye a los isomorfismos entre subestructuras de estructuras de \mathcal{K} , chequeados para el Teorema 15.

Teorema 17. Sean $\mathcal{L} \subseteq \mathcal{L}'$ lenguajes de primer orden, sea $R \in \mathcal{L}' - \mathcal{L}$ un símbolo de relación n -ario. Para una clase finita \mathcal{K} de \mathcal{L}' -estructuras, los siguientes son equivalentes:

1. Hay una fórmula en $\text{Op}^+(\mathcal{L})$ que define R en \mathcal{K} .
2. Para todas $\mathbf{A}, \mathbf{B} \in \mathcal{K}$, todas $\mathbf{A}_0 \leq \mathbf{A}_{\mathcal{L}}, \mathbf{B}_0 \leq \mathbf{B}_{\mathcal{L}}$, todo homomorfismo $h : \mathbf{A}_0 \rightarrow \mathbf{B}_0$ preserva R .

Demostración. Veamos $1 \Rightarrow 2$:

Sea $\varphi(\bar{x})$ la fórmula abierta positiva que define R en \mathcal{K} , sean $\mathbf{A}, \mathbf{B} \in \mathcal{K}$ y sean $\mathbf{A}_0 \leq \mathbf{A}_{\mathcal{L}}$ y $\mathbf{B}_0 \leq \mathbf{B}_{\mathcal{L}}$ tales que $h : \mathbf{A}_0 \rightarrow \mathbf{B}_0$ sea un homomorfismo. Sea $\bar{a} \in R^{\mathbf{A}_0} \cap A_0$ veamos que $h(\bar{a}) \in R^{\mathbf{B}_0}$:

$\bar{a} \in R^{\mathbf{A}_0} \Rightarrow \bar{a} \in R^{\mathbf{A}} \Leftrightarrow \mathbf{A} \models \varphi[\bar{a}]$, por Lema 7, $\mathbf{A}_0 \models \varphi[\bar{a}]$ y como h es un homomorfismo de \mathbf{A}_0 en \mathbf{B}_0 y φ es abierta positiva por Lema 10, $\mathbf{B}_0 \models \varphi[h(\bar{a})]$. Nuevamente por Lema 7 $\mathbf{B} \models \varphi[h(\bar{a})]$ y como por hipótesis φ define a R en \mathcal{K} , $h(\bar{a}) \in R^{\mathbf{B}}$.

Veamos $2 \Rightarrow 1$:

Sea

$$\varphi = \bigvee_{\mathbf{A} \in \mathcal{K}} \left(\bigvee_{\bar{a} \in R^{\mathbf{A}}} \left(\bigwedge_{\alpha \in \Delta_{\mathbf{A}\bar{a}}^+} \alpha(\bar{x}) \right) \right)$$

, la cual es fórmula por el Teorema 5. Sea $\mathbf{B} \in \mathcal{K}$:

\Rightarrow Supongamos $\mathbf{B} \models \varphi[\bar{b}]$, entonces en particular para algún $\mathbf{A} \in \mathcal{K}$ y $\bar{a} \in R^{\mathbf{A}}, \mathbf{B} \models \Delta_{\mathbf{A}\bar{a}}^+[\bar{b}]$. Por Lema 16, hay un homomorfismo $h : \langle \bar{a} \rangle^{\mathbf{A}} \rightarrow \langle \bar{b} \rangle^{\mathbf{B}}$ tal que $h(\bar{a}) = \bar{b}$. Entonces por hipótesis h preserva R . Finalmente como $\bar{a} \in R^{\mathbf{A}}$, entonces $h(\bar{a}) \in R^{\mathbf{B}}$ o sea que $\bar{b} \in R^{\mathbf{B}}$.

\Leftarrow Supongamos $\bar{b} \in R^{\mathbf{B}}$. Entonces $\varphi = \dots \vee \dots \vee \left(\bigwedge_{\alpha \in \Delta_{\mathbf{B}\bar{b}}^+} \alpha(\bar{x}) \right)$, evidentemente $\mathbf{B} \models \Delta_{\mathbf{B}\bar{b}}^+[\bar{b}]$, entonces $\mathbf{B} \models \varphi[\bar{b}]$. \square

2.5. Definibilidad por fórmulas existenciales

Ahora estableceremos la noción semántica para la definibilidad existencial.

En el siguiente lema, construimos una formula equivalente al diagrama abierto para un conjunto de elementos tal que sea capaz de generar la estructura completa.

Lema 18. Sea \mathbf{A} una estructura finita y $a_1, \dots, a_n \in A$. Entonces hay una fórmula existencial $\varphi(x_1, \dots, x_n)$ tal que para toda estructura \mathbf{B} y para todo $\bar{b} \in B^n$ los siguientes son equivalentes:

1. $\mathbf{B} \models \varphi[\bar{b}]$.
2. Hay un embedding $\gamma : \mathbf{A} \rightarrow \mathbf{B}$ tal que $\gamma(\bar{a}) = \bar{b}$

Demostración. Sean a'_1, \dots, a'_m tales que $\langle a_1, \dots, a_n, a'_1, \dots, a'_m \rangle^{\mathbf{A}} = \mathbf{A}$ y sea

$$\varphi = \exists y_1, \dots, y_m \left(\bigwedge_{\alpha \in \Delta_{(\bar{a}, \bar{a}')}} \alpha(\bar{x}, \bar{y}) \right)$$

que es fórmula por el Teorema 5.

Supongamos $\mathbf{B} \models \varphi [\bar{b}]$, entonces existen $b'_1, \dots, b'_m \in B$ tales que

$$\mathbf{B} \models \bigwedge_{\alpha \in \Delta_{(\bar{a}, \bar{a}'), \mathbf{A}}} \alpha(\bar{x}, \bar{y}) [\bar{b}, \bar{b}']$$

por Lema 9 hay isomorfismo $\gamma : \langle \bar{a}, \bar{a}' \rangle^{\mathbf{A}} \rightarrow \langle \bar{b}, \bar{b}' \rangle^{\mathbf{B}}$ tal que $\gamma(\bar{a}, \bar{a}') = (\bar{b}, \bar{b}')$, y como $\langle \bar{a}, \bar{a}' \rangle^{\mathbf{A}} = \mathbf{A}$ y $\langle \bar{b}, \bar{b}' \rangle^{\mathbf{B}} \leq \mathbf{B}$, γ es un embedding de \mathbf{A} en \mathbf{B} tal que $\gamma(\bar{a}) = \bar{b}$.

Supongamos que hay un embedding $\gamma : \mathbf{A} \rightarrow \mathbf{B}$ tal que $\gamma(\bar{a}) = \bar{b}$, por lo tanto $\gamma : \mathbf{A} \rightarrow \mathbf{S} \leq \mathbf{B}$ es un isomorfismo. Como $\mathbf{A} \models \varphi[\bar{a}]$, por ser γ un isomorfismo $\mathbf{S} \models \varphi[\gamma(\bar{a})]$ y entonces $\mathbf{S} \models \varphi[\bar{b}]$. Por lo tanto existe \bar{b}' en $S \subseteq B$ tal que $\mathbf{S} \models \bigwedge_{\alpha \in \Delta_{(\bar{a}, \bar{a}'), \mathbf{A}}} \alpha(\bar{x}, \bar{y}) [\bar{b}, \bar{b}']$. Como $\bigwedge_{\alpha \in \Delta_{(\bar{a}, \bar{a}'), \mathbf{A}}} \alpha(\bar{x}, \bar{y})$ es una fórmula abierta, por Lema 7, $\mathbf{B} \models \bigwedge_{\alpha \in \Delta_{(\bar{a}, \bar{a}'), \mathbf{A}}} \alpha(\bar{x}, \bar{y}) [\bar{b}, \bar{b}']$, entonces $\mathbf{B} \models \varphi[\bar{b}]$ \square

En el siguiente teorema se establece que la definibilidad existencial depende de la preservación en los embeddings entre las estructuras de \mathcal{K} . Notar que esta condición incluye a los isomorfismos de estructuras de \mathcal{K} , chequeados para el Teorema 13, pero no a todos los isomorfismos entre subestructuras de estructuras de \mathcal{K} , chequeados para el Teorema 15.

Teorema 19. *Sean $\mathcal{L} \subseteq \mathcal{L}'$ lenguajes de primer orden, sea $R \in \mathcal{L}' - \mathcal{L}$ un símbolo de relación n -ario. Para una clase finita \mathcal{K} de \mathcal{L}' -estructuras, los siguientes son equivalentes:*

1. Hay una fórmula en $E(\mathcal{L})$ que define R en \mathcal{K} .
2. Para todas $\mathbf{A}, \mathbf{B} \in \mathcal{K}$, todo embedding $\gamma : \mathbf{A}_{\mathcal{L}} \rightarrow \mathbf{B}_{\mathcal{L}}$ preserva R .

Demostración. $1 \Rightarrow 2$) Sea φ existencial que define a R en \mathcal{K} , y sea $\gamma : \mathbf{A} \rightarrow \mathbf{B}$ un embedding. Entonces $\gamma : \mathbf{A} \rightarrow \mathbf{S} \leq \mathbf{B}$ es un isomorfismo. Supongamos $\mathbf{A} \models \varphi[\bar{a}]$ entonces $\mathbf{S} \models \varphi[\gamma(\bar{a})]$. Como φ es de la forma $\varphi = \exists \bar{y} \psi(\bar{x}, \bar{y})$ con ψ abierta, existe un \bar{s} tal que $\mathbf{S} \models \psi[\gamma(\bar{a}), \bar{s}]$. Como ψ es abierta y $\mathbf{S} \leq \mathbf{B}$, por el Lema 7 $\mathbf{B} \models \psi[\gamma(\bar{a}), \bar{s}]$. Entonces $\mathbf{B} \models \varphi[\gamma(\bar{a})]$ y como φ define a R en \mathcal{K} , entonces $\gamma(\bar{a}) \in R^{\mathbf{B}}$.

$2 \Rightarrow 1$) Para cada $\mathbf{A} \in \mathcal{K}$, cada $\bar{a} \in R^{\mathbf{A}}$, tomo \bar{a}' tal que $\langle \bar{a}, \bar{a}' \rangle^{\mathbf{A}} = \mathbf{A}$. Ahora tomo

$$\delta_{\mathbf{A}, (\bar{a}, \bar{a}')}(\bar{x}, \bar{y}) = \bigwedge_{\alpha \in \Delta_{(\bar{a}, \bar{a}'), \mathbf{A}}} \alpha(\bar{x}, \bar{y})$$

, que es fórmula por Teorema 5. Ahora tomo

$$\varphi = \bigvee_{\mathbf{A} \in \mathcal{K}} \left(\bigvee_{\bar{a} \in R^{\mathbf{A}}} (\exists \bar{y} \delta_{\mathbf{A}, (\bar{a}, \bar{a}')}(\bar{x}, \bar{y})) \right)$$

Sea $(b_1, \dots, b_n) \in R^{\mathbf{B}}$. Entonces como

$$\varphi = \dots \vee \exists \bar{y} \delta_{\mathbf{B}, (\bar{b}, \bar{b}')}(\bar{x}, \bar{y}) \vee \dots,$$

y como

$$\mathbf{B} \models \exists \bar{y} \delta_{\mathbf{B}, (\bar{b}, \bar{b}')}(\bar{x}, \bar{y}) [\bar{b}, \bar{b}'],$$

entonces $\mathbf{B} \models \varphi [\bar{b}]$.

Sea $\mathbf{B} \models \varphi [b_1, \dots, b_n]$. Entonces para algún $\mathbf{A} \in \mathcal{K}$, y algún $\bar{a} \in R^{\mathbf{A}}$ tales que $\mathbf{B} \models \exists \bar{y} \delta_{\mathbf{A}, (\bar{a}, \bar{a}')} (\bar{x}, \bar{y}) [\bar{b}]$. Entonces por el Lema 18 hay $\gamma : \mathbf{A} \rightarrow \mathbf{B}$ embedding tal que $\gamma(\bar{a}) = \bar{b}$. Finalmente como $\bar{a} \in R^{\mathbf{A}}$, $\gamma(\bar{a}) \in R^{\mathbf{B}}$, $\bar{b} \in R^{\mathbf{B}}$.

Notar que como φ es disyunción de existenciales, por teorema puede ser convertida en una fórmula existencial. \square

2.6. Definibilidad por fórmulas existenciales positivas

Ahora estableceremos la noción semántica para la definibilidad existencial positiva.

En el siguiente lema, construimos una formula equivalente al diagrama atómico positivo para un conjunto de elementos tal que sea capaz de generar la estructura completa análogamente al Lema 18.

Lema 20. *Sea \mathbf{A} una estructura finita y $a_1, \dots, a_n \in A$. Entonces hay una fórmula existencial positiva $\varphi(x_1, \dots, x_n)$ tal que para toda estructura \mathbf{B} y para todo $\bar{b} \in B^n$ los siguientes son equivalentes:*

1. $\mathbf{B} \models \varphi [\bar{b}]$.
2. Hay un homomorfismo $h : \mathbf{A} \rightarrow \mathbf{B}$ tal que $h(\bar{a}) = \bar{b}$

Demostración. Sean a'_1, \dots, a'_m tales que $\langle a_1, \dots, a_n, a'_1, \dots, a'_m \rangle^{\mathbf{A}} = \mathbf{A}$ y sea

$$\varphi = \exists y_1, \dots, y_m \left(\bigwedge_{\alpha \in \Delta_{(\bar{a}, \bar{a}')}^+, \mathbf{A}} \alpha(\bar{x}, \bar{y}) \right)$$

que es fórmula por el Teorema 5.

Supongamos $\mathbf{B} \models \varphi [\bar{b}]$, entonces existen $b'_1, \dots, b'_m \in B$ tales que

$$\mathbf{B} \models \bigwedge_{\alpha \in \Delta_{(\bar{a}, \bar{a}')}^+, \mathbf{A}} \alpha(\bar{x}, \bar{y}) [\bar{b}, \bar{b}']$$

por Lema 12 hay homomorfismo

$$h : \langle \bar{a}, \bar{a}' \rangle^{\mathbf{A}} \rightarrow \langle \bar{b}, \bar{b}' \rangle^{\mathbf{B}}$$

tal que $h(\bar{a}, \bar{a}') = (\bar{b}, \bar{b}')$, y como $\langle \bar{a}, \bar{a}' \rangle^{\mathbf{A}} = \mathbf{A}$ y $\langle \bar{b}, \bar{b}' \rangle^{\mathbf{B}} \leq \mathbf{B}$, h es un homomorfismo de \mathbf{A} en \mathbf{B} tal que $h(\bar{a}) = \bar{b}$.

Supongamos que hay un homomorfismo $h : \mathbf{A} \rightarrow \mathbf{B}$ tal que $h(\bar{a}) = \bar{b}$. Como $\mathbf{A} \models \varphi [\bar{a}]$, entonces $\mathbf{A} \models \Delta_{(\bar{a}, \bar{a}')}^+, \mathbf{A} [\bar{a}, \bar{a}']$ y por Lema 10

$$\mathbf{B} \models \Delta_{(\bar{a}, \bar{a}')}^+, \mathbf{A} [h(\bar{a}), h(\bar{a}')] .$$

Entonces $\mathbf{B} \models \Delta_{(\bar{a}, \bar{a}')}^+, \mathbf{A} [\bar{b}, \bar{b}']$, por lo que claramente $\mathbf{B} \models \varphi [\bar{b}]$. \square

El siguiente teorema establece que la definibilidad existencial positiva depende de la preservación en los homomorfismos entre las estructuras de \mathcal{K} . Notar que esta condición incluye a los embeddings entre estructuras de \mathcal{K} , chequeados para el Teorema 19, pero no a todos los homomorfismos entre subestructuras de estructuras de \mathcal{K} , chequeados para el Teorema 17, mientras que es disjunto con los isomorfismos entre subestructuras de estructuras de \mathcal{K} , chequeados para el Teorema 15.

Teorema 21. *Sean $\mathcal{L} \subseteq \mathcal{L}'$ lenguajes de primer orden, sea $R \in \mathcal{L}' - \mathcal{L}$ un símbolo de relación n -ario. Para una clase finita \mathcal{K} de \mathcal{L}' -estructuras, los siguientes son equivalentes:*

1. *Hay una fórmula en $E^+(\mathcal{L})$ que define R en \mathcal{K} .*
2. *Para todas $A, B \in \mathcal{K}$, todo homomorfismo $\gamma : A_{\mathcal{L}} \rightarrow B_{\mathcal{L}}$ preserva R .*

Demostración. Igual a la prueba del Teorema 19, pero utilizando las fórmulas dadas por el Lema 20. \square

2.7. Definibilidad por conjunción de atómicas y primitivas positivas

En este tipo de definibilidad se establece una relación con el producto de estructuras en \mathcal{K} .

Lo primero es definir nuevamente el concepto de preservación pero esta vez a través del producto de estructuras. Podemos pensar la siguiente definición matricialmente, donde las filas pertenecen a D mientras que cada i -ésima columna pertenece a R^{A_i}

Definición 22. Sean A_1, \dots, A_m, B conjuntos, $R^{A_i} \subseteq A_i^n$ para cada $i \in [1, m]$, $R^B \subseteq B^n$ y $h : D \subseteq A_1 \times \dots \times A_m \rightarrow B$ una función. Diremos que h *preserva* R si dados

$$\bar{a}_1 = (a_{11}, \dots, a_{1m}), \dots, \bar{a}_n = (a_{n1}, \dots, a_{nm}) \in D$$

tales que

$$(a_{1i}, \dots, a_{ni}) \in R^{A_i} \text{ para cada } i \in [1, m]$$

se tiene que $(h(\bar{a}_1), \dots, h(\bar{a}_n)) \in R^B$.

El siguiente lema establece que el producto entre estructuras preserva las formulas atómicas.

Lema 23. *Sean A_1, \dots, A_m estructuras y $\varphi = \exists \bar{y} \psi(\bar{x}, \bar{y})$ con ψ una conjunción de fórmulas atómicas. Entonces son equivalentes:*

1. *Para todo $i = 1, \dots, m$ se da que $A_i \models \varphi[a_{1i}, \dots, a_{ni}]$*
2. *$A_1 \times \dots \times A_m \models \varphi[\bar{a}_1, \dots, \bar{a}_n]$, con $\bar{a}_j = (a_{j1}, \dots, a_{jm})$*

Demostración. Rutina. \square

En el siguiente teorema se establece la dependencia de la definibilidad por conjunción de atómicas con la preservación en homomorfismos desde subestructuras de los productos de las estructuras de \mathcal{K} en las estructuras de \mathcal{K} .

Teorema 24. Sean $\mathcal{L} \subseteq \mathcal{L}'$ lenguajes de primer orden, sea $R \in \mathcal{L}' - \mathcal{L}$ un símbolo de relación n -ario. Para una clase finita \mathcal{K} de \mathcal{L}' -estructuras, los siguientes son equivalentes:

1. Para cada $m \geq 1$, para todas $\mathbf{A}_1, \dots, \mathbf{A}_m, \mathbf{B} \in \mathcal{K}$, para cada

$$\mathbf{S} \leq (\mathbf{A}_1 \times \dots \times \mathbf{A}_m)_{\mathcal{L}},$$

todo homomorfismo $h : \mathbf{S} \rightarrow \mathbf{B}_{\mathcal{L}}$ preserva R .

2. Hay una conjunción finita de \mathcal{L} -fórmulas atómicas que define R en \mathcal{K} .

Demostración. $2 \Rightarrow 1$) Sea $\varphi(\bar{x})$ conjunción de atómicas que define a R en \mathcal{K} . Sean $m \geq 1$, $\mathbf{A}_1, \dots, \mathbf{A}_m, \mathbf{B} \in \mathcal{K}$, $\mathbf{S} \leq (\mathbf{A}_1 \times \dots \times \mathbf{A}_m)_{\mathcal{L}}$, y $h : \mathbf{S} \rightarrow \mathbf{B}_{\mathcal{L}}$ un homomorfismo. Sean

$$\bar{s}_1 = (s_{11}, \dots, s_{1m}), \dots, \bar{s}_n = (s_{n1}, \dots, s_{nm}) \in S$$

tales que

$$(s_{1i}, \dots, s_{ni}) \in R^{A_i} \text{ para cada } i \in \{1, \dots, m\}.$$

Veamos que $(h(\bar{s}_1), \dots, h(\bar{s}_n)) \in R^{\mathbf{B}}$. Como $(s_{1i}, \dots, s_{ni}) \in R^{A_i}$ y φ define a R en \mathcal{K} , tenemos que

$$\mathbf{A}_i \models \varphi[s_1, \dots, s_{ni}] \text{ para } i \in \{1, \dots, m\}.$$

Luego, por el Lema 23,

$$\mathbf{A}_1 \times \dots \times \mathbf{A}_m \models \varphi[\bar{s}_1, \dots, \bar{s}_n].$$

Como φ es abierta y $\mathbf{S} \leq (\mathbf{A}_1 \times \dots \times \mathbf{A}_m)_{\mathcal{L}}$, entonces el Lema 7 implica que $\mathbf{S} \models \varphi[\bar{s}_1, \dots, \bar{s}_n]$, y aplicando luego el Lema 10 obtenemos que

$$\mathbf{B} \models \varphi[h(\bar{s}_1), \dots, h(\bar{s}_n)].$$

Luego, como φ define a R en \mathcal{K} y $\mathbf{B} \in \mathcal{K}$, concluimos que

$$(h(\bar{s}_1), \dots, h(\bar{s}_n)) \in R^{\mathbf{B}}$$

- 1 \Rightarrow 2) Supongamos $\mathcal{K} = \{\mathbf{A}_1, \dots, \mathbf{A}_m\}$, R n -aria. Sean

$$\bar{x}_1, \dots, \bar{x}_n \in \mathbf{A}_1^{|R^{\mathbf{A}_1}|} \times \dots \times \mathbf{A}_m^{|R^{\mathbf{A}_m}|}$$

tales que

$$\bar{x}_i = (x_{i1}^1, \dots, x_{i|R^{\mathbf{A}_1}|}^1, \dots, x_{i1}^m, \dots, x_{i|R^{\mathbf{A}_m}|}^m)$$

para cada $i \in \{1, \dots, m\}$ con

$$\bigcup R^{\mathbf{A}_j} = \left\{ (x_{1i}^j, \dots, x_{ni}^j) \text{ con } i \in \{1, |R^{\mathbf{A}_j}|\} \right\}$$

para cada $i \in \{1, \dots, m\}$. Sea

$$\varphi = \bigwedge_{\alpha \in \Delta^+} \alpha(\bar{x})$$

$$\alpha \in \Delta^+ \text{ sobre } \mathbf{A}_1^{|R^{\mathbf{A}_1}|} \times \dots \times \mathbf{A}_m^{|R^{\mathbf{A}_m}|}, (\bar{x}_1, \dots, \bar{x}_n)$$

. Supongamos $\mathbf{B} \models \varphi[b_1, \dots, b_n]$ y veamos que $(b_1, \dots, b_n) \in R^{\mathbf{B}}$. Como

$$\mathbf{B} \models \Delta^+_{\mathbf{A}_1^{|R^{\mathbf{A}_1|}} \times \dots \times \mathbf{A}_m^{|R^{\mathbf{A}_m|}}, (\bar{x}_1, \dots, \bar{x}_n)} [b_1, \dots, b_n]$$

, por Lema 12 hay h homomorfismo de $\langle \bar{x}_1, \dots, \bar{x}_n \rangle_{\mathbf{A}_1^{|R^{\mathbf{A}_1|}} \times \dots \times \mathbf{A}_m^{|R^{\mathbf{A}_m|}}}$ en $\langle b_1, \dots, b_n \rangle^{\mathbf{B}}$ tal que $h(\bar{x}_i) = b_i$. Como por hipótesis h preserva R , entonces como $(x_{1i}^j, \dots, x_{ni}^j) \in R^{\mathbf{A}_j}$, $(h(\bar{x}_1), \dots, h(\bar{x}_n)) \in R^{\mathbf{B}}$, que es exactamente $(b_1, \dots, b_n) \in R^{\mathbf{B}}$.

Ahora supongamos $(b_1, \dots, b_n) \in R^{\mathbf{B}}$ y veamos que $\mathbf{B} \models \varphi[b_1, \dots, b_n]$, entonces hay algún j tal que $\mathbf{A}_j = \mathbf{B}$ y hay algún k y algún j tal que

$$(x_{1k}^j, \dots, x_{nk}^j) = (b_1, \dots, b_n)$$

. Entonces como

$$\mathbf{A}_1^{|R^{\mathbf{A}_1|}} \times \dots \times \mathbf{B}^{|R^{\mathbf{B}}|} \times \dots \times \mathbf{A}_m^{|R^{\mathbf{A}_m|}} \models \varphi[\bar{x}_1, \dots, \bar{x}_n]$$

por Lema 23 $\mathbf{B} \models \varphi[x_{1k}^j, \dots, x_{nk}^j]$, por lo tanto $\mathbf{B} \models \varphi[b_1, \dots, b_n]$. \square

El siguiente teorema establece la dependencia de la definibilidad primitiva positiva con la preservación en homomorfismos de los productos de las estructuras de \mathcal{K} en las estructuras de \mathcal{K} .

Teorema 25. Sean $\mathcal{L} \subseteq \mathcal{L}'$ lenguajes de primer orden, sea $R \in \mathcal{L}' - \mathcal{L}$ un símbolo de relación n -ario. Para una clase finita \mathcal{K} de \mathcal{L}' -estructuras, los siguientes son equivalentes:

1. Para cada $m \geq 1$, para todas $\mathbf{A}_1, \dots, \mathbf{A}_m, \mathbf{B} \in \mathcal{K}$, todo homomorfismo $h : (\mathbf{A}_1 \times \dots \times \mathbf{A}_m)_{\mathcal{L}} \rightarrow \mathbf{B}_{\mathcal{L}}$ preserva R .
2. Hay una \mathcal{L} -fórmula primitiva positiva que define R en \mathcal{K} .

Demostración. $2 \Rightarrow 1$) Sea $\varphi = \exists \bar{y} \psi(\bar{x}, \bar{y})$ primitiva positiva que define a R en \mathcal{K} . Sean $m \geq 1$, $\mathbf{A}_1, \dots, \mathbf{A}_m, \mathbf{B} \in \mathcal{K}$ y $h : (\mathbf{A}_1 \times \dots \times \mathbf{A}_m)_{\mathcal{L}} \rightarrow \mathbf{B}_{\mathcal{L}}$ un homomorfismo. Sean

$$\bar{a}_1 = (a_{11}, \dots, a_{1m}), \dots, \bar{a}_n = (a_{n1}, \dots, a_{nm}) \in S$$

tales que

$$(a_{1i}, \dots, a_{ni}) \in R^{\mathbf{A}_i} \text{ para cada } i \in \{1, \dots, m\}.$$

Veamos que $(h(\bar{a}_1), \dots, h(\bar{a}_n)) \in R^{\mathbf{B}}$. Como $(a_{1i}, \dots, a_{ni}) \in R^{\mathbf{A}_i}$ y φ define a R en \mathcal{K} , tenemos que

$$\mathbf{A}_i \models \varphi[a_1, \dots, a_{ni}] \text{ para } i \in \{1, \dots, m\}.$$

Luego, por el Lema 23,

$$\mathbf{A}_1 \times \dots \times \mathbf{A}_m \models \varphi[\bar{a}_1, \dots, \bar{a}_n].$$

Entonces hay $b_1, \dots, b_k \in A_1 \times \dots \times A_m$, tales que

$$\mathbf{A}_1 \times \dots \times \mathbf{A}_m \models \psi[\bar{a}_1, \dots, \bar{a}_n, b_1, \dots, b_k].$$

Como ψ es abierta positiva, aplicando el Lema 10 obtenemos que

$$\mathbf{B} \models \varphi[h(\bar{a}_1), \dots, h(\bar{a}_n)].$$

Luego, como φ define a R en \mathcal{K} y $\mathbf{B} \in \mathcal{K}$, concluimos que

$$(h(\bar{a}_1), \dots, h(\bar{a}_n)) \in R^{\mathbf{B}}$$

1 \Rightarrow 2) Supongamos $\mathcal{K} = \{\mathbf{A}_1, \dots, \mathbf{A}_m\}$, R n -aria. Sean

$$\bar{x}_1, \dots, \bar{x}_n \in \mathbf{A}_1^{|R^{\mathbf{A}_1}|} \times \dots \times \mathbf{A}_m^{|R^{\mathbf{A}_m}|}$$

tales que

$$\bar{x}_i = (x_{i1}^1, \dots, x_{i|R^{\mathbf{A}_1}|}^1, \dots, x_{i1}^m, \dots, x_{i|R^{\mathbf{A}_m}|}^m)$$

para cada $i \in \{1, \dots, m\}$ con

$$\bigcup R^{\mathbf{A}_j} = \left\{ (x_{1i}^j, \dots, x_{ni}^j) \text{ con } i \in \{1, |R^{\mathbf{A}_j}|\} \right\}$$

para cada $i \in \{1, \dots, m\}$. Sean

$$\bar{x}'_1, \dots, \bar{x}'_k \in \mathbf{A}_1^{|R^{\mathbf{A}_1}|} \times \dots \times \mathbf{A}_m^{|R^{\mathbf{A}_m}|},$$

tales que

$$\langle \bar{x}_1, \dots, \bar{x}_n, \bar{x}'_1, \dots, \bar{x}'_k \rangle_{\mathbf{A}_1^{|R^{\mathbf{A}_1}|} \times \dots \times \mathbf{A}_m^{|R^{\mathbf{A}_m}|}} = \mathbf{A}_1^{|R^{\mathbf{A}_1}|} \times \dots \times \mathbf{A}_m^{|R^{\mathbf{A}_m}|}.$$

Sea

$$\varphi = \exists \bar{y} \bigwedge_{\substack{\alpha \in \Delta^+ \\ \mathbf{A}_1^{|R^{\mathbf{A}_1}|} \times \dots \times \mathbf{A}_m^{|R^{\mathbf{A}_m}|}, (\bar{x}_1, \dots, \bar{x}_n, \bar{x}'_1, \dots, \bar{x}'_k)}} \alpha(\bar{x}, \bar{y}).$$

Supongamos $\mathbf{B} \models \varphi[b_1, \dots, b_n]$ y veamos que $(b_1, \dots, b_n) \in R^{\mathbf{B}}$. Entonces hay b'_1, \dots, b'_k tal que

$$\mathbf{B} \models \Delta^+_{\mathbf{A}_1^{|R^{\mathbf{A}_1}|} \times \dots \times \mathbf{A}_m^{|R^{\mathbf{A}_m}|}, (\bar{x}_1, \dots, \bar{x}_n, \bar{x}'_1, \dots, \bar{x}'_k)} [b_1, \dots, b_n, b'_1, \dots, b'_k],$$

por Lema 12 hay h homomorfismo de

$$\langle \bar{x}_1, \dots, \bar{x}_n, \bar{x}'_1, \dots, \bar{x}'_k \rangle_{\mathbf{A}_1^{|R^{\mathbf{A}_1}|} \times \dots \times \mathbf{A}_m^{|R^{\mathbf{A}_m}|}}$$

en

$$\langle b_1, \dots, b_n, b'_1, \dots, b'_k \rangle^{\mathbf{B}}$$

tal que $h(\bar{x}_i) = b_i$ y $h(\bar{x}'_i) = b'_i$, claramente h es homomorfismo de

$$\mathbf{A}_1^{|R^{\mathbf{A}_1}|} \times \dots \times \mathbf{A}_m^{|R^{\mathbf{A}_m}|}$$

en \mathbf{B} . Como por hipótesis h preserva R , entonces como $(x_{1i}^j, \dots, x_{ni}^j) \in R^{\mathbf{A}_j}$ se da que $(h(\bar{x}_1), \dots, h(\bar{x}_n)) \in R^{\mathbf{B}}$, que es exactamente $(b_1, \dots, b_n) \in R^{\mathbf{B}}$.

2.7. DEFINIBILIDAD POR CONJUNCIÓN DE ATÓMICAS Y PRIMITIVAS POSITIVAS 29

Ahora supongamos $(b_1, \dots, b_n) \in R^{\mathbf{B}}$ y veamos que $\mathbf{B} \models \varphi[b_1, \dots, b_n]$, entonces hay algún j tal que $\mathbf{A}_j = \mathbf{B}$ y hay algún k y algún j tal que

$$(x_{1k}^j, \dots, x_{nk}^j) = (b_1, \dots, b_n).$$

Entonces como

$$\mathbf{A}_1^{|R^{\mathbf{A}_1}|} \times \dots \times \mathbf{B}^{|R^{\mathbf{B}}|} \times \dots \times \mathbf{A}_m^{|R^{\mathbf{A}_m}|} \models \varphi[\bar{x}_1, \dots, \bar{x}_n]$$

por Lema 23 $\mathbf{B} \models \varphi[x_{1k}^j, \dots, x_{nk}^j]$, por lo tanto $\mathbf{B} \models \varphi[b_1, \dots, b_n]$. □

Capítulo 3

Algoritmos de chequeo de definibilidad

En este capítulo describimos algoritmos para el chequeo de definibilidad de relaciones en conjuntos finitos de estructuras finitas por fragmentos de primer orden.

3.1. Definiciones e ideas preliminares

Para evitar las complicaciones resultantes de que tanto una estructura \mathbf{A} y una subestructura propia \mathbf{A}_0 pertenezcan al conjunto \mathcal{K} , definiremos el concepto de clase *disjunta*. En general asumiremos que nuestra clase \mathcal{K} es siempre disjunta. Notar que no hay pérdida de generalidad ya que bastaría con renombrar elementos para convertir una clase no disjunta en una disjunta equivalente. Una clase \mathcal{K} de estructuras será *disjunta* si dadas $\mathbf{A}, \mathbf{B} \in \mathcal{K}$ se tiene que $A \cap B \neq \emptyset$ si y solo si $\mathbf{A} = \mathbf{B}$. La clase \mathcal{K} será *normal* si es disjunta, finita y cada una de las estructuras en \mathcal{K} es finita.

A la hora de reducir la complejidad de nuestros algoritmos en la búsqueda de homomorfismos utilizaremos conjuntos reducidos de flechas, que *generan* el resto. Dados \mathcal{F}_0 y \mathcal{F} dos conjuntos de funciones, diremos que \mathcal{F}_0 *genera* a \mathcal{F} si $\mathcal{F}_0 \subseteq \mathcal{F}$ y para toda $f \in \mathcal{F}$ hay $f_1, \dots, f_n \in \mathcal{F}_0$ (quizás repetidas) tales que $f = f_1 \circ \dots \circ f_n$.

Dado un conjunto \mathcal{K} de estructuras, para referirnos a las flechas nombradas en los Teoremas 13, 15, 21, 17 definimos los siguientes conjuntos:

$$\text{iso}(\mathcal{K}) = \{\gamma : \gamma \text{ isomorfismo de } \mathbf{A} \text{ en } \mathbf{B} \text{ con } \mathbf{A}, \mathbf{B} \in \mathcal{K}\}$$

$$\text{sub iso}(\mathcal{K}) = \{\gamma : \gamma \text{ isomorfismo de } \mathbf{A}_0 \text{ en } \mathbf{B}_0 \text{ con } \mathbf{A}_0, \mathbf{B}_0 \in \mathcal{S}(\mathcal{K})\}$$

$$\text{hom}(\mathcal{K}) = \{\gamma : \gamma \text{ homomorfismo de } \mathbf{A} \text{ en } \mathbf{B} \text{ con } \mathbf{A}, \mathbf{B} \in \mathcal{K}\}$$

$$\text{sub hom}(\mathcal{K}) = \{\gamma : \gamma \text{ homomorfismo de } \mathbf{A}_0 \text{ en } \mathbf{B}_0 \text{ con } \mathbf{A}_0, \mathbf{B}_0 \in \mathcal{S}(\mathcal{K})\}$$

3.2. Preprocesamiento

Una vez que tenemos una clase normal \mathcal{K} en la que chequear definibilidad tratamos de reducir su redundancia basándonos en el siguiente resultado:

Lema 26. *Sean $\mathcal{L} \subseteq \mathcal{L}'$ lenguajes de primer orden, \mathcal{K} una clase normal de \mathcal{L}' -estructuras y $\mathbf{A}, \tilde{\mathbf{A}} \in \mathcal{K}$, con $\mathbf{A} \neq \tilde{\mathbf{A}}$, tales que hay un \mathcal{L} -isomorfismo $\gamma : \mathbf{A} \rightarrow \tilde{\mathbf{A}}$. Entonces para toda $R \in \mathcal{L}' - \mathcal{L}$ y toda \mathcal{L} -fórmula φ son equivalentes:*

1. φ define a R en \mathcal{K} .
2. γ es un $\mathcal{L} \cup \{R\}$ -isomorfismo y φ define a R en $\mathcal{K} - \{\tilde{\mathbf{A}}\}$.

Demostración. $1 \Rightarrow 2$) Trivial.

$2 \Rightarrow 1$) $\tilde{\mathbf{A}} \models R[\bar{a}]$ sii $\mathbf{A} \models R[\gamma^{-1}(\bar{a})]$ sii $\mathbf{A} \models \varphi[\gamma^{-1}(\bar{a})]$ sii $\tilde{\mathbf{A}} \models \varphi[\bar{a}]$. \square

Independientemente del tipo de definibilidad que uno quiera chequear el Lema 26 sugiere que uno puede comenzar por reducir \mathcal{K} eliminando copias de estructuras $\mathcal{L} \cup \{R\}$ -isomorfas. El Algoritmo 3.1 aprovecha este resultado aprovechando además para chequear por \mathcal{L} -isomorfismos y al descubrir uno de estos comprobar si es un $\mathcal{L} \cup \{R\}$ -isomorfismo, lo que permite encontrar contraejemplos de definibilidad desde esta etapa ya que el Teorema 13 establece que los isomorfismos entre estructuras de \mathcal{K} deben preservar para que R sea definible en primer orden.

Algorithm 3.1 Preprocesamiento

```

1: for  $\mathbf{A} \in \mathcal{K}$  do
2:   for  $\mathbf{B} \in \mathcal{K} - \{\mathbf{A}\}$  do
3:     if hay  $\gamma : \mathbf{A} \rightarrow \mathbf{B}$   $\mathcal{L}$ -isomorfismo then
4:       if  $\gamma$  es un  $\mathcal{L} \cup \{R\}$ -isomorfismo then
5:          $\mathcal{K} = \mathcal{K} - \{\mathbf{B}\}$ 
6:       else
7:         return  $\gamma$   $\triangleright R$  no es definible y  $\gamma$  es contraejemplo
8:       end if
9:     end if
10:  end for
11: end for
12: return  $\mathcal{K}$   $\triangleright \mathcal{K}$  sin estructuras  $\mathcal{L}$ -isomorfas

```

Notar que los isomorfismos revisados por este algoritmo deben ser necesariamente revisados para comprobar definibilidad en cualquiera de los formatos, con la ventaja de que al chequear estos en primer lugar, podría reducirse la clase \mathcal{K} en el proceso.

Observar además que, para poder aplicar los teoremas de definibilidad de las secciones 2.3 y 2.4, habría que verificar que cada uno de los \mathcal{L} -isomorfismos entre estructuras de \mathcal{K} preserven R y vemos que basta con chequear sólo uno gracias al Lema 26.

Para ejemplificar el funcionamiento del algoritmo, supongamos que \mathcal{K} está dado por la Figura 3.2.1a, donde cada tipo de isomorfismo está representado por una forma geométrica.

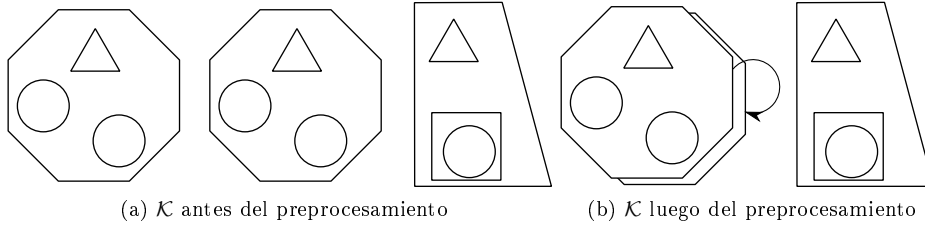


Figura 3.2.1

El algoritmo de preprocesamiento detecta el \mathcal{L} -isomorfismo entre los dos octógonos y revisa que sea también un $\mathcal{L} \cup \{R\}$ -isomorfismo, dando lugar a la Figura 3.2.1b, donde la flecha es el isomorfismo a chequear.

3.3. Definibilidad abierta

A partir del Teorema 15, habría que chequear todo isomorfismo entre subestructuras de estructuras de \mathcal{K} , como en el siguiente lema.

Lema 27. *Sean $\mathcal{L} \subseteq \mathcal{L}'$ lenguajes de primer orden tales que $R \in \mathcal{L}' - \mathcal{L}$ es un símbolo de relación n -ario y \mathcal{K} un conjunto normal de \mathcal{L}' -estructuras. Entonces si cada $\gamma \in \text{sub iso}(\mathcal{K}_{\mathcal{L}})$ preserva R , R es definible por una \mathcal{L} -fórmula abierta en \mathcal{K} .*

Demostración. Directo del Teorema 15.

En el siguiente teorema describimos un conjunto suficiente de flechas para generar $\text{sub iso}(\mathcal{K}_{\mathcal{L}})$. Notar que al tomar \mathcal{K} como un conjunto normal de \mathcal{L} -estructuras sin estructuras isomorfas estamos pensando en la aplicación previa del Algoritmo 3.1, basado en el Lema 26. \square

Teorema 28. *Sea \mathcal{K} un conjunto normal de \mathcal{L} -estructuras donde no hay estructuras isomorfas. Sea $\mathcal{S} \subseteq \mathcal{S}(\mathcal{K})$ tal que contiene exactamente un representante por cada tipo de isomorfismo en $\mathcal{S}(\mathcal{K})$. Sea $\mathcal{F} \subseteq \text{sub iso}(\mathcal{K})$ tal que:*

1. *para cada $\mathbf{S} \in \mathcal{S}$ se tiene que $\text{aut}(\mathbf{S}) \subseteq \mathcal{F}$,*
2. *para cada $\mathbf{A} \in \mathcal{S}(\mathcal{K}) - \mathcal{S}$ hay $\gamma, \gamma^{-1} \in \mathcal{F}$ para algún isomorfismo γ de \mathbf{A} en el representante del tipo de isomorfismo de \mathbf{A} en \mathcal{S} .*

Entonces \mathcal{F} genera $\text{sub iso}(\mathcal{K})$.

Demostración. Sea $\gamma \in \text{sub iso}(\mathcal{K})$, entonces $\gamma : \mathbf{A}_0 \leq \mathbf{A} \rightarrow \mathbf{B}_0 \leq \mathbf{B}$ isomorfismo, con $\mathbf{A}, \mathbf{B} \in \mathcal{K}$.

Si $\mathbf{A}_0, \mathbf{B}_0 \in \mathcal{S}$, como no hay estructuras isomorfas en \mathcal{S} , $\mathbf{A}_0 = \mathbf{B}_0$, por lo tanto $\gamma \in \text{aut}(\mathbf{A}_0) \subseteq \mathcal{F}$.

Si $\mathbf{A}_0 \in \mathcal{S}$ pero $\mathbf{B}_0 \notin \mathcal{S}$, entonces \mathbf{A}_0 es el representante de \mathbf{B}_0 en \mathcal{S} , y por lo tanto hay un isomorfismo $\delta : \mathbf{B}_0 \rightarrow \mathbf{A}_0$ tal que $\delta, \delta^{-1} \in \mathcal{F}$. Claramente $\delta\gamma = \lambda \in \text{aut}(\mathbf{A}_0) \subseteq \mathcal{F}$, por lo tanto $\gamma = \delta^{-1}\lambda$.

Si $\mathbf{A}_0, \mathbf{B}_0 \notin \mathcal{S}$, hay $\mathbf{C}_0 \in \mathcal{S}$ que es representante de \mathbf{A}_0 y \mathbf{B}_0 . Luego hay $\delta : \mathbf{A}_0 \rightarrow \mathbf{C}_0$ y $\delta' : \mathbf{B}_0 \rightarrow \mathbf{C}_0$ tales que $\delta, \delta^{-1}, \delta', \delta'^{-1} \in \mathcal{F}$. Claramente $\delta'\gamma\delta^{-1} = \lambda \in \text{aut}(\mathbf{C}_0) \subseteq \mathcal{F}$, por lo tanto $\gamma = \delta'^{-1}\lambda\delta$. \square

Corolario 29. Sean $\mathcal{L} \subseteq \mathcal{L}'$ lenguajes de primer orden tales que $R \in \mathcal{L}' - \mathcal{L}$ es un símbolo de relación n -ario y \mathcal{K} un conjunto normal de \mathcal{L}' -estructuras. Supongamos \mathcal{F} es como en el Teorema 28 para la familia $\mathcal{K}_{\mathcal{L}}$. Entonces si cada $\gamma \in \mathcal{F}$ preserva R , se tiene que R es definible por una \mathcal{L} -fórmula abierta en \mathcal{K} .

Demostración. Directa ya que la composición de funciones que preservan R , trivialmente también preserva R . Luego aplicando el Teorema 28 y el Lema 27 queda probado. \square

El Teorema 28 sugiere un subconjunto de isomorfismos que bastan para revisar definibilidad abierta a través del Corolario 29.

Algorithm 3.2 Chequeo de definibilidad abierta

```

1:  $\mathcal{S} = \emptyset$ 
2: for  $\mathbf{A} \in \mathcal{K}$ , desde la mayor a la menor cardinalidad do
3:    $Sub = [\mathbf{C} : \mathbf{C} \in \mathbb{S}_{\mathcal{L}}(\mathbf{A}), \text{ de la mayor a menor cardinalidad}]$ 
4:   for  $\mathbf{B} \in Sub$  do
5:      $(iso, ce) = \text{TIENEREPRESENTANTE}(\mathbf{B}, \mathcal{S})$      $\triangleright$  hay un  $\mathcal{L}$ -iso de  $\mathbf{B}$  en
        $\mathcal{S} \in \mathcal{S}$ 
6:     if  $ce \neq \text{Null}$  then
7:       return  $ce$      $\triangleright$   $ce$  es contraejemplo
8:     end if
9:     if  $iso$  then
10:       $Sub = [\mathbf{C} : \mathbf{C} \in Sub \text{ tal que } C \not\subseteq B]$ 
11:    else
12:       $\mathcal{S} = \mathcal{S} \cup \{\mathbf{B}\}$ 
13:      for  $\alpha \in \text{Aut}_{\mathcal{L}}(\mathbf{B})$  do
14:        if  $\alpha$  no preserva  $R$  then
15:          return  $\alpha$      $\triangleright$   $\alpha$  es contraejemplo
16:        end if
17:      end for
18:    end if
19:  end for
20: end for
21: return     $\triangleright R$  es abierta-definible

22: function TIENEREPRESENTANTE( $\mathbf{A}_0, \mathcal{S}$ )
23:   for  $\mathbf{S} \in \mathcal{S}$ , con  $|\mathbf{A}_0| = |\mathbf{S}|$  do
24:     if hay  $\gamma : \mathbf{A}_0 \rightarrow \mathbf{S}$   $\mathcal{L}$ -isomorfismo then
25:       if  $\gamma$  no es un  $\mathcal{L} \cup \{R\}$ -isomorfismo then
26:         return  $(\text{True}, \gamma)$      $\triangleright \gamma$  es contraejemplo
27:       else
28:         return  $(\text{True}, \text{Null})$      $\triangleright \gamma$  es isomorfismo con el representante
29:       end if
30:     end if
31:   end for
32:   return  $(\text{False}, \text{Null})$      $\triangleright$  No tiene representante
33: end function

```

En el Algoritmo 3.2 se construye \mathcal{S} a la vez que se van revisando los iso-

morfismos en \mathcal{F} . Para esto recorremos las estructuras en \mathcal{K} de mayor a menor tamaño. Al momento de procesar una $\mathbf{A} \in \mathcal{K}$ lo primero que se hace es buscar un \mathcal{L} -isomorfismo γ entre \mathbf{A} y un miembro \mathcal{S} . Si un tal γ existe, tanto γ como su inversa son revisados (miembros de \mathcal{F} correspondientes al punto (2) del Teorema 28). Si no hay un tal γ , la estructura \mathbf{A} es agregada a \mathcal{S} y sus \mathcal{L} -automorfismos son revisados (miembros de \mathcal{F} correspondientes al punto (1) del Teorema 28). A continuación las subestructuras de \mathbf{A} son procesadas de la misma manera.

Notar que en la línea 14 no se necesita revisar si los automorfismos son $\mathcal{L} \cup \{R\}$ -automorfismos ya que, como se recorren todos su inversa también es revisada por preservación.

Además, una vez que una subestructura resulta ser $\mathcal{L} \cup \{R\}$ -isomorfa a un representante en \mathcal{S} , sabemos que todas sus subestructuras ya tienen representante en \mathcal{S} y que los isomorfismos internos preservan. Esto permite disminuir la cantidad de subestructuras a revisar, como se ve en la línea 10.

Para esquematizar la ganancia obtenida en cuanto al chequeo de una menor cantidad de flechas podemos observar en la Figura 3.3.1a el conjunto \mathcal{K} junto con todas las flechas que deberían ser revisadas según el Teorema 15. Mientras que en la Figura 3.3.1b, se puede ver las flechas que bastan para comprobar definibilidad abierta según el Corolario 29 a la manera del algoritmo anterior.

Notar que en la Figura 3.3.1b, la aplicación del Algoritmo 3.1 (Preprocesamiento) genera que uno de los octágonos aparezca como *sombra* del otro y baste solo revisar el isomorfismo entre ellos por el Lema 26.



Figura 3.3.1

3.4. Definibilidad abierta-positiva

A partir del Teorema 17, habría que chequear todo homomorfismo entre subestructuras de estructuras de \mathcal{K} , como en el siguiente lema.

Lema 30. Sean $\mathcal{L} \subseteq \mathcal{L}'$ lenguajes de primer orden tales que $R \in \mathcal{L}' - \mathcal{L}$ es un símbolo de relación n -ario y \mathcal{K} un conjunto normal de \mathcal{L}' -estructuras. Entonces si cada $\gamma \in \text{sub hom}(\mathcal{K}_{\mathcal{L}})$ preserva R , R es definible por una \mathcal{L} -fórmula abierta positiva en \mathcal{K} .

Demostración. Directo del Teorema 17.

En el siguiente teorema describimos un conjunto suficiente de flechas para generar $\text{sub hom}(\mathcal{K}_{\mathcal{L}})$. Notar que, nuevamente, al tomar \mathcal{K} como un conjun-

to normal de \mathcal{L} -estructuras sin estructuras isomorfas estamos pensando en la aplicación previa del Algoritmo 3.1, basado en el Lema 26. \square

Teorema 31. *Sea \mathcal{K} un conjunto normal de \mathcal{L} -estructuras donde no hay estructuras isomorfas. Sea $\mathcal{S} \subseteq \mathbb{S}(\mathcal{K})$ tal que contiene exactamente un representante por cada tipo de isomorfismo en $\mathbb{S}(\mathcal{K})$. Sea $\mathcal{H} \subseteq \text{sub hom}(\mathcal{K})$ tal que:*

- $\mathcal{F} \subseteq \mathcal{H}$ con \mathcal{F} como en el Teorema 28,
- para cada $\mathbf{A}, \mathbf{B} \in \mathcal{S}$ todo $h : \mathbf{A} \rightarrow \mathbf{B}$ homomorfismo sobreyectivo esta en \mathcal{H} .

Entonces \mathcal{H} genera $\text{sub hom}(\mathcal{K})$.

Demostración. Sea $h \in \text{sub hom}(\mathcal{K})$, entonces $h : \mathbf{A}_0 \leq \mathbf{A} \rightarrow \mathbf{B}_0 \leq \mathbf{B}$ homomorfismo, con $\mathbf{A}, \mathbf{B} \in \mathcal{K}$.

Si $\mathbf{A}_0, h(\mathbf{A}_0) \in \mathcal{S}$, h es homomorfismo sobreyectivo entre estructuras de \mathcal{S} , por lo que $h \in \mathcal{H}$.

Si $\mathbf{A}_0 \in \mathcal{S}$ pero $h(\mathbf{A}_0) \notin \mathcal{S}$, hay \mathbf{S} representante de $h(\mathbf{A}_0)$ en \mathcal{S} y un isomorfismo $\delta : h(\mathbf{A}_0) \rightarrow \mathbf{S}$ tal que $\delta, \delta^{-1} \in \mathcal{F} \subseteq \mathcal{H}$. Luego $\delta^{-1}h = f$ donde f es un homomorfismo claramente sobreyectivo de \mathbf{A}_0 en \mathbf{S} , por lo que $f \in \mathcal{H}$. Finalmente $h = \delta f$.

Si $\mathbf{A}_0, h(\mathbf{A}_0) \notin \mathcal{S}$, hay respectivos $\mathbf{S}, \mathbf{S}' \in \mathcal{S}$ representantes de \mathbf{A}_0 y $h(\mathbf{A}_0)$. Luego hay $\delta : \mathbf{A}_0 \rightarrow \mathbf{S}$ y $\delta' : h(\mathbf{A}_0) \rightarrow \mathbf{S}'$ tales que $\delta, \delta^{-1}, \delta', \delta'^{-1} \in \mathcal{F} \subseteq \mathcal{H}$. Luego $\delta' h \delta^{-1} = f$ donde f es un homomorfismo claramente sobreyectivo de \mathbf{S} en \mathbf{S}' , por lo que $f \in \mathcal{H}$. Finalmente $h = \delta'^{-1} f \delta$. \square

Corolario 32. *Sean $\mathcal{L} \subseteq \mathcal{L}'$ lenguajes de primer orden tales que $R \in \mathcal{L}' - \mathcal{L}$ es un símbolo de relación n -ario y \mathcal{K} un conjunto normal de \mathcal{L}' -estructuras. Supongamos \mathcal{F} es como en el Teorema 31 para la familia $\mathcal{K}_{\mathcal{L}}$. Entonces si cada $\gamma \in \mathcal{F}$ preserva R , se tiene que R es definible por una \mathcal{L} -fórmula abierta positiva en \mathcal{K} .*

Demostración. Directa ya que la composición de funciones que preservan R , trivialmente también preserva R . Luego aplicando el Teorema 31 y el Lema 30 queda probado. \square

El Teorema 31 sugiere un subconjunto de homomorfismos que bastan para revisar definibilidad abierta positiva a través del Corolario 32. A su vez, a la hora de buscar homomorfismos entre estructuras aprovechamos el siguiente resultado, que nos permite organizar la búsqueda de homomorfismos biyectivos entre dos subestructuras en un único sentido.

Teorema 33 (Las estructuras finitas cumplen la propiedad de Cantor-Bernstein). *Dadas \mathbf{A} y \mathbf{B} estructuras finitas, si hay $f : \mathbf{A} \rightarrow \mathbf{B}$ y $g : \mathbf{B} \rightarrow \mathbf{A}$ homomorfismos inyectivos entonces hay un $\gamma : \mathbf{A} \rightarrow \mathbf{B}$ isomorfismo.*

Demostración. Sean \mathbf{A}, \mathbf{B} estructuras finitas y $f : \mathbf{A} \rightarrow \mathbf{B}$ y $g : \mathbf{B} \rightarrow \mathbf{A}$ homomorfismos inyectivos. Notar que gf es una permutación de \mathbf{A} , y como \mathbf{A} es finito hay $k \geq 1$ tal que $(gf)^k = \text{Id}_{\mathbf{A}}$. Luego como $g^{-1} = f(gf)^{k-1}$ vemos que g^{-1} es homomorfismo. \square

En el Algoritmo 3.3 se construye \mathcal{S} a la vez que se van revisando los homomorfismos en \mathcal{F} . Para esto recorremos las estructuras en \mathcal{K} de mayor a menor tamaño. Al momento de procesar una $\mathbf{A} \in \mathcal{K}$, mientras que para definibilidad abierta lo primero que se hacía era buscar un \mathcal{L} -isomorfismo para cada $\mathbf{S} \in \mathcal{S}$, ahora se buscan homomorfismos biyectivos primero, chequeándolos por preservación. Esto da lugar a tres casos:

1. Uno de estos homomorfismos biyectivos es un \mathcal{L} -isomorfismo.
2. Hay homomorfismos biyectivos de \mathbf{A} en \mathbf{S} pero ninguno es un \mathcal{L} -isomorfismo.
3. No hay homomorfismos biyectivos de \mathbf{A} en \mathbf{S} .

En el primer caso se procede como en el Algoritmo 3.2. En el segundo caso podemos estar seguros de que no hay homomorfismos biyectivos de \mathbf{S} en \mathbf{A} , ya que por el Teorema 33, debería haber un \mathcal{L} -isomorfismo entre ellos. Mientras que en el tercer caso deberemos buscar homomorfismos biyectivos de \mathbf{S} en \mathbf{A} , para revisar su preservación.

Esto nos permite estar seguros de que todos los homomorfismos biyectivos entre estructuras de \mathcal{S} han sido chequeados. Finalmente lo único que falta es chequear los homomorfismos sobreyectivos pero no inyectivos entre estructuras de \mathcal{S} . Para lo cual basta con buscar homomorfismos sobreyectivos de \mathbf{A} en \mathbf{B} tales que $|A| > |B|$.

Para esquematizar la ganancia obtenida en cuanto al chequeo de una menor cantidad de flechas podemos observar en la Figura 3.4.1a el conjunto \mathcal{K} junto con todas las flechas que deberían ser revisadas según el Teorema 17, donde las flechas punteadas son los homomorfismos. Mientras que en la Figura 3.4.1b, se puede ver las flechas que bastan para comprobar definibilidad abierta según el Teorema 31 a la manera del algoritmo anterior.



Figura 3.4.1

Algorithm 3.3 Chequeo de definibilidad abierta-positiva

```

1:  $\mathcal{S} = \emptyset$ 
2: for  $\mathbf{A} \in \mathcal{K}$ , desde la mayor a la menor cardinalidad do
3:    $Sub = [\mathbf{C} : \mathbf{C} \in \mathbb{S}_{\mathcal{L}}(\mathbf{A}), \text{ de la mayor a menor cardinalidad}]$ 
4:   for  $\mathbf{B} \in Sub$  do
5:      $(iso, ce) = \text{TIENEREPRESENTANTE}(\mathbf{B}, \mathcal{S})$        $\triangleright$  hay un  $\mathcal{L}$ -iso de  $\mathbf{B}$  en
        $\mathbf{S} \in \mathcal{S}$ 
6:     if  $ce \neq \text{null}$  then
7:       return  $ce$        $\triangleright$   $ce$  es contraejemplo
8:     end if
9:     if  $iso$  then
10:       $Sub = [\mathbf{C} : \mathbf{C} \in Sub \text{ tal que } C \not\subseteq B]$ 
11:    else
12:       $\mathcal{S} = \mathcal{S} \cup \{\mathbf{B}\}$ 
13:      for  $\alpha \in \text{Aut}_{\mathcal{L}}(\mathbf{B})$  do
14:        if  $\alpha$  no preserva  $R$  then
15:          return  $\alpha$        $\triangleright$   $\alpha$  es contraejemplo
16:        end if
17:      end for
18:    end if
19:  end for
20: end for
21: for  $\mathbf{A} \in \mathcal{S}$  do
22:   for  $\mathbf{B} \in \mathcal{S}$ , con  $|\mathbf{B}| < |\mathbf{A}|$  do
23:    for  $\gamma : \mathbf{A} \rightarrow \mathbf{B}$  con  $\gamma$   $\mathcal{L}$ -homomorfismo sobreyectivo do
24:      if  $\gamma$  no preserva  $R$  then
25:        return  $\gamma$        $\triangleright$   $\gamma$  es contraejemplo
26:      end if
27:    end for
28:  end for
29: end for
30: return       $\triangleright R$  es definible por una abierta positiva

31: function TIENEREPRESENTANTE( $\mathbf{A}_0, \mathcal{S}$ )
32:   for  $\mathbf{S} \in \mathcal{S}$ , con  $|\mathbf{A}_0| = |\mathbf{S}|$  do
33:    bihomos =  $[\gamma : \gamma : \mathbf{A}_0 \rightarrow \mathbf{S} \text{ con } \gamma \text{ } \mathcal{L}\text{-homomorfismo biyectivo}]$ 
34:    for  $\gamma \in \text{bihomos}$  do
35:      if  $\gamma$  es un  $\mathcal{L}$ -isomorfismo, con  $\mathbf{S} \in \mathcal{S}$  then
36:        if  $\gamma$  no es un  $\mathcal{L} \cup \{R\}$ -isomorfismo then
37:          return (True,  $\gamma$ )       $\triangleright$   $\gamma$  es contraejemplo
38:        else
39:          return (True, Null)       $\triangleright$   $\gamma$  es isomorfismo con el
representante
40:        end if
41:      else if  $\gamma$  no preserva  $R$  then
42:        return (False,  $\gamma$ )       $\triangleright$   $\gamma$  es contraejemplo
43:      end if
44:    end for
45:    if bihomos =  $\emptyset$  then
46:      for  $\gamma : \mathbf{S} \rightarrow \mathbf{A}_0$  con  $\gamma$   $\mathcal{L}$ -homomorfismo biyectivo do
47:        if  $\gamma$  no preserva  $R$  then
48:          return (False,  $\gamma$ )       $\triangleright$   $\gamma$  es contraejemplo
49:        end if
50:      end for
51:    end if
52:    return (False, Null)       $\triangleright$  No tiene representante
53:  end for
54: end function

```

3.5. Definibilidad de primer orden

Por el Teorema 13, luego de haber aplicado el Algoritmo 3.1 para preprocesamiento, solo basta con revisar los automorfismos para $\mathbf{A} \in \mathcal{K}$, como hacemos en el Algoritmo 3.4.

Algorithm 3.4 Chequeo de definibilidad de primer orden

```

1: for  $\mathbf{A} \in \mathcal{K}$  do
2:   for  $\alpha \in \text{Aut}_{\mathcal{L}}(\mathbf{A})$  do
3:     if  $\alpha$  no es un  $\mathcal{L} \cup \{R\}$ -automorfismo then
4:       return  $\alpha$   $\triangleright \alpha$  es contraejemplo
5:     end if
6:   end for
7: end for
8: return  $\triangleright R$  es definible en primer orden

```

3.6. Definibilidad existencial

Por el Teorema 19, luego de aplicar el Algoritmo 3.1 de preprocesamiento, solo basta con revisar los embeddings entre $\mathbf{A}, \mathbf{B} \in \mathcal{K}$, como hacemos en el Algoritmo 3.5.

Algorithm 3.5 Chequeo de definibilidad existencial

```

1: for  $\mathbf{A} \in \mathcal{K}$  do
2:   for  $\mathbf{B} \in \mathcal{K}$  do
3:     for  $\gamma \in \text{Emb}_{\mathcal{L}}(\mathbf{A}, \mathbf{B})$  do
4:       if  $\gamma$  no es un  $\mathcal{L} \cup \{R\}$ -embedding then
5:         return  $\gamma$   $\triangleright \gamma$  es contraejemplo
6:       end if
7:     end for
8:   end for
9: end for
10: return  $\triangleright R$  es existencial definible

```

3.7. Definibilidad existencial-positiva

Por el Teorema 21, luego de aplicar el Algoritmo 3.1 de preprocesamiento, solo basta con revisar los homomorfismos cada par $\mathbf{A}, \mathbf{B} \in \mathcal{K}$, como hacemos en el Algoritmo 3.6.

Algorithm 3.6 Chequeo de definibilidad existencial-positiva

```

1: for  $\mathbf{A} \in \mathcal{K}$  do
2:   for  $\mathbf{B} \in \mathcal{K}$  do
3:     for  $\alpha \in \text{Hom}_{\mathcal{L}}(\mathbf{A}, \mathbf{B})$  do
4:       if  $h$  no preserva  $R$  then
5:         return  $h$   $\triangleright h$  es contraejemplo
6:       end if
7:     end for
8:   end for
9: end for
10: return  $\triangleright R$  es existencial positiva definible

```

3.8. Detección de homomorfismos

Para la detección de homomorfismos modelizamos el problema como un CSP, y utilizamos Minion (Gent, Jefferson y Miguel 2006) como solucionador.

3.8.1. CSP

Un CSP (constraint satisfaction problem) (ver, e.g., Russell y Norvig 2010, Capítulo 6) es definido como una terna $\langle X, D, C \rangle$, donde

$X = \{X_1, \dots, X_n\}$ es un conjunto de variables

$D = \{D_1, \dots, D_n\}$ es un conjunto con los respectivos dominios de los valores

$C = \{C_1, \dots, C_m\}$ es un conjunto de restricciones

Cada variable X_i se mueve en los valores del respectivo dominio no vacío D_i . Cada restricción $C_j \in C$ es un par $\langle t_j, R_j \rangle$, donde t_j es una k -upla de variables y R_j es una relación k -aria en el correspondiente dominio de cada variable. Una valuación sobre las variables es una función desde un subconjunto de las variables a un particular conjunto de valores en los correspondientes dominios de valores. Una valuación v satisface la restricción $\langle t_j, R_j \rangle$ si los valores asignados a las variables t_j satisfacen la relación R_j .

Una valuación es consistente si no viola ninguna de las restricciones. Una valuación es completa si incluye todas las variables. Una valuación es solución si es consistente y completa. En ese caso diremos que la valuación resuelve el CSP.

3.8.2. CSP para calcular homomorfismos

Sean \mathbf{A}, \mathbf{B} \mathcal{L} -estructuras, queremos encontrar homomorfismos de \mathbf{A} en \mathbf{B} resolviendo una instancia de CSP. Tomamos

$$X = \{X_i : i \in A\}$$

$$D_i = B$$

$$C = C^R \cup C^f$$

donde C^R es tal que para cada $R \in \mathcal{L}$ n -aria, para cada $(a_1, \dots, a_n) \in R^{\mathbf{A}}$, se da que $((X_{a_1}, \dots, X_{a_n}), R^{\mathbf{B}}) \in C^R$ y C^f es tal que para cada $f \in \mathcal{L}$ n -aria, para cada $(a_1, \dots, a_n, a') \in \text{graph}(f^{\mathbf{A}})$, se da que $((X_{a_1}, \dots, X_{a_n}, X_{a'}), \text{graph}(f^{\mathbf{B}})) \in C^f$.

Supongamos que $V : X \rightarrow B$ es una valuación solución de (X, D, C) , entonces el homomorfismo γ determinado por V será $\gamma(a) = V(X_a)$.

Si además quisiéramos que el homomorfismo fuera inyectivo, bastaría con agregar una restricción:

$$((X_1, \dots, X_m), \text{para todo } (i, j) \text{ con } i \neq j \text{ y } i, j \in \{1, \dots, m\} \text{ se da } X_i \neq X_j)$$

donde X_1, \dots, X_m son todas las variables en X .

Para que fuera sobreyectivo bastaría agregar la siguiente restricción:

$$\left((X_1, \dots, X_m), \bigcup_{i=1}^m \{X_i\} = B \right)$$

En el caso particular de la búsqueda de automorfismos, el lema 34 nos permite simplificar la búsqueda, encontrando endomorfismos inyectivos.

Lema 34. *Sea \mathbf{A} una estructura finita, entonces todo endomorfismo inyectivo de \mathbf{A} es un automorfismo de \mathbf{A} .*

Demostración. Sea γ un endomorfismo inyectivo de \mathbf{A} . Como A es finito tenemos que γ es sobre y además $\gamma^{-1} = \gamma^k$ para algún $k \geq 1$. \square

3.9. Generación de subestructuras

Dada una \mathcal{L} -estructura \mathbf{A} , generamos sus subestructuras recorriendo $\mathcal{P}(A)$, desde la mayor a la menor cardinalidad filtrando aquellos $A_0 \in \mathcal{P}(A)$ que no son cerrados bajo \mathcal{L} .

Además cuando detectamos que una subestructura \mathbf{A}_0 ya tiene representante en \mathcal{S} durante nuestros algoritmos, disminuimos los subconjuntos a chequear, revisando solo $\mathcal{P}(A) - \mathcal{P}(A_0)$, ya que si \mathbf{A}_0 tiene representante en \mathcal{S} , entonces cada subestructura de \mathbf{A}_0 tiene representante en \mathcal{S} .

Capítulo 4

Álgebras de Lindenbaum

4.1. Definiciones e ideas básicas

Sea \mathbf{A} una \mathcal{L} -estructura, y sea

$$\Sigma \in \{\text{Fo}(\mathcal{L}), \text{E}(\mathcal{L}), \text{E}^+(\mathcal{L}), \text{Op}(\mathcal{L}), \text{Op}^+(\mathcal{L})\}.$$

Notar que la colección de relaciones n -arias definibles en \mathbf{A} por fórmulas en Σ es cerrada bajo uniones e intersecciones (y también bajo complementación cuando $\Sigma \in \{\text{Fo}(\mathcal{L}), \text{Op}(\mathcal{L})\}$). Llamaremos *Álgebra de Lindenbaum* (de relaciones n -arias definibles por Σ en \mathbf{A}) al reticulado distributivo (o álgebra de Boole) resultante. Utilizaremos la siguiente notación:

- $\mathbf{Fo}_n(\mathbf{A})$ = Álgebra de Boole de relaciones n -arias definibles en \mathbf{A} ,
- $\mathbf{E}_n(\mathbf{A})$ = Reticulado de relaciones n -arias definibles por existenciales en \mathbf{A} ,
- $\mathbf{E}_n^+(\mathbf{A})$ = Reticulado de relaciones n -arias definibles por existenciales positivas en \mathbf{A} ,
- $\mathbf{Op}_n(\mathbf{A})$ = Álgebra de Boole de relaciones n -arias definibles por abiertas en \mathbf{A} ,
- $\mathbf{Op}_n^+(\mathbf{A})$ = Reticulado de relaciones n -arias definibles por abiertas positivas en \mathbf{A}

Definimos:

$$\text{sub hom}(\mathbf{A}) = \{\gamma : \gamma \text{ homomorfismo de } \mathbf{A}_0 \text{ en } \mathbf{B}_0 \text{ con } \mathbf{A}_0, \mathbf{B}_0 \in \mathbb{S}(\mathbf{A})\}$$

$$\text{sub iso}(\mathbf{A}) = \{\gamma : \gamma \text{ isomorfismo de } \mathbf{A}_0 \text{ en } \mathbf{B}_0 \text{ con } \mathbf{A}_0, \mathbf{B}_0 \in \mathbb{S}(\mathbf{A})\}$$

Dado que estas álgebras podrían ser muy grandes utilizamos los siguientes resultados para representarlas mediante un subconjunto de sus elementos.

Teorema 35 (Teorema de representación de Stone). *Toda álgebra de Boole finita \mathbf{A} es isomorfa al álgebra definida por $\mathcal{P}(At)$ donde $At \subseteq A$ es el conjunto de átomos en \mathbf{A} .*

Teorema 36 (Teorema de representación de Birkhoff). *Todo reticulado distributivo finito \mathbf{L} es isomorfo al reticulado de conjuntos descendientes del poset de elementos join-irreducibles en \mathbf{L} .*

Lema 37. *Dada un álgebra de Boole finita \mathbf{A} , un elemento es un átomo sii es join-irreducible.*

Los resultados anteriores nos permiten centrarnos únicamente en los elementos join-irreducibles de estas álgebras. Mientras que los Lemas 38 y 39, presentados a continuación, sugieren una manera clara de calcularlos.

Lema 38. *Una relación $r \subseteq A^n$ es join-irreducible en $\mathbf{Op}(\mathbf{A})$ sii hay $\bar{a} \in A^n$ tal que $r = \{h(\bar{a}) : h \in \text{sub iso}(\mathbf{A})\}$.*

Demostración. Supongamos que $r \subseteq A^n$ es join-irreducible en $\mathbf{Op}(\mathbf{A})$. Como r es cerrado bajo $\text{sub iso}(\mathbf{A})$ por pertenecer a $\mathbf{Op}(\mathbf{A})$, es claro que

$$r = \bigcup_{\bar{x} \in r} \{h(\bar{x}) : h \in \text{sub iso}(\mathbf{A})\}$$

, pero como r es join-irreducible hay una $\bar{a} \in r$ tal que $r = \{h(\bar{a}) : h \in \text{sub iso}(\mathbf{A})\}$.

Supongamos que $r = \{h(\bar{a}) : h \in \text{sub iso}(\mathbf{A})\}$ para algún $\bar{a} \in A^n$. Sean $r_1, \dots, r_m \in \mathbf{Op}(\mathbf{A})$ tales que

$$r = r_1 \cup \dots \cup r_m$$

, es claro que hay un r_j tal que $\bar{a} \in r_j$. Además como r_j es cerrado bajo $\text{sub iso}(\mathbf{A})$ por pertenecer a $\mathbf{Op}(\mathbf{A})$, entonces $r = r_j$. \square

Lema 39. *Una relación $r \subseteq A^n$ es join-irreducible en $\mathbf{Op}^+(\mathbf{A})$ sii hay $\bar{a} \in A^n$ tal que $r = \{h(\bar{a}) : h \in \text{sub hom}(\mathbf{A})\}$.*

Demostración. Igual a la del Lema 38 \square

4.2. Algoritmos

Con la intención de simplificar la exposición, supondremos que \mathcal{K} tiene una única estructura \mathbf{A} . La generalización es fácilmente deducible ya que los algoritmos son casi iguales a los presentados en el Capítulo 3.

Como un álgebra de Lindenbaum de relaciones definibles puede llegar a ser un objeto muy grande y difícil de manipular, nos basta generar los elementos join-irreducibles del álgebra según los Teoremas 35 para aquellas álgebras de Lindenbaum que son álgebras de Boole y los átomos (que también son los elementos join-irreducibles según el Lema 37) según el Teorema 36, para aquellas que son reticulados.

Para generar los elementos join-irreducibles del álgebra de Lindenbaum de relaciones de ancho n definibles por abiertas en \mathbf{A} basta con obtener el conjunto \mathcal{F} de flechas que se chequean por preservación en los algoritmos del Capítulo 3 que genera $\text{sub iso}(\mathbf{A})$ por el Teorema 28, y luego calcular $\{h(\bar{a}) : h \in \text{sub iso}(\mathbf{A})\}$ para cada $\bar{a} \in A^n$, ya que por el Lema 38, todos los elementos join-irreducibles son de esta forma. De la misma manera podemos construir el álgebra de Lindenbaum de relaciones de ancho n definibles por abiertas positivas en \mathbf{A} , basándonos en el Teorema 31 y en el Lema 39.

Los algoritmos 4.1 y 4.2, generan el conjunto \mathcal{F} definidos por los Teoremas 28 y 31.

Algorithm 4.1 Construcción de $\mathbf{Op}_n(\mathbf{A})$

```

1:  $\mathcal{S} = \emptyset$ 
2:  $\mathcal{F} = \emptyset$ 
3:  $Sub = [\mathbf{C} : \mathbf{C} \in \mathbb{S}_{\mathcal{L}}(\mathbf{A}), \text{ de la mayor a menor cardinalidad}]$ 
4: for  $\mathbf{B} \in Sub$  do
5:    $iso = \text{BUSCAISO}(\mathbf{B}, \mathcal{S})$ 
6:   if  $iso \neq \text{Null}$  then
7:      $\mathcal{F} = \mathcal{F} \cup \{iso\}$ 
8:      $Sub = [\mathbf{C} : \mathbf{C} \in Sub \text{ tal que } C \not\subseteq B]$ 
9:   else
10:     $\mathcal{S} = \mathcal{S} \cup \{\mathbf{B}\}$ 
11:    for  $\alpha \in \text{Aut}_{\mathcal{L}}(\mathbf{B})$  do
12:       $\mathcal{F} = \mathcal{F} \cup \{\alpha\}$ 
13:    end for
14:  end if
15: end for
16: return  $\mathcal{F}$   $\triangleright R$  es abierta-definible

17: function  $\text{BUSCAISO}(\mathbf{A}_0, \mathcal{S})$ 
18:   for  $\mathbf{S} \in \mathcal{S}$ , con  $|\mathbf{A}_0| = |\mathbf{S}|$  do
19:     if hay  $\gamma : \mathbf{A}_0 \rightarrow \mathbf{S}$  isomorfismo then
20:       return  $\gamma$   $\triangleright \gamma$  es isomorfismo con el representante
21:     end if
22:   end for
23:   return Null  $\triangleright$  No tiene representante
24: end function

```

Algorithm 4.2 Construcción de $\mathbf{Op}_n^+(\mathbf{A})$

```

1:  $\mathcal{S} = \emptyset$ 
2:  $\mathcal{F} = \emptyset$ 
3:  $Sub = [\mathbf{C} : \mathbf{C} \in \mathbb{S}_{\mathcal{L}}(\mathbf{A}), \text{ de la mayor a menor cardinalidad}]$ 
4: for  $\mathbf{B} \in Sub$  do
5:    $(iso, \mathcal{F}_0) = \text{BUSCAISO\_BIHOMOS}(\mathbf{B}, \mathcal{S})$ 
6:    $\mathcal{F} = \mathcal{F} \cup \mathcal{F}_0$ 
7:   if  $iso \neq \text{null}$  then
8:      $\mathcal{F} = \mathcal{F} \cup \{iso\}$ 
9:      $Sub = [\mathbf{C} : \mathbf{C} \in Sub \text{ tal que } C \not\subseteq B]$ 
10:  else
11:     $\mathcal{S} = \mathcal{S} \cup \{\mathbf{B}\}$ 
12:    for  $\alpha \in \text{Aut}_{\mathcal{L}}(\mathbf{B})$  do
13:       $\mathcal{F} = \mathcal{F} \cup \{\alpha\}$ 
14:    end for
15:  end if
16: end for
17: for  $\mathbf{A} \in \mathcal{S}$  do
18:   for  $\mathbf{B} \in \mathcal{S}$ , con  $|\mathbf{B}| < |\mathbf{A}|$  do
19:    for  $\gamma : \mathbf{A} \rightarrow \mathbf{B}$  con  $\gamma$  homomorfismo sobreyectivo do
20:       $\mathcal{F} = \mathcal{F} \cup \{\gamma\}$ 
21:    end for
22:  end for
23: end for
24: return  $\mathcal{F}$   $\triangleright R$  es definible por una abierta positiva

25: function  $\text{BUSCAISO\_BIHOMOS}(\mathbf{A}_0, \mathcal{S})$ 
26:    $\mathcal{F}_0 = \emptyset$ 
27:   for  $\mathbf{S} \in \mathcal{S}$ , con  $|\mathbf{A}_0| = |\mathbf{S}|$  do
28:      $bihomo = \text{False}$ 
29:     for  $\gamma : \mathbf{A}_0 \rightarrow \mathbf{S}$  con  $\gamma$  homomorfismo biyectivo do
30:        $bihomo = \text{True}$ 
31:       if  $\gamma$  es un  $\mathcal{L}$ -isomorfismo, con  $\mathbf{S} \in \mathcal{S}$  then
32:         return  $(\gamma, \emptyset)$   $\triangleright \gamma$  es isomorfismo con el representante
33:       else
34:          $\mathcal{F}_0 = \mathcal{F}_0 \cup \{\gamma\}$ 
35:       end if
36:     end for
37:     if  $\neg bihomo$  then
38:       for  $\gamma : \mathbf{S} \rightarrow \mathbf{A}_0$  con  $\gamma$  homomorfismo biyectivo do
39:          $\mathcal{F}_0 = \mathcal{F}_0 \cup \{\gamma\}$ 
40:       end for
41:     end if
42:     return  $(\text{Null}, \mathcal{F}_0)$   $\triangleright$  No tiene representante
43:   end for
44: end function

```

Una vez calculado \mathcal{F} basta con calcular $\{h(\bar{a}) : h \in \mathcal{F}\}$ para cada $\bar{a} \in A^n$, ya que \mathcal{F} genera al conjunto correspondiente al tipo de definibilidad. Esto lo

hacemos con el Algoritmo 4.3, lo cual ya nos da los elementos join-irreducibles o los átomos, según el tipo de definibilidad en cuestión.

Algorithm 4.3 Saturación por un conjunto de flechas

```

1:  $\mathcal{A} = \emptyset$ 
2: for  $\bar{a} \in A^n$  do
3:    $\mathcal{A} = \mathcal{A} \cup \{\text{CLAUSURA}(\bar{a}, \mathcal{E})\}$ 
4: end for
5: return  $\mathcal{A}$   $\triangleright R$  es abierta-definible

6: function  $\text{CLAUSURA}(\bar{a}, \mathcal{E})$ 
7:    $e = \{\bar{a}\}$ 
8:   while  $e$  crezca do
9:     for  $\bar{a} \in e$  do
10:      for  $\gamma \in \mathcal{E}$  do
11:         $e = e \cup \{\gamma(\bar{b})\}$ 
12:      end for
13:    end for
14:  end while
15:  return  $e$ 
16: end function

```

Para los demás tipos de definibilidad, el procedimiento sería análogo. Generar \mathcal{F} basados en los algoritmos del capítulo 3 y saturar según el Algoritmo 4.3.

4.3. Ejemplos y aplicación

En esta sección mostramos como la utilización del paquete permitió conjeturar una caracterización de las relaciones binarias definibles por existenciales positivas en reticulados distributivos.

4.3.1. Utilizando el paquete Definability para SageMath

A manera de prueba de las herramientas utilizamos Definability para recorrer todas las relaciones binarias definibles en el reticulado $\mathbf{2} \times \mathbf{2}$, representado en la figura 4.3.1.



Figura 4.3.1: Reticulado $\mathbf{2} \times \mathbf{2}$

Al correr el algoritmo de generación de los elementos join-irreducibles de $\mathbf{E}_2(\mathbf{2} \times \mathbf{2})$, Definability nos devolvió las siguientes relaciones:

Figura 4.3.2: Reticulado $\mathbf{E}_2(2 \times 2)$

- $\{(0, 0)\}$
- $\{(0, 1)\}$
- $\{(0, 2), (0, 3)\}$
- $\{(1, 0)\}$
- $\{(1, 1)\}$
- $\{(1, 2), (1, 3)\}$
- $\{(2, 0), (3, 0)\}$
- $\{(2, 1), (3, 1)\}$
- $\{(2, 2), (3, 3)\}$
- $\{(2, 3), (3, 2)\}$

Aquí se puede notar las ventajas de utilizar el Teorema 36, ya que el reticulado descrito por estos 10 elementos (representado en la figura ??) tiene cardinalidad $2^{10} = 1024$, ya que son todas las uniones entre elementos join-irreducibles.

En cuanto a los elementos join-irreducibles de $\mathbf{E}_2^+(2 \times 2)$ nos devolvió:

- $\{(0, 0), (1, 1), (2, 2), (3, 3)\}$
- $\{(0, 0), (0, 1), (0, 2), (0, 3), (1, 1), (2, 1), (2, 2), (3, 1), (3, 3)\}$
- $\{(0, 0), (1, 0), (1, 1), (1, 2), (1, 3), (2, 0), (2, 2), (3, 0), (3, 3)\}$
- $\{(0, 0), (0, 1), (0, 2), (0, 3), \dots, (3, 0), (3, 1), (3, 2), (3, 3)\}$

Las cuales pueden ser interpretadas respectivamente como:

- Δ
- $\{(x, y) : x \leq y\}$

Figura 4.3.4: Reticulado 3×3

- $\{(x, y) : x \geq y\}$
- A^2

Lo que mediante el Teorema 36, define un subreticulado de $\mathbf{E}_2(2 \times 2)$ anterior, mucho mas pequeño, como se puede ver en la Figura 4.3.3, donde los nodos sin rellenar son los elementos join-irreducibles devueltos por el paquete.

Figura 4.3.3: Reticulado de relaciones binarias definibles por existenciales positivas en 2×2

Luego probamos en el reticulado 3×3 representado en la figura 4.3.4.

Al interpretar los elementos join-irreducibles de $\mathbf{E}_2^+(3 \times 3)$ calculados por el paquete, nos dio nuevamente:

- Δ
- $\{(x, y) : x \leq y\}$
- $\{(x, y) : x \geq y\}$
- A^2

Lo que representa un reticulado isomorfo al presentado en 4.3.3. Esto nos llevo a conjeturar el Teorema 43, que probamos a continuación.

4.3.2. Caracterización de las relaciones binarias definibles por existenciales positivas en reticulados distributivos

Lema 40. Sea \mathbf{L} un reticulado y R una relación binaria no vacía sobre L preservada por endomorfismos en \mathbf{L} . Entonces vale que $\Delta^L \subseteq R$.

Demostración. Esto es una consecuencia directa de que para cada $a \in L$ la función de L en L que vale constantemente a es un endomorfismo. \square

Teorema 41 (Teorema del filtro primo). Sea $\langle L, \vee, \wedge \rangle$ un reticulado distributivo y F un filtro. Supongamos $x \in L - F$. Entonces hay un filtro primo P tal que $x \notin P$ y $F \subseteq P$.

Lema 42. Sea \mathbf{L} un reticulado distributivo y R una relación binaria no vacía sobre L preservada por endomorfismos en \mathbf{L} . Si hay $(a, b) \in R$ tal que $a \not\leq b$ (respectivamente $b \not\leq a$), entonces $\{(x, y) : x \geq y\} \subseteq R$ (respectivamente $\{(x, y) : x \leq y\} \subseteq R$).

Demostración. Fijamos $(a, b) \in R$ tal que $a \not\leq b$, y sean $c, d \in L$ tales que $c \geq d$. Veremos que $(c, d) \in R$. Por el Teorema 41 hay un filtro primo P que contiene al filtro generado por a y además $b \notin P$. Definimos $h : L \rightarrow L$ por

$$h(x) = \begin{cases} c & \text{si } x \in P, \\ d & \text{si } x \notin P. \end{cases}$$

Es fácil ver que h es un endomorfismo. Finalmente como $(a, b) \in R$ y h preserva R , $(h(a), h(b)) = (c, d) \in R$. \square

Teorema 43. Sea \mathbf{L} un reticulado distributivo y R una relación binaria sobre L definible por un fórmula existencial positiva en \mathbf{L} . Se da una de las siguientes:

- $R = \Delta$,
- $R = \{(x, y) : x \leq y\}$,
- $R = \{(x, y) : x \geq y\}$,
- $R = \{(x, y) : x \leq y\} \cup \{(x, y) : x \geq y\}$,
- $R = L \times L$.

Demostración. Si $R \subseteq \Delta$, por el Lema 40 $R = \Delta$.

Si $R \subseteq \{(x, y) : x \leq y\}$, pero $R \not\subseteq \Delta$, hay $(a, b) \in R$ tales que $a < b$, entonces $a \not\leq b$ y por el Lema 42 $R = \{(x, y) : x \leq y\}$. Análogo para $R \subseteq \{(x, y) : x \geq y\}$.

Si $R \not\subseteq \{(x, y) : x \leq y\}$ y $R \not\subseteq \{(x, y) : x \geq y\}$, aplicando dos veces el Lema 42 $\{(x, y) : x \leq y\} \cup \{(x, y) : x \geq y\} \subseteq R$. Si además $(a, b) \in R$ pero $a \not\leq b$ y $a \not\geq b$, tomo $(c, d) \in R$, si son comparables ya esta están en R por lo anterior. Si son incomparables aplico dos veces el Teorema 41 y tomo dos filtros primos P y Q tales que $a \in P$ pero $b \notin P$ y $b \in Q$ pero $a \notin Q$. Ahora definimos la siguiente función

$$h(x) = \begin{cases} c \vee d & \text{si } x \in P \cup Q \\ c & \text{si } x \in P - Q \\ d & \text{si } x \in Q - P \\ c \wedge d & \text{si } x \notin P \text{ y } x \notin Q \end{cases}$$

que fácilmente puede verse que es un endomorfismo. Por lo tanto como $(a, b) \in R$ y h preserva R , $(h(a), h(b)) = (c, d) \in R$. Por lo tanto $R = L \times L$. \square

Capítulo 5

Documentación de Definability

5.1. Introducción

Definability es un paquete que desarrollamos para SageMath, un software matemático licenciado bajo la GPL. Nuestro paquete implementa los algoritmos vistos en las secciones anteriores, permitiendo, dada una clase de estructuras, decidir definibilidad o generar las álgebras de Lindenbaum de relaciones definibles.

5.2. Arquitectura del paquete

Nuestro paquete se basa en parte en las librerías desarrolladas por Peter Jipsen para utilizar Minion, Universal Álgebra Calculator, Mace4 y Prover9 desde SageMath que se pueden obtener en <http://math.chapman.edu/~jipsen/sagepkg/>.

El paquete se desarrolla en <https://github.com/pablogventura/definability>.

5.2.1. Programas externos necesarios

Definability, así como SageMath han sido desarrollados en Python 2.7, por lo que necesitan del interprete instalado. Dado que Definability es un paquete desarrollado para SageMath, necesita de una instalación de este. Además, utiliza interfaces con otros programas. Particularmente utiliza Minion, y LADR (interfaz de línea de comandos de Prover9, Mace4, y otros programas). Definability ha sido probado en versiones particulares de estos programas y no se sabe si funcionara en otras. Las versiones utilizadas son las siguientes:

- Python 2.7.9,
- SageMath 6.7,
- Minion 1.8,
- LADR versión de noviembre de 2009.

En la siguiente sección explicaremos nuestra motivación detrás de desarrollar un paquete para SageMath y en las siguientes dos secciones explicaremos brevemente qué hacen los programas Minion y LADR.

5.2.1.1. SageMath

SageMath es un software matemático libre licenciado bajo la GPL, con el objetivo de ser una alternativa libre a Magma, Maple, Mathematica y Matlab.

SageMath es un software en pleno desarrollo que incluye herramientas para trabajar con reticulados y muchas otras estructuras algebraicas por lo que nos pareció interesante utilizarlo como base para nuestra herramienta.

5.2.1.2. Minion

Minion es un solucionador de CSP, rápido y escalable respecto del incremento del tamaño del problema. Es un solucionador de propósito general con un lenguaje de entrada muy expresivo. Minion es software libre, licenciado bajo la GPL 2.

En nuestro caso utilizamos la interfaz con Minion a la hora de resolver nuestros CSP para buscar morfismos.

5.2.1.3. LADR

LADR es la interfaz de línea de comandos para Prober9 y Mace4. Prover9 es un probador automático de teoremas para primer orden y lógica ecuacional y Mace4 es un software capaz de encontrar modelos finitos y contraejemplos.

En nuestra herramienta solo se utiliza la interfaz con Mace4. Esto es para generar automáticamente modelos de teorías de primer orden.

5.2.2. Módulos y organización del código fuente

Cada archivo .py en la carpeta src implementa los siguientes módulos:

config Maneja las rutas para acceder a Minion y a LADR.

minion Implementa la interfaz con Minion.

mace4 Implementa la interfaz con LADR, para utilizar Mace4.

fotype Implementa los tipos de primer orden

model Implementa la clase FO_Model que representa las estructuras de primer orden.

fotheory Implementa las teorías de primer orden.

functions Implementa la clase general de las funciones, de las que heredan los morfismos y las funciones y relaciones de primer orden.

fofunctions Implementa las funciones y relaciones de primer orden.

morphisms Implementa los morfismos.

constellation Implementa la clase `Constellation` que representa a la clase de estructuras \mathcal{K} , y contiene las implementaciones de los algoritmos vistos en el Capítulo 3.

lindenbaum Implementa los algoritmos de generación de álgebras de Lindenbaum presentados en el Capítulo 4.

examples Incluye algunos ejemplos de estructuras, y de tipos de primer orden utilizados fundamentalmente para testing.

fotheories Incluye varias teorías de primer orden, para generar estructuras automáticamente mediante Mace4.

sage_to Implementa entre conversión de estructuras algebraicas de Sage a nuestra clase `FO_Model`.

5.3. Instalación

5.3.1. Instalación del paquete en SageMath

Suponiendo que tenemos instalado SageMath en la ruta absoluta `SAGEDIR`, nos movemos al directorio donde se quiere descargar el paquete `Definability`, seguimos los siguientes comandos para instalar el paquete en nuestra instalación de SageMath:

```
$ git clone https://github.com/pablogventura/
  definability.git
$ cd tesis/package
$ ./make_spkg.sh
$ mv definability-hash.spkg SAGEDIR
$ cd SAGEDIR
$ ./sage -i definability-hash.spkg
```

5.4. Uso del paquete

Una vez instalado el paquete basta con arrancar Sage con el comando:

```
$ ./sage
```

y una vez en la consola de Sage, importamos el paquete con el comando:

```
import definability
```

5.5. Generación y entrada de estructuras

Al momento de ingresar estructuras para su chequeo puede optar por la generación automática de estructuras a partir de una teoría de primer orden mediante la interfaz a Mace4, o la entrada de una estructura en particular manualmente.

5.5.1. Generación de estructuras de una teoría de primer orden

Para definir un objeto del tipo `FO_Theory` que implementa a una teoría de primer orden basta con un nombre, una descripción, y una lista de axiomas en la sintaxis propia de Mace4. Por ejemplo para definir la teoría de reticulados bastaría con la siguiente línea:

```
Lat = definability.FO_Theory("Lat",
                              "Lattices",
                              ['(x v y) v z = x v (y v
                                z)',
                               'x v y = y v x',
                               '(x^y)^z = x^(y^z)',
                               'x^y = y^x',
                               '(x v y)^x = x',
                               '(x^y) v x = x'])
```

Ahora para recorrer los reticulados (filtrando isomorfismos) basta con llamar al método `find_models` con la cardinalidad buscada, el cual devuelve un generador. Por ejemplo para recorrer los reticulados de 5 elementos basta con

```
list(Lat.find_models(5))
```

que devolverá una lista con todos los reticulados de 5 elementos filtrando isomorfismos.

Nuestro paquete incluye una gran variedad de teorías de primer orden ya cargadas en el módulo `fotheories`. Por ejemplo: `Lat` para reticulados, `Graph` para grafos, `Grp` para grupos, etc. La lista completa se puede revisar mediante listar el módulo `fotheories` de nuestro paquete con el siguiente comando:

```
dir(definability.fotheories)
```

5.5.2. Entrada manual de una estructura

Supongamos que queremos ingresar manualmente el reticulado 2×2 . Lo primero es definir el tipo de primer orden al que pertenece. Para crear un objeto de la clase `FO_Type` basta con pasar dos diccionarios, el primero para las funciones y el segundo para las relaciones. Los diccionarios contienen las aridades y se indexan por los símbolos correspondientes. Por ejemplo, para definir el tipo de los reticulados, junto con una relación unaria P para chequear:

```
tlat = definability.FO_Type({'v': 2, '^': 2}, {"P": 1})
```

Luego se debe definir un universo para la estructura, como por ejemplo

```
universe = [0,1,2,3]
```

Luego se necesitan definir las funciones supremo e ínfimo, y una relación unaria P para su chequeo.

Una función, implementada por la clase `FO_Operation` se puede definir mediante un diccionario con claves de tuplas que apuntan al valor de la función, o en el caso de funciones binarias, una matriz A donde el valor $A_{i,j}$ se corresponde a la función evaluada en (i,j) . Para definir una relación utilizando la clase

FO_Relation basta con una lista de las tuplas que pertenecen a la relación y el universo sobre el que la relación esta definida.

Como ejemplo, definimos el ínfimo y el supremo utilizando cada una de las maneras para definir funciones, y definimos la relación P :

```
meet = definability.FO_Operation({(0, 0): 0,
                                   (0, 1): 0,
                                   (0, 2): 0,
                                   (0, 3): 0,
                                   (1, 0): 0,
                                   (1, 1): 1,
                                   (1, 2): 2,
                                   (1, 3): 3,
                                   (2, 0): 0,
                                   (2, 1): 2,
                                   (2, 2): 2,
                                   (2, 3): 0,
                                   (3, 0): 0,
                                   (3, 1): 3,
                                   (3, 2): 0,
                                   (3, 3): 3})
join = definability.FO_Operation([[0, 1, 2, 3],
                                   [1, 1, 1, 1],
                                   [2, 1, 2, 1],
                                   [3, 1, 1, 3]])
rP = definability.FO_Relation([(2, ), (3, )], uni)
```

Finalmente, para definir el reticulado basta con hacer:

```
l2x2 = definability.FO_Model(tlat, uni, {'^': meet,
                                          'v': join},
                               {'P': rP})
```

5.6. Chequeo de definibilidad

Una vez definidas las estructuras, para chequear definibilidad se utiliza un objeto de la clase Constellation, pasando una lista de las estructuras en \mathcal{K} . Esta clase implementa los algoritmos vistos en el Capítulo 3. Por ejemplo siguiendo con el ejemplo anterior:

```
c = definability.Constellation([l2x2])
```

Para chequear definibilidad están los siguientes métodos:

- is_existential_definable
- is_existential_positive_definable
- is_open_definable
- is_positive_open_definable

los cuales toman dos tipos $\mathcal{L} \subseteq \mathcal{L}'$ para decidir definibilidad de las relaciones del tipo $\mathcal{L}' - \mathcal{L}$ en \mathcal{L} , y devuelven una tupla con un booleano y un contraejemplo, si es que lo hay.

Por ejemplo para chequear definibilidad abierta de la relación P :

```
c.is_open_definable(tlat, tlat + definability.F0_Type
                    ({}, {"P": 1}))
```

Lo cual en particular es falso, por lo que el paquete devuelve la tupla:

```
(False, Embedding([0] -> 2,
                   F0_Type({'v': 2, '^': 2}, {}),
                   antitype= ['P'],
                   Injective, ))
```

Donde devuelve un contraejemplo, ya que el embedding del subreticulado formado por el elemento 0 en el subreticulado formado por el elemento 2 (un isomorfismo entre subestructuras de 2×2) no preserva P , negando el Teorema 15.

5.7. Generación de álgebras de Lindenbaum

Para la generación de las álgebras de Lindenbaum desarrolladas en el capítulo 4 el paquete dispone del modulo lindenbaum.

Este modulo contiene las siguientes funciones:

ji_of_existencial_definable_algebra para generar los elementos join-irreducibles de $\mathbf{E}_n(\mathcal{K})$

ji_of_existencial_positive_definable_algebra para generar los elementos join-irreducibles de $\mathbf{E}_n^+(\mathcal{K})$

atoms_of_open_definable_algebra para generar los átomos de $\mathbf{Op}_n(\mathcal{K})$

ji_of_open_positive_definable_algebra para generar los elementos join-irreducibles de $\mathbf{Op}_n^+(\mathcal{K})$

Estas funciones toman un objeto del tipo Constellation que determina \mathcal{K} , un tipo para los morfismos en cuestión y una aridad.

Por ejemplo para obtener los elementos join-irreducibles del reticulado de relaciones ternarias definibles en 2×2 bastaría con la siguiente línea:

```
definability.lindenbaum.
    ji_of_existencial_positive_definable_algebra(c,
    definability.examples.tiporet, 3)
```

Lo cual devuelve una lista de 22 relaciones join-irreducibles, que acortamos como ejemplo:

```
[Relation([0, 0, 0],
          [1, 1, 1],
          [2, 2, 2],
          [3, 3, 3]),
 .
```

```
.  
.  
Relation([0, 0, 0],  
          [0, 1, 1],  
          [0, 2, 2],  
          [0, 3, 3],  
          [1, 0, 0],  
          [1, 1, 1],  
          [1, 2, 2],  
          [1, 3, 3],  
          [2, 0, 0],  
          [2, 1, 1],  
          [2, 2, 2],  
          [2, 3, 3],  
          [3, 0, 0],  
          [3, 1, 1],  
          [3, 2, 2],  
          [3, 3, 3]])
```


Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

En este trabajo se hicieron los siguientes aportes:

- Se desarrollaron algoritmos para decidir definibilidad de relaciones en primer orden, en formulas abiertas, abiertas positivas, existenciales y existenciales positivas.
- Se desarrollaron los respectivos algoritmos para obtener el álgebra de relaciones definibles en cada fragmento de primer orden mencionado anteriormente.
- Se desarrolló una herramienta que implementa dichos algoritmos.
- Al probar la herramienta, se conjeturo y luego se probó una caracterización de las relaciones binarias definibles por existenciales positivas en reticulados distributivos (Teorema 43).

Dado lo inicial de la investigación, las conclusiones tienen que ver con la dirección futura de la investigación y las preguntas interesantes que han surgido durante este trabajo, que se encuentran en la siguiente sección.

Por otra parte, resulta muy alentador que el primer ejemplo visto seriamente haya permitido conjeturar el Teorema 43. Esperamos que sea una herramienta muy útil como asistente de investigación para el grupo de Semántica Algebraica.

6.2. Trabajo futuro

La experiencia durante el desarrollo nos ha hecho notar que la una gran parte de de los cálculos vienen por el calculo de morfismos. En relación a la incidencia que puedan tener propiedades de \mathcal{K} en la complejidad de decidir definibilidad es interesante destacar el extenso trabajo existente acerca de la complejidad computacional del CSP (ver, e.g., Creignou, Kolaitis y Vollmer 2008). Es de esperar que si el problema de calcular morfismos para la clase \mathcal{K} (y estructuras derivadas) está bien condicionado, esto tendrá un efecto positivo sobre el costo computacional de nuestro problema. Un artículo que merece mención especial es Idziak y col. 2010, en donde se demuestra que el cálculo de morfismos se puede

resolver en tiempo polinomial si se cuenta con la existencia de términos con ciertas propiedades ecuacionales (*cube terms*). Una situación particular donde se cuenta con cube terms es bajo la presencia de un *Near-Unanimity term*, como por ejemplo en el caso de estructuras que tienen un reducto de reticulado. Debido a esto resulta interesante el desarrollo de algoritmos que decidan definibilidad para este tipo de estructuras. Recientemente se ha demostrado que la presencia de cube terms es decidible Horowitz 2015, aunque, dada la naturaleza teórica de este resultado, no está claro si el algoritmo descubierto tiene la eficiencia suficiente para que resulte relevante a nuestros propósitos.

Queda pendiente explorar y aprovechar en la implementación, la interacción entre los diferentes tipos de definibilidad. Queda muy claro que hay una jerarquía entre ellas que podría ser aprovechada, al calcular diferentes tipos de definibilidad sobre la misma clase \mathcal{K} .

También esperamos explorar mejores implementaciones para la generación de subestructuras, investigando a fondo los algoritmos utilizados por el Universal Algebra Calculator.

En cuanto a nuestra implementación, pensamos que se podría mejorar portando algunas de las partes mas criticas a C.

Bibliografía

- Campercholi, Miguel y Diego Vaggione (2015). *Semantical conditions for the definability of functions and relations*. Aceptado para su publicación en Algebra Universalis. eprint: 1506.07501. URL: <http://www.arxiv.org/abs/1506.07501>.
- Creignou, Nadia, Phokion Kolaitis y Heribert Vollmer (2008). *Complexity of Constraints: An Overview of Current Research Themes*. Springer.
- Gent, Ian, Chris Jefferson y Ian Miguel (2006). “MINION: A Fast, Scalable, Constraint Solver”. En: *Proceedings of the 2006 Conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29 – September 1, 2006, Riva Del Garda, Italy*. IOS Press, págs. 98-102.
- Horowitz, Jonah (2015). “Testing for edge terms is decidable”. En: *Algebra universalis* 73.3-4, págs. 321-334.
- Idziak, Paweł y col. (2010). “Tractability and Learnability Arising from Algebras with Few Subpowers”. En: *SIAM Journal on Computing* 39.7, págs. 3023-3037.
- Russell, Stuart y Peter Norvig (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall.