ASPECTOS MATEMÁTICOS DE LA SEMÁNTICA DE LOS PROGRAMAS LÓGICOS NORMALES

María López Palmero

ASPECTOS MATEMÁTICOS DE LA SEMÁNTICA DE LOS PROGRAMAS LÓGICOS NORMALES

María López Palmero

Memoria presentada como parte de los requisitos para la obtención del título de Grado en Matemáticas por la Universidad de Sevilla.

Tutorizada por

Prof. Andrés Cordón Franco

Índice general

Resumen

El presente trabajo se enmarca en el campo de la Programación Lógica y, más concretamente, en el estudio de la semántica declarativa de los programas lógicos normales, colocando el enfoque principalmente en sus aspectos matemáticos. El objetivo general es capturar mediante definiciones formales la noción intuitiva de "modelo canónico natural" o "significado pretendido" de un programa lógico.

Nuestro punto de partida será el estudio de la semántica de punto fijo para programas definidos (programas sin negación). A continuación, estudiaremos programas lógicos normales generales, en los que se permite el uso de la conectiva not. Presentaremos un buen número de las distintas semánticas que se han propuesto para capturar el significado de la negación en Programación Lógica: la semántica de los modelos justificados, la semántica de los modelos estables, los modelos de Fitting 3-valorados y los modelos perfectos para programas localmente estratificados.

Prestaremos especial atención al estudio de las relaciones entre las diferentes semánticas presentadas.

Abstract

This present work is developed within the field of Logic Programming. More specifically, it deals with the study of the declarative semantic of normal logic programs, focusing principally on its mathematical aspects. The overall objective is to capture through formal definitions the intuitive notion of "natural canonical model" or "intended meaning" of a logic program.

Our starting point will be the study of the fixed point's semantic for definite programs (programs without negation). Then we will study general normal logic programs which can use the not connective. We will propose a variety of different semantics which have set out to capture the meaning of the negation in Logic Programming: the supported models' semantic, the stable models' semantic, the three-valued Fitting models and perfect models for locally stratified programs.

We will pay special attention to the study of the relations between the different proposed semantics.

1 Introducción

El objetivo principal del presente trabajo es el estudio de la semántica declarativa de los programas lógicos con negación y, por tanto, se enmarca dentro del campo de la teoría de la Programación Lógica.

En líneas muy generales, la Programación Lógica consiste en usar la lógica de primer orden como un lenguaje de representación del conocimiento para especificar un determinado problema y emplear como método de computación para resolver dicho problema la deducción de nuevo conocimiento a partir de la especificación dada.

Los fundamentos de la Programación Lógica se encuentran en un importante artículo de Robert Kowalski (Kowalski, 1974)[9] que, a su vez, se basa en las ideas de un conocido artículo de John Alan Robinson en el que se establecieron las bases de la deducción automatizada utilizando el principio de resolución (Robinson, 1965)[10]. Estas ideas dieron lugar, casi simultáneamente, al lenguaje de programación *Prolog*, dado a conocer por Alain Colmerauer y otros en Marsella en 1973. Prolog nació como un lenguaje de programación de uso general y fue dado a conocer por el siguiente eslogan de Kowalski:

ALGORITMO = LÓGICA + CONTROL

Lógica representa la descripción del problema que queremos resolver y **Control** el proceso de cómo debe resolverse. Desde entonces, la Programación Lógica se ha convertido en un paradigma bien establecido de programación y se ha desarrollado de numerosas maneras y en diversas direcciones, como en el procesamiento del lenguaje natural, la deducción automática (en el contexto del *model checking*), la representación declarativa del conocimiento y del razonamiento no monótono o el campo de la programación con conjuntos de respuestas (*answer set programming*), entre otras.

El principal objeto de estudio de la teoría de la Programación Lógica son los *programas lógicos*. Desde un punto de vista intuitivo, un programa lógico es un conjunto

de hechos y reglas, escritos en un fragmento restringido de la lógica de primer orden, que típicamente corresponde a la representación del conocimiento sobre un determinado escenario.

Vamos a ver una serie de ejemplos sencillos:

Ejemplo 1.1. Sea P el siguiente programa:

```
humano(juan)←
mortal(X)← humano(X)
```

Este ejemplo presenta un programa lógico P en el que se expresa el conocimiento siguiente: la regla "Si X es humano, X es mortal" y el hecho "Juan es humano". Entonces, del programa P puede deducirse el nuevo hecho "Juan es mortal". Por tanto, el *significado natural* del programa P puede capturarse mediante el modelo

$$M = \{humano(juan), mortal(juan)\}$$

Nótese que M es un modelo de Herbrand de P visto como un conjunto de cláusulas de la lógica de primer orden.

Ejemplo 1.2. Sea P el siguiente programa:

```
nodo(a) ←
nodo(b) ←
nodo(c) ←
arco(a,b) ←
arco(b,c) ←
camino(X,X) ← nodo(X)
camino(X,Y) ← arco(X,Z),camino(Z,Y)
```

P es un programa lógico que expresa la relación "hay un camino en el grafo entre los nodos X e Y". Volveremos a este ejemplo en el Capítulo 3. De nuevo, el significado natural del programa es claro y puede expresarse mediante el modelo
$M = \{nodo(a), nodo(b), nodo(c), arco(a, b), arco(b, c), camino(a, a), camino(b, b), camino(c, c), camino(a, b), camino(b, c), camino(a, c)\}$

Los dos ejemplos anteriores son *programas definidos*, esto es, programas lógicos donde no aparece la negación ¬. Para estos programas, no hay ambigüedad sobre cúal

debe ser su significado natural o modelo canónico y dicho modelo canónico puede ser fácilmente definido desde el punto de vista formal: el menor modelo de Herbrand de P visto como un conjunto de cláusulas de primer orden o, equivalentemente, el menor punto fijo de un cierto operador (el operador de consecuencia inmediata asociado T_P) que actúa sobre interpretaciones de Herbrand.

Sin embargo, la situación cambia completamente cuando tratamos con *programas lógicos normales*, esto es, programas donde se permite el uso de la negación ¬.

Ejemplo 1.3. Sea P el siguiente programa:

```
pajaro(a)←
pajaro(b)←
pinguino(b)←
vuela(X) ← pajaro(X),¬ pinguino(X)
```

En este caso, tanto
$M_1 = \{pajaro(a), pajaro(b), pinguino(b), vuela(a)\}$
como
$M_2 = \{pajaro(a), pajaro(b), pinguino(b), pinguino(a)\}$
son modeles de Herbrand de P. Observése que M_1 y M_2 son incomparables ($M_1 \not\subseteq M_2$ y $M_2 \not\subseteq M_1$) y ahora no existe el menor modelo de Herbrand del programa. ¿Cuál debe ser el significado natural de P? ¿Cómo puede caracterizarse formalmente dicho modelo canónico elegido?[1]

Ejemplo 1.4. Sea P el programa:

```
p(a) ← ¬ p(b)
p(b) ← ¬ p(a)
```

Tanto $M_1 = \{p(a)\}$ como $M_2 = \{p(b)\}$ son modelos de Herbrand de P. Pero, ¿cuál debe ser el significado natural de P? ¿Cómo puede resolverse la evidente simetría entre ambos modelos para elegir el modelo canónico de P?[2]

De hecho, ejemplos como el anterior ponen de manifiesto otro debate a la hora de elegir el significado natural de la negación en Programación Lógica, que algunos

[1] Veremos más adelante que en este caso el modelo M_1 será el elegido como modelo canónico de P.

[2] Veremos más adelante que en este caso la solución no es unánime y depende de la semántica de los programas lógicos normales que se adopte.

autores califican como el "cisma de la Programación Lógica". A la hora de elegir el significado natural de un programa lógico normal, ¿hemos de elegir necesariamente un *único* modelo canónico para P? ¿O es más adecuado identificar en algunos casos *varios modelos* de P como los modelos naturales del programa, sin privilegiar ninguno de ellos?

Veremos que no hay una solución única y que a lo largo de las últimas décadas se han propuesto distintas soluciones de ambos tipos para intentar capturar el significado de la negación en Programación Lógica. El objetivo del presente trabajo es estudiar de manera formal, y con especial énfasis en sus propiedades matemáticas, varias de estas semánticas declarativas propuestas para los programas lógicos normales.

La presente memoria está dividida en seis capítulos.

El primero de ellos es introductorio. En el segundo, **Conjuntos ordenados y Lógica**, recopilamos las definiciones y los resultados fundamentales de la lógica de primer orden y de la teoría de órdenes que usaremos en el trabajo, con especial énfasis en los teoremas de punto fijo. En este capítulo, está agrupada tanto la notación necesaria para entender bien el resto del trabajo como los resultados fundamentales que vamos a usar. Al final del capítulo, se introducen también dos temas relevantes para la Programación Lógica: las lógicas no clásicas multivaluadas y el espacio de valoraciones de verdad como conjunto parcialmente ordenado. Finalmente, hay un anexo sobre *ordinales y construcciones transfinitas*, el cual es necesario para entender las construcciones de puntos fijos de operadores asociados a un programa P.

En el Capítulo 3, **Modelos justificados y Modelos estables**, introducimos estas dos semánticas para los programas lógicos normales. Como punto de partida, estudiamos la semántica clásica de punto fijo para los programas definidos. Para los programas definidos, sí es posible de manera sencilla elegir un modelo o canónico del programa. Este es un resultado bien conocido y puede verse como el punto de partida de cualquier discusión sobre la semántica declarativa de los programas lógicos. En particular, nos permitirá introducir el importante operador de consecuencia inmediata T_P. Una vez entendido este caso fundamental, veremos con detalle cómo ampliar la semántica a un programa lógico normal con negación e introduciremos dos de las semánticas más estudiadas en el campo: los modelos justificados y los modelos estables. Esta última es especialmente relevante también desde el punto de vista de las aplicaciones y es la base teórica de la llamada *Programación con conjuntos de respuestas* (ver [2] para más información).

En el Capítulo 4, **Modelos de Fitting**, vamos a estudiar una nueva noción de se-

mántica de un programa normal que permite elegir un único modelo canónico. El concepto de modelo estable estudiado en el Capítulo 3 resulta satisfactorio pues, en muchos casos, captura nuestra intuición sobre el modelo natural de un programa lógico y, por otra parte, la noción de modelo estable tiene buenas propiedades matemáticas. Sin embargo, un programa normal puede tener más de un modelo estable, hecho que supone una debilidad a la vista de algunos autores. Sería deseable disponer de una semántica declarativa que nos permitiese asociar de manera natural a cada programa lógico normal un *único* modelo canónico. En el Capítulo 4, estudiaremos una nueva noción de semántica para elegir exactamente un único modelo: los llamados *modelos de Fitting* (también conocidos bajo el nombre de *semántica de Kripke-Kleene*). Para ello, haremos uso de la lógica fuerte 3-valorada de Kleene en lugar de la lógica clásica.

En el Capítulo 5, **Programas localmente estratificados. Modelos Perfectos**, se propone una manera alternativa de afrontar el problema de la ambigüedad en la semántica de los programas normales, sin tener que abandonar los modelos de Herbrand basados en la lógica clásica. En lugar de variar la noción de modelo para conseguir la unicidad como se hizo en el Capítulo 4, en este capítulo impondremos ciertas restricciones sintácticas a los programas lógicos de manera que un programa que satisfaga estas restricciones tenga asociado un único modelo canónico. Dichas condiciones sintácticas dan lugar a la importante clase de los *programas localmente estratificados*. Asociada a estos programas, introducimos también en este capítulo la semántica de los *modelos perfectos*. El principal resultado que presentaremos es el teorema que dice que todo programa lógico normal localmente estratificado posee un *único modelo estable*. Luego para esa clase de programas lógicos el principal punto débil de la semántica de los modelos estables desaparece.

Finalmente, el Capítulo 6 contiene las conclusiones a las que hemos llegado a lo largo de la realización del trabajo.

Las principales técnicas matemáticas novedosas para una graduada de la Facultad de Matemáticas de la Universidad de Sevilla que hemos tenido que desarrollar han sido las siguientes.

Desde el punto de vista lógico, en el trabajo, ha sido conveniente manejar más valores de verdad aparte de los dos convencionales, $TWO = \{\mathbf{f}, \mathbf{t}\}$. Es por eso que en el Capítulo 2, definimos la lógica fuerte de Kleene que emplea el conjunto de valores de verdad $THREE = \{\mathbf{u}, \mathbf{f}, \mathbf{t}\}$ (y donde $\mathbf{u}$ representa valor indeterminado) y también, la lógica 4-valorada de Belnap, cuyo conjunto de valores de verdad es $FOUR = \{\mathbf{u}, \mathbf{f}, \mathbf{t}, \mathbf{b}\}$ (donde $\mathbf{b}$ representa ambos o sobredefinido).

Ha sido necesario estudiar también el espacio de las valoraciones de verdad como un orden parcial completo y estudiar teoremas de punto fijo para operadores que actúan sobre estos espacios de valoraciones de verdad.

Finalmente, ha sido necesario estudiar un mínimo de conceptos y resultados sobre ordinales y construcciones transfinitas, ya que hemos necesitado de ellos en determinados casos al considerar iteraciones de un función monótona un número transfinito de veces.

Por último, es importante hacer notar que la principal referencia que hemos seguido para desarrollar el presente trabajo ha sido el libro [3], complementando y ampliando algunos temas concretos con otros trabajos que están recogidos en la bibliografía.

2 Conjuntos ordenados y Lógica

El estudio formal de la semántica de los programas lógicos tradicionalmente se basa en la combinación de elementos de la teoría de órdenes (con especial énfasis en los teoremas de punto fijo) y elementos de la lógica de primer orden.

En este primer capítulo, recopilamos las definiciones y los resultados fundamentales al respecto. El objetivo es doble: por una parte, agrupar los resultados básicos debería permitir una lectura más cómoda de la presente memoria y, por otra parte, este capítulo también nos servirá para introducir notación y terminología básica.

Además, cabe destacar que al final del capítulo trataremos dos temas especialmente relevantes para el estudio de la semántica de los programas lógicos: lógicas no clásicas multivaluadas y el espacio de valoraciones de verdad como conjunto ordenado.

2.1 Conjuntos Ordenados y Teoremas de Punto Fijo

En primer lugar, introducimos las definiciones básicas sobre la teoría de conjuntos ordenados que usaremos a lo largo de la memoria.

Definición 2.1 (Órdenes parciales). *Sea D un conjunto y sea R una relación binaria sobre D, esto es, R es un subconjunto del producto cartesiano $D \times D$. Diremos que R es una relación de orden parcial sobre D si se cumplen las siguientes condiciones:*

Reflexiva: *para todo $a \in D$, $(a, a) \in R$.*
Antisimétrica: *para todo $a, b \in D$, si $(a, b) \in R$ y $(b, a) \in R$, entonces $a = b$.*
Transitiva: *para todo $a, b, c \in D$, si $(a, b) \in R$ y $(b, c) \in R$, entonces $(a, c) \in R$.*

En este caso, diremos que el par (D, R), o bien solo D cuando se sobreentiende la relación R por el contexto, es un conjunto parcialmente ordenado.

Definición 2.2 (Órdenes totales). *Sea (D, R) un conjunto parcialmente ordenado.*

1. *Dos elementos $a, b \in D$ se dirán comparables si se tiene o bien $(a, b) \in R$ o bien $(b, a) \in R$. En otro caso, a y b se dirán incomparables.*
2. *Sea $A \subseteq D$ no vacío. Diremos que A está totalmente ordenado por R (o que A es una cadena) si dos elementos cualesquiera de A son comparables respecto a R.*
3. *(D, R), o bien solo D cuando se sobreentiende la relación R por el contexto, es un conjunto totalmente ordenado si D está totalmente ordenado por la relación R.*

Como es habitual y para hacer la notación más intuitiva, si R es una relación de orden sobre un conjunto D, escribiremos $R = \leq$ y usaremos notación infija. Por ejemplo, en lugar de $(a, b) \in R$ escribiremos $a \leq b$. Usaremos también $a < b$ para significar $a \leq b$ y $a \neq b$. Las notaciones $a \geq b$ y $a > b$ también se usarán con su significado habitual.

Definición 2.3 (Conjuntos dirigidos). *Sean $(D, \leq)$ un conjunto parcialmente ordenado y $A \subseteq D$. El conjunto A se dirá dirigido si es no vacío y, dados $a, b \in A$ cualesquiera, existe $c \in A$ tal que $a \leq c$ y $b \leq c$.*

Obsérvese que toda cadena es, en particular, un conjunto dirigido. Ahora bien, en un conjunto parcialmente ordenado que no es un orden total, no todo conjunto dirigido tiene por qué ser una cadena.

Definición 2.4. *Sean $(D, \leq)$ un conjunto parcialmente ordenado y $A \subseteq D$.*

1. *Un elemento $b \in D$ se denomina cota superior de A si tenemos que $a \leq b$ para todo $a \in A$.*
2. *Además, b se llama supremo de A (sup A o $\sqcup A$) si es cota superior de A y para todo b' que sea cota superior de A, se tiene que $b \leq b'$. (El supremo, si existe, es único por la propiedad de antisimetría).*
3. *Análogamente, podemos definir cota inferior y el ínfimo de A (inf A o $\sqcap A$).*
4. *Un elemento $a \in D$ se dirá elemento maximal de D si no existe $a' \in D$ tal que $a < a'$.*
5. *Un elemento $a \in D$ se dirá elemento minimal de D si no existe $a' \in D$ tal que $a' < a$.*

6. *Definimos el orden dual de $\leq$ en D, $\leq^d$, como la relación de orden dada por: $x \leq^d y$ si, y solo si, $y \leq x$. Nótese que cotas inferiores, ínfimos, ... en $\leq$ se corresponden con cotas superiores, supremos, ... en $\leq^d$.*

Ejemplo 2.1. Consideremos el conjunto de los números naturales, $\mathbb{N} = \{0, 1, 2, ...\}$, con el orden usual. Es claro que $\mathbb{N}$ es un conjunto totalmente ordenado, $\mathbb{N}$ no tiene cota superior y 0 es un elemento minimal de $\mathbb{N}$ que coincide con el ínfimo del conjunto. Puesto que el orden es total, todos los subconjuntos no vacíos de $\mathbb{N}$ son cadenas y, por tanto, conjuntos dirigidos.

Ejemplo 2.2. Consideremos el conjunto de los naturales positivos $\mathbb{N}^+ = \{1, 2, ...\}$ con el orden de la divisibilidad: $n < m$ si, y solo si, n divide a m. Entonces, $\mathbb{N}^+$ es un conjunto parcialmente ordenado que no es un orden total.

Sea A el conjunto de los números pares $A = \{2, 4, 6, ...\} \subset \mathbb{N}^+$. Es obvio que A es un conjunto dirigido ya que $A \neq \emptyset$ y, si tomamos dos elementos cualesquiera de A, por ejemplo 2 y 8, existe otro elemento $c \in A$ tal que $2 \leq c$ y $8 \leq c$, sea $c = 2 \cdot 8 = 16$ por ejemplo. Sin embargo, A no es una cadena ya que no es cierto que dos elementos cualesquiera sean comparables. Por ejemplo, 4 y 6 son incomparables ya que ni $4 \leq 6$ ni $6 \leq 4$.

Ejemplo 2.3. Sea X no vacío y tomemos el conjunto de las partes de X, $\mathcal{P}(X)$. Consideramos el orden $A < B$ si, y solo si, $A \subseteq B$ como subconjuntos de X. Entonces $(\mathcal{P}(X), \subseteq)$ es un conjunto parcialmente ordenado.

Definición 2.5. *Sea $(D, \leq)$ un conjunto parcialmente ordenado.*

1. *Un conjunto $A \subseteq D$ es una ω-cadena si A es una sucesión creciente de la forma $a_0 \leq a_1 \leq a_2 \leq \dots$*
2. *$(D, \leq)$ se dirá un orden parcial ω-completo (ω-cpo, por su nombre en inglés) si satisface las siguientes propiedades:*
   a) *Para cada ω-cadena A de D, existe el supremo de A en D.*
   b) *D tiene elemento mínimo. Es decir, D tiene un elemento distinguido, $\bot$, tal que $\bot \leq a$ para todo $a \in D$.*
3. *$(D, \leq)$ se dirá cadena-completo si toda cadena A de D tiene supremo en D.*
4. *$(D, \leq)$ es un orden parcial completo (o cpo, por su nombre en inglés) si satisface las siguientes propiedades:*
   a) *Para cada $A \subseteq D$ dirigido, existe el supremo de A en D.*
   b) *D tiene elemento mínimo, $\bot$.*

5. *$(D, \leq)$ es un semi-retículo superior completo si satisface las siguientes propiedades:*

   a) *Para cada $A \subseteq D$ dirigido, existe el supremo de A en D.*
   b) *Para todo $A \subseteq D$ no vacío, existe el ínfimo de A en D.*

6. *$(D, \leq)$ es un retículo completo si satisface las siguientes propiedades:*

   a) *Para todo $A \subseteq D$ cualesquiera, existe el supremo de A en D.*
   b) *Para todo $A \subseteq D$ cualesquiera, existe el ínfimo de A en D.*

Observación 2.1. Sea $(D, \leq)$ un orden parcial. Con las definiciones dadas, el supremo del conjunto vacío $\emptyset$ existe en D solo si D tiene elemento mínimo $\bot$ y el ínfimo de $\emptyset$ existe en D solo si D tiene elemento máximo $\top$. Por otra parte, se sigue de las definiciones anteriores que todo semi-retículo superior completo tiene elemento mínimo y todo retículo completo ha de poseer tanto elemento mínimo como elemento máximo.

Existe una relación inmediata entre las nociones definidas:

retículo completo $\Rightarrow$ semi-retículo superior completo $\Rightarrow$ orden parcial completo $\Rightarrow$ orden parcial cadena-completo con $\bot$ $\Rightarrow$ orden parcial ω-completo

La siguiente proposición, que enunciamos sin prueba, establece que una de estas flechas es, en realidad, una equivalencia.

Proposición 2.1. Un conjunto parcialmente ordenado $(D, \leq)$ es un orden parcial completo si y, solo si, D es cadena-completo y posee elemento mínimo.

Los conjuntos parciales ordenados, en general, y los órdenes parciales completos, en particular, desempeñan un papel muy relevante en el estudio formal de muchos aspectos de la Computación teórica. Sin embargo, para aplicaciones elaboradas, a veces es necesario que el orden parcial en cuestión posea más propiedades estructurales (aparte de la completitud o la ω-completitud ya vistas). A continuación introducimos una de esas nociones estructurales: los llamados *dominios de Scott*.

Definición 2.6. *Sea $(D, \leq)$ un conjunto parcialmente ordenado. Diremos que $a \in D$ es un elemento compacto o finito de D si siempre que $A \subseteq D$ es un conjunto dirigido y $a \leq \sup A$, se tiene que $a \leq x$ para algún $x \in A$. El conjunto de los elementos compactos de D se denota D_c.*

Definición 2.7. *Diremos que $(D, \leq)$ es un dominio de Scott, o simplemente un dominio, si satisface las siguientes propiedades.*

1. *$(D, \leq)$ es un orden parcial completo.*

2. *(Algebraicidad de D) Para cada $x \in D$, se tiene que el conjunto $approx(x) = \{a \in D_c | a \leq x\}$ es un conjunto dirigido y $x = sup\ approx(x)$.*
3. *(Completitud consistente de D) Si el conjunto $\{a, b\} \subseteq D_c$ es consistente (esto es, existe $x \in D$ tal que $a \leq x$ y $b \leq x$), entonces $sup\ \{a, b\}$ existe en D.*

Vamos a ver una serie de ejemplos sencillos para entender mejor estos conceptos.

Ejemplo 2.4.

1. Sea M el conjunto de las partes de $\mathbb{N}$, $M = \mathcal{P}(\mathbb{N})$, y consideremos el orden de la inclusión conjuntista dado en el ejemplo 2.3. El conjunto vacío es el elemento minimal de M que coincide con el ínfimo y, también vemos fácilmente que $\mathbb{N}$ es el elemento máximo de M. De hecho, M es un retículo completo y M es un dominio en el que los elementos compactos son los subconjuntos finitos de $\mathbb{N}$.
2. Sea X un conjunto no vacío y sea Y el conjunto de los pares de la forma (I^+, I^-) con $I^+, I^- \subseteq X$ y $I^+ \cap I^- = \emptyset$. Consideramos el orden en Y siguiente: $(I^+, I^-) \leq (J^+, J^-)$ si, y solo si, $I^+ \subseteq J^+$ y $I^- \subseteq J^-$. Entonces Y es un semi-retículo superior completo (y, por tanto, un orden parcial completo) pero no es un retículo completo. De hecho, Y carece de elemento máximo y los elementos maximales son los pares (I^+, I^-) tales que $I^+ \cup I^- = X$. El elemento mínimo de Y es $(\emptyset, \emptyset)$.
3. Sea D el conjunto de las funciones parciales $f : \mathbb{N}^n \to \mathbb{N}$ con el siguiente orden: $f \leq g$ si, y solo si $grafo(f) \subseteq grafo(g)$. D es orden parcial completo, el mínimo elemento es la función vacía y los elementos maximales de D son las funciones totales. D no es un retículo completo.

Pasamos ahora a estudiar funciones entre conjuntos ordenados y teoremas de punto fijo para funciones monótonas, que serán los principales resultados teóricos sobre conjuntos ordenados que necesitaremos en los capítulos siguientes para analizar la semántica de los programas lógicos. Empezamos con unas definiciones básicas.

Definición 2.8. *Sean $(D, \leq)$ y $(E, \leq)$ conjuntos parcialmente ordenados.*

1. *Una función $f : D \longrightarrow E$ se denomina monótona si para todo $a, b \in D$ con $a \leq b$, se tiene que $f(a) \leq f(b)$.*
   Una función $f : D \longrightarrow E$ se denomina anti-monótona si para todo $a, b \in D$ con $a \leq b$, se tiene que $f(b) \leq f(a)$.
2. *Si D y E son órdenes parciales ω-completos, entonces una función $f : D \longrightarrow E$ se denomina ω-continua si es monótona y $sup\ f(A) = f(sup\ A)$ para toda ω-cadena A de D.*

3. *Si D y E son órdenes parciales completos, una función $f : D \longrightarrow E$ se dirá continua si es monótona y $sup\ f(A) = f(sup\ A)$ para cada A subconjunto dirigido de D.*

Incluimos, sin prueba, la observación de que en la definición de función continua entre órdenes parciales completos, podemos reemplazar subconjuntos dirigidos por cadenas.

Proposición 2.2. Sean $(D, \leq)$ y $(E, \leq)$ órdenes parciales completos. Una función $f : D \longrightarrow E$ es continua si y solo si es monótona y para cada cadena A en D, se tiene que $\sup\ f(A) = f(supA)$.

Definición 2.9. *Sea D un conjunto parcialmente ordenado y $f : D \longrightarrow D$.*

1. *$x \in D$ es punto fijo de f si $f(x) = x$.*
2. *$x \in D$ es un punto prefijo de f si $f(x) \leq x$.*
3. *$x \in D$ es un punto postfijo de f si $x \leq f(x)$.*
4. *El menor punto fijo de f, denotado por $lfp(f)$ (por su nombre en inglés), es un punto fijo x de f que cumple: si y es otro punto fijo de f, entonces $x \leq y$.*
5. *Las nociones de menor punto prefijo y menor punto postfijo de f se definen de manera análoga.*

Para obtener puntos fijos de funciones monótonas, necesitaremos, en general, considerar iteraciones de una función f un número *transfinito* de veces α, donde α es un ordinal. Se escapa de los objetivos del presente trabajo dar un tratamiento detallado de los ordinales. Pero, para hacer el trabajo lo más autocontenido posible, se ha incluido un pequeño apéndice al respecto al final del presente capítulo.

Definición 2.10 (Potencias ordinales). *Sea $(D, \leq)$ un orden parcial completo (recordemos que $\bot$ denota su elemento mínimo). Definimos inductivamente las potencias ordinales de una función monótona $f : D \longrightarrow D$ como sigue:*

$$f \uparrow 0 = \bot$$

$$f \uparrow (\alpha + 1) = f(f \uparrow \alpha) \text{ para todo ordinal } \alpha$$

$$f \uparrow \alpha = sup\ \{f \uparrow \beta | \beta < \alpha\} \text{ si } \alpha \text{ es un ordinal límite}$$

Observación 2.2. Nótese que $\{f \uparrow \beta | \beta < \alpha\}$ es una cadena en D (la prueba formal de este hecho usa el principio de inducción transfinita) y, por tanto, puesto que D es

cadena-completo, el supremo de dicho conjunto existe en D y las potencias ordinales de f están bien definidas. En general, lo mismo sucede para para las potencias ordinales $f^\alpha(x)$ para cualquier $x \in D$ que cumpla $x \leq f(x)$ si definimos:

$$f^0(x) = x$$

$$f^{\alpha+1}(x) = f(f^\alpha(x)) \text{ para todo ordinal } \alpha$$

$$f^\alpha(x) = sup\ \{f^\beta(x) | \beta < \alpha\} \text{ si } \alpha \text{ es un ordinal límite}$$

(la definición 2.10 es el caso particular $x = \bot$).

Los dos siguientes teoremas de punto fijo son herramientas fundamentales para el manejo de las semánticas de los programas lógicos.

Teorema 2.1 (Kleene). *Sea $(D, \leq)$ un orden parcial ω-completo y $f : D \longrightarrow D$ una función ω-continua. Entonces, f tiene un menor punto fijo $x \in D$ que viene dada por la potencia ordinal $x = f \uparrow \omega$. Además, dicho x es también el menor punto prefijo de f.*

Demostración. El resultado es bien conocido. Daremos un esquema de su prueba.

Primero, puesto que $(f \uparrow n)_{n\in\mathbb{N}}$ una ω-cadena en D y D es un orden parcial ω-completo, $(f \uparrow n)_{n\in\mathbb{N}}$ tiene un supremo en D. Sea $x = f \uparrow \omega$.

Puesto que f es ω-continua, tenemos que

$$x = f \uparrow \omega = \sqcup\{f \uparrow (n+1) | n \in N\} = f(\sqcup\{f \uparrow n | n \in N\}) = f(x).$$

Por lo tanto, x es un punto fijo de f.

Sea ahora y un punto prefijo de f cualquiera. Por la monotonía de f y $\bot \leq y$, se tiene:

$$f \uparrow 1 = f(\bot) \leq f(y) \leq y.$$

Por inducción en n, se sigue que $f \uparrow n \leq y$ para todo $n \in \mathbb{N}$. Luego $x = f \uparrow \omega \leq y$. Por lo tanto, x es el menor punto prefijo de f y, en consecuencia, x es también su menor punto fijo, como queríamos demostrar.

El siguiente teorema nos garantiza la existencia de un punto fijo incluso en el caso en el que la función f no es ω-continua. Ahora bien, hay un precio a pagar: para obtener un punto fijo tendremos que considerar, en general, potencias ordinales para ordinales por encima de ω.

Teorema 2.2 (Knaster-Tarski). *Sea $(D, \leq)$ un orden parcial completo. Sean $f : D \longrightarrow D$ una función monótona y $x \in D$ tal que $x \leq f(x)$. Entonces, f tiene un menor punto fijo a sobre x (es decir, con $x \leq a$), que es también el menor punto prefijo de f sobre x. Además, existe un menor ordinal α tal que $a = f^{\alpha}(x)$.*
En particular, f tiene un menor punto fijo a, que también es su menor punto prefijo (basta considerar su menor punto fijo sobre $\bot$).

Demostración. El resultado es bien conocido. Daremos un esquema de la prueba.

Sea γ un ordinal tal que $\#D < \#\gamma$, esto es, el cardinal de γ es mayor que el cardinal del conjunto D. Consideremos el conjunto $\{f^{\beta}(x) \mid \beta \leq \gamma\}$. Por razones de cardinalidad, tiene que haber ordinales $\alpha < \beta \leq \gamma$ con $f^{\alpha}(x) = f^{\beta}(x)$ y podemos asumir sin pérdida de generalidad que α es el menor ordinal con esta propiedad. Puesto que

$$f^{\alpha}(x) \leq f(f^{\alpha}(x)) \leq f^{\beta}(x) = f^{\alpha}(x),$$

tenemos que $f^{\alpha}(x) = f(f^{\alpha}(x))$, y que $a = f^{\alpha}(x)$ es un punto fijo de f. Claramente, $x \leq a$. Luego, a es un punto fijo de f sobre x.

Además, si b es un punto prefijo de f tal que $x \leq b$ cualquiera, tenemos que $f^{\beta}(x) \leq b$ para todo ordinal β gracias a la monotonía de f y al hecho de que $f(b) \leq b$. Por tanto, $a \leq b$, y a es tanto el menor punto prefijo como el menor punto fijo de f sobre x.

Observación 2.3. Si suponemos que $(D, \leq)$ es un retículo completo (no solo un orden parcial completo) el Teorema de Knaster-Tarski nos dice que una función monótona f tiene tanto un menor como un mayor punto fijo. Más aún, el conjunto de los puntos fijos de f forma un nuevo retículo completo.

2.2 Lógica de Primer Orden

2.2.1 Sintaxis de la Lógica de Primer Orden

Nuestra aproximación a la sintaxis de la lógica de primer orden es completamente estándar. Recapitulamos las definiciones fundamentales para fijar notación y terminología.

Definición 2.11. *Un lenguaje de primer orden $\mathcal{L}$ (o alfabeto) es un conjunto de símbolos. Los símbolos de $\mathcal{L}$ son de los siguientes tipos:*

1. *Símbolos lógicos (comunes a todos los alfabetos de primer orden):*
   (a) Variables: $u, v, w, x, y, z, \ldots$
   (b) Conectivas: $\neg, \vee, \wedge, \rightarrow, \leftrightarrow, \exists, \forall$
   (c) Símbolos auxiliares: $(,)$
2. *Símbolos no lógicos:*
   (a) Símbolos de constantes: $a, b, c, d, \ldots$
   (b) Símbolos de función (con aridad): $f, g, h, \ldots$
   (c) Símbolos de predicado (con aridad): $p, q, r, \ldots$

En lo que sigue, suponemos que $\mathcal{L}$ denota un lenguaje arbitrario pero fijo.

Definición 2.12 (Términos). *Definimos un término sobre* $\mathcal{L}$ *como sigue:*

1. *Cada símbolo de constante en* $\mathcal{L}$ *es un término.*
2. *Cada símbolo de variable en* $\mathcal{L}$ *es un término.*
3. *Si* f *es una función n-aria y* $t_1, \ldots, t_n$ *son términos, entonces* $f(t_1, ..., t_n)$ *es un término.*

Definición 2.13 (Términos básicos). *Un término* t *de* $\mathcal{L}$ *se dirá un término básico (ground term, en inglés) si no contiene símbolos de variables.*

Definición 2.14 (Átomos). *Un átomo (o proposición)* A *sobre* $\mathcal{L}$ *es una expresión de la forma* $p(t_1, ..., t_n)$*, donde* p *es un símbolo de predicado n-ario en* $\mathcal{L}$ *y* $t_1, ..., t_n$ *son términos de* $\mathcal{L}$*.*

Definición 2.15 (Literal). *Un literal* L *es un átomo* A *o su negación* $\neg A$*, que se denominan literal positivo y literal negativo, respectivamente.*

Definición 2.16 (Átomos y Literales básicos). *Un átomo* A *o un literal* L *de* $\mathcal{L}$ *se dirá básico si no contiene símbolos de variables.*

Definición 2.17 (Fórmulas). *Una fórmula sobre* $\mathcal{L}$ *se define como sigue:*

1. *Cada átomo es una fórmula.*
2. *Si* F *y* G *son fórmulas, entonces* $\neg F$*,* $F \vee G$*,* $F \wedge G$*,* $F \rightarrow G$*,* $F \leftrightarrow G$ *lo son también.*
3. *Si* F *es una fórmula y* x *es un símbolo de variable, entonces* $\forall x\, F$ *y* $\exists x\, F$ *son también fórmulas.*

Observación 2.4. Como es habitual, es necesario usar paréntesis para evitar ambigüedades al escribir las fórmulas. La precedencia entre las distintas conectivas el la usual: $\neg, \forall, \exists, \vee, \wedge, \rightarrow \leftrightarrow$. Omitimos los detalles, que son bien conocidos.

2.2.2 Semántica de la Lógica de Primer Orden

El objetivo de esta subsección es describir brevemente la semántica de la lógica de primer orden. Al hacer esto, adoptamos el enfoque habitual de la teoría de modelos, pero con dos importantes cambios.

En primer lugar, puesto que el objetivo del presente trabajo es estudiar la semántica de los programas lógicos y un programa lógico puede reducirse a un conjunto de fórmulas sin variables y sin cuantificadores de primer orden, no será necesario tratar las variables libres y los cuantificadores ∃,∀ en nuestra aproximación a la semántica.

En segundo lugar, y de manera destacada, tendremos que manejar más valores de verdad que los dos convencionales $TWO = \{\mathbf{f}, \mathbf{t}\}$. En la lógica clásica de primer orden, y en la mayoría de los campos de las matemáticas, lo usual es emplear los valores de verdad clásicos: verdadero **t** y falso **f**. Sin embargo, en muchos lugares de la programación lógica y otras áreas de la Computación, ha resultado muy conveniente el uso de lógicas con más valores de verdad.

Por ejemplo, ya en los inicios del desarrollo de la programación lógica, Melvin Fitting propuso el uso de la lógica fuerte de Kleene (*Kleene's strong logic*), la cual emplea el conjunto de valores de verdad $THREE = \{\mathbf{u}, \mathbf{f}, \mathbf{t}\}$. Las constantes **f**, **t** denotan, como es habitual, los valores falso y verdadero y la constante **u** denota un tercer valor de verdad que puede interpretarse como *indeterminado* o *ninguno de los dos*. Fitting también consideró la llamada *lógica 4-valorada de Belnap*, cuyo conjunto de valores de verdad es $FOUR = \{\mathbf{u}, \mathbf{f}, \mathbf{t}, \mathbf{b}\}$. Ahora, **b** es un cuarto valor de verdad que representa *ambos* o *sobredefinido* (es decir, hay información contradictoria).

En general, podemos considerar la siguiente definición.

Definición 2.18 (Lógica). *Una lógica consta de:*

1. *Un conjunto de valores de verdad $\mathcal{T}$ que contiene, al menos, dos elementos y que contiene un elemento destacado **t** que denota Verdadero.*
2. *Definiciones de las conectivas proposicionales $\wedge, \vee, \neg, \rightarrow$ como funciones sobre $\mathcal{T}$.*

Definición 2.19 (Lógica clásica). *La lógica clásica corresponde a:*

- $\mathcal{T} = \mathcal{TWO} = \{\boldsymbol{f}, \boldsymbol{t}\}$.
- *Definiciones de $\wedge, \vee, \neg, \rightarrow$ habituales.*

Definición 2.20 (Lógica fuerte de Kleene). *La lógica fuerte de Kleene es la siguiente lógica trivaluada.*

- $\mathcal{T} = \mathcal{THREE} = \{\boldsymbol{u}, \boldsymbol{f}, \boldsymbol{t}\}$.
- *Las definiciones de* $\wedge$, $\vee$, $\neg$, $\rightarrow$ *de la siguiente tabla:*

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$
$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{t}$
$\boldsymbol{u}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{f}$
$\boldsymbol{u}$	$\boldsymbol{t}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{t}$	$\boldsymbol{t}$
$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{t}$
$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{t}$
$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{t}$
$\boldsymbol{t}$	$\boldsymbol{u}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{t}$	$\boldsymbol{f}$
$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{f}$
$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{t}$

Definición 2.21 (Lógica 4-valuada de Belnap). *La lógica de Belnap es la siguiente lógica 4-valuada.*

- $\mathcal{T} = \mathcal{FOUR} = \{\boldsymbol{u}, \boldsymbol{f}, \boldsymbol{t}, \boldsymbol{b}\}$.
- *Las definiciones de* $\wedge$, $\vee$, $\neg$, $\rightarrow$ *de la siguiente tabla:*

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$
$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{t}$
$\boldsymbol{u}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{f}$
$\boldsymbol{u}$	$\boldsymbol{t}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{t}$	$\boldsymbol{t}$
$\boldsymbol{u}$	$\boldsymbol{b}$	$\boldsymbol{u}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{t}$
$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{t}$
$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{t}$
$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{t}$
$\boldsymbol{f}$	$\boldsymbol{b}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{b}$	$\boldsymbol{t}$
$\boldsymbol{t}$	$\boldsymbol{u}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{t}$	$\boldsymbol{f}$
$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{f}$
$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{t}$
$\boldsymbol{t}$	$\boldsymbol{b}$	$\boldsymbol{f}$	$\boldsymbol{b}$	$\boldsymbol{t}$	$\boldsymbol{f}$
$\boldsymbol{b}$	$\boldsymbol{u}$	$\boldsymbol{b}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{t}$
$\boldsymbol{b}$	$\boldsymbol{f}$	$\boldsymbol{b}$	$\boldsymbol{f}$	$\boldsymbol{b}$	$\boldsymbol{f}$
$\boldsymbol{b}$	$\boldsymbol{t}$	$\boldsymbol{b}$	$\boldsymbol{b}$	$\boldsymbol{t}$	$\boldsymbol{t}$
$\boldsymbol{b}$	$\boldsymbol{b}$	$\boldsymbol{b}$	$\boldsymbol{b}$	$\boldsymbol{b}$	$\boldsymbol{t}$

Nótese que tanto la lógica clásica como la lógica fuerte de Kleene pueden verse como *sublógicas* de la lógica 4-valuada de Belnap.

Introducimos ahora la noción de preinterpretación. Dicho informalmente, asignamos un significado a cada constante y a cada símbolo de función de un lenguaje de primer orden.

Definición 2.22 (Preinterpretación). *Sean $\mathcal{L}$ un lenguaje de primer orden y D un conjunto no vacío.*
Una preinterpretación J para $\mathcal{L}$ con dominio D es una aplicación $\cdot^J$ que satisface:

1. *$c^J \in D$ para cada c símbolo de constante de $\mathcal{L}$.*
2. *$f^J : D^n \rightarrow D$ es una función n-aria sobre D para cada f símbolo de función n-aria de $\mathcal{L}$.*

Una asignación de variables sobre D es un aplicación, θ, que asocia a cada símbolo de variable un elemento del dominio D.

Dadas una preinterpretación J con dominio D y una asignación de variables θ, podemos asignar a cada término t de $\mathcal{L}$ su denotación o significado, que será un elemento del conjunto D, definido como sigue:

- Si t es un símbolo de variable, $(x\theta)^J = \theta(x)$.
- Si t es un símbolo constante, $(c\theta)^J = c^J$.
- Si $t = f(t_1, ..., t_n)$ para alguna función n-aria f y siendo $t_1, ..., t_n$ términos,

$$(t\theta)^J = (f^J)((t_1\theta)^J, ..., (t_n\theta)^J).$$

Definición 2.23 (Instancias básicas, $B_{\mathcal{L},\mathcal{J}}$). *Fijemos una preinterpretación J con dominio D.*

- *Sea θ una asignación de variables sobre D y sea $A = p(t_1, ..., t_n)$ un átomo del lenguaje $\mathcal{L}$. Definimos $(A\theta)^J$ como $p((t_1\theta)^J, ..., (t_n\theta)^J)$ y lo llamaremos una instancia básica del átomo A.*
- *Denotamos por $B_{\mathcal{L},\mathcal{J}}$ el conjunto de todas las instancias básicas de átomos de $\mathcal{L}$ usando asignaciones de variables sobre D.*

Nótese que en $B_{\mathcal{L},\mathcal{J}}$ están todas las expresiones $p(d_1, ..., d_n)$ donde p es un símbolo de predicado n-ario y $d_1, ..., d_n \in D$.

Introducimos ahora una noción fundamental para el desarrollo del presente trabajo: el de *valoración de verdad* o *interpretación.*

Definición 2.24 (Valoración de verdad o interpretación). *Consideramos un lenguaje de primer orden $\mathcal{L}$, una preinterpretación J para $\mathcal{L}$ con dominio D y una lógica $\mathcal{T}$. Una valoración de verdad o interpretación es una función*

$$v : B_{\mathcal{L},\mathcal{J}} \to \mathcal{T}$$

que asigna a cada átomo en $B_{\mathcal{L},\mathcal{J}}$ uno de los valores de verdad de la lógica $\mathcal{T}$.

Dadas una valoración de verdad v y una asignación de variables θ, podemos extender la valoración de verdad v a cualquier *fórmula abierta* (sin cuantificadores de primer orden) del lenguaje $\mathcal{L}$ de manera natural. Para calcular $v(F)$ basta considerar la construcción inductiva de la fórmula abierta F y las definiciones de las conectivas proposicionales presentes en la lógica $\mathcal{T}$. Omitimos los detalles, por otra parte bien conocidos.

Definición 2.25 (Modelo). *Consideramos un lenguaje de primer orden $\mathcal{L}$, una preinterpretación J para $\mathcal{L}$ con dominio D y una lógica $\mathcal{T}$. Sean v una valoración de verdad y F una fórmula abierta sin variables. Diremos que v es un modelo de F, escrito $v \vDash F$, si v asocia a F el valor $\mathbf{t}$ de la lógica $\mathcal{T}$.*

Observación 2.5. Obsérvese que en la definición anterior no es necesario el uso de una asignación de variables θ porque la fórmula abierta F no contiene variables. Por otra parte, si los valores de verdad de la lógica $\mathcal{T}$ están ordenados con estructura de retículo completo (veremos ejemplos justo a continuación), entonces una valoración v puede extenderse de manera canónica a cualquier fórmula F del lenguaje sin variables libres: el cuantificador universal $\forall$ corresponde al ínfimo de los valores de verdad, y el cuantificador existencial $\exists$ corresponde al supremo de los valores de verdad.

2.3 El espacio de las valoraciones de verdad como conjunto ordenado

Consideramos un lenguaje de primer orden $\mathcal{L}$, una preinterpretación J para $\mathcal{L}$ con dominio D y una lógica $\mathcal{T}$.

Definición 2.26. *Denotamos por $I(B_{\mathcal{L},J}, \mathcal{T})$ al conjunto de todas las valoraciones de verdad basadas en J con valores en $\mathcal{T}$.*

En esta sección veremos cómo podemos otorgar a $I(B_{\mathcal{L},J}, \mathcal{T})$ estructura de conjunto ordenado. De hecho, bastará ordenar el conjunto de valores de verdad de la lógica subyacente $\mathcal{T}$ y el espacio $I(B_{\mathcal{L},J}, \mathcal{T})$ heredará dicha estructura de manera canónica.

En lo que sigue, supondremos que los valores de verdad de la lógica $\mathcal{T}$ vienen dados con una estructura de orden: $(\mathcal{T}, \leq)$. Por ejemplo, a lo largo del trabajo, emplearemos los siguientes órdenes.

Definición 2.27 (Órdenes sobre valores de verdad).

1. *Para $TWO = \{\boldsymbol{f}, \boldsymbol{t}\}$, consideramos el orden $<_t$ dado por*

$$\boldsymbol{f} <_t \boldsymbol{t}.$$

2. *Para $THREE = \{\boldsymbol{u}, \boldsymbol{f}, \boldsymbol{t}\}$, consideraremos dos órdenes distintos.*

   Orden de la verdad $<_t$: $\boldsymbol{f} <_t \boldsymbol{u} <_t \boldsymbol{t}$.

   Orden del conocimiento $<_k$: $\boldsymbol{u} <_k \boldsymbol{f}$ *y* $\boldsymbol{u} <_k \boldsymbol{t}$ *($\boldsymbol{f}$ y $\boldsymbol{t}$ incomparables).*

Observación 2.6. Nótese que tanto $(TWO, \leq_t)$ como $(THREE, \leq_t)$ son retículos completos con elemento mínimo $\bot = \mathbf{f}$ y elemento máximo $\top = \mathbf{t}$. Por otra parte, $(THREE, \leq_k)$ no es un retículo completo (el conjunto $\{\mathbf{f}, \mathbf{t}\}$ no tiene supremo) pero sí es un semi-retículo superior completo con elemento mínimo $\bot = \mathbf{u}$.

Veamos ahora cómo extender la estructura de orden de la lógica subyacente al espacio de las valoraciones de verdad $I(B_{\mathcal{L},\mathcal{J}}, \mathcal{T})$. En realidad, la naturaleza del conjunto $B_{\mathcal{L},\mathcal{J}}$ no desempeña ningún papel especial y podemos dar la definición en un contexto general.

Definición 2.28 (Órdenes sobre valoraciones). *Sea X un conjunto no vacío y sea $I(X, \mathcal{T})$ el espacio de las funciones de X en el conjunto de los valores de verdad de la lógica $(\mathcal{T}, \leq)$. Sean u, v elementos de $I(X, \mathcal{T})$. Definimos*

$$u \leq v \text{ si y solo si } \forall x \in X\ (u(x) \leq v(x)).$$

Es fácil comprobar que si $\leq$ es un orden parcial en $\mathcal{T}$, entonces el orden inducido en $I(X, \mathcal{T})$ es también un orden parcial. Además, si $\mathcal{T}$ tiene elemento mínimo $\bot$, entonces la función dada por $v(x) = \bot$ para todo $x \in X$ es el elemento mínimo de $I(X, \mathcal{T})$. Y así podríamos continuar con el resto de nociones de conjuntos ordenados que hemos estudiado. La siguiente proposición recopila estos hechos.

Proposición 2.3. Sean $X \neq \emptyset$ y $(\mathcal{T}, \leq)$ un conjunto de valores de verdad con elemento mínimo $\bot$. Entonces:

1. Si $(\mathcal{T}, \leq)$ es un orden parcial, también lo es $I(X, \mathcal{T})$.
2. Si $(\mathcal{T}, \leq)$ es un orden parcial ω-completo, también lo es $I(X, \mathcal{T})$.
3. Si $(\mathcal{T}, \leq)$ es un orden parcial completo, también lo es $I(X, \mathcal{T})$.
4. Si $(\mathcal{T}, \leq)$ es un semi-retículo superior completo, también lo es $I(X, \mathcal{T})$.
5. Si $(\mathcal{T}, \leq)$ es un retículo completo, también lo es $I(X, \mathcal{T})$.
6. Si $(\mathcal{T}, \leq)$ es un dominio de Scott, también lo es $I(X, \mathcal{T})$.

Demostración. Es rutina comprobar que las correspondientes propiedades se transfieren del orden $(\mathcal{T}, \leq)$ al espacio $I(X, \mathcal{T})$.

De la observación 2.6 y la Proposición anterior, inmediatamente se sigue que:

Teorema 2.3. *Sea X un conjunto no vacío.*

1. *Si $\mathcal{T} = \mathcal{TWO}$, entonces $I(X, \mathcal{T})$ es un retículo completo.*

2. *Si $\mathcal{T} = \mathcal{THREE}$, entonces $I(X, \mathcal{T})$ es un semi-retículo superior completo para el orden $\leq_k$ y es un retículo completo para el orden $\leq_t$.*

Aunque la definición "oficial" de valoración de verdad o interpretación es en términos de funciones, resulta muy útil en determinados contextos disponer de una notación conjuntista alternativa. La describimos a continuación y la usaremos libremente en el resto del trabajo.

Empecemos con valoraciones de verdad sobre el conjunto $TWO = \{\mathbf{f}, \mathbf{t}\}$. Sea una tal $v : X \to TWO$. Está claro que v está determinada por el conjunto

$$v^{-1}(\mathbf{t}) = \{x \in X \ : \ v(x) = \mathbf{t}\} \subseteq X.$$

Y, a la inversa, todo $I \subseteq X$ determina una única valoración de verdad:

$$v(x) = \mathbf{t} \text{ si } x \in I \qquad v(x) = \mathbf{f} \text{ si } x \notin I.$$

Luego, podemos identificar valoraciones de verdad sobre TWO con subconjuntos I de X, entendiendo que los elementos que aparecen en I tienen valor de verdad *Verdadero* y los elementos de $X - I$, *Falso*. En particular, la interpretación $\emptyset$ representa un escenario donde todos los átomos se consideran falsos.

Consideremos ahora valoraciones de verdad sobre el conjunto $THREE = \{\mathbf{u}, \mathbf{f}, \mathbf{t}\}$. Sea una tal $v : X \to THREE$. Puesto que ahora tenemos un tercer valor de verdad $\mathbf{u}$, necesitamos dos subconjuntos de X para determinar la valoración v:

$$v^{-1}(\mathbf{t}) = \{x \in X \ : \ v(x) = \mathbf{t}\} \text{ y } v^{-1}(\mathbf{f}) = \{x \in X \ : \ v(x) = \mathbf{f}\}.$$

A la inversa, cualquier pareja de subconjuntos de X, (I_1, I_2), con $I_1 \cap I_2 = \emptyset$ determina una única valoración de verdad:

$$v(x) = \mathbf{t} \text{ si } x \in I_1 \qquad v(x) = \mathbf{f} \text{ si } x \in I_2 \qquad v(x) = \mathbf{u} \text{ si } x \notin (I_1 \cup I_2).$$

Podemos pues identificar valoraciones de verdad sobre $THREE$ con pares (I_t, I_f), donde $I_t, I_f \subseteq X$ y $I_t \cap I_f = \emptyset$. Entendemos que los elementos que aparecen en I_t son verdaderos, los elementos en I_f son falsos y los elementos en $X - (I_t \cup I_f)$ tienen el valor de verdad *indeterminado*. En particular, la interpretación $(\emptyset, \emptyset)$ representa un escenario donde todos los átomos tienen el valor *indeterminado*.

Para valoraciones de verdad sobre $THREE$, consideraremos una tercera representación, los llamados *conjuntos signados*. Para ello identificamos una valoración (I_t, I_f), donde $I_t, I_f \subseteq X$ y $I_t \cap I_f = \emptyset$, con el conjunto

$$I_t \cup \{\neg x \mid x \in I_f\}.$$

Observación 2.7. Aunque pasaremos de una representación a otra según convenga, a veces se reserva el término de *valoración de verdad* para referirnos a funciones y el término de *interpretación* para referirnos a su versión conjuntista. Una valoración o interpretación que sobre TWO se dirá 2-valorada (o 2-valuada); sobre $THREE$, 3-valorada, ...

Ejemplo 2.5. Consideremos $B_{\mathcal{L}} = \{p(a), q(a), p(b), q(b), p(c), q(c)\}$. Sea la valoración de verdad 3-valorada v dada por:

$$v(p(a)) = \mathbf{u}, v(q(a)) = \mathbf{u}, v(p(b)) = \mathbf{t}, v(q(b)) = \mathbf{f}, v(p(c)) = \mathbf{t}, v(q(c)) = \mathbf{t}$$

Su representación como interpretación sería: $I = (\{p(b), p(c), q(c)\}, \{q(b)\})$.
Su representación como conjunto signado sería $I = \{p(b), p(c), q(c), \neg q(b)\}$.
En ambas, los átomos que no aparecen tienen el valor $\mathbf{u}$.

La siguiente proposición describe el orden en el espacio de las interpretaciones $I(X, \mathcal{T})$ en términos conjuntistas.

Proposición 2.4. Sea X un conjunto no vacío.

1. Si I y K son interpretaciones 2-valoradas, entonces $I \leq_t K$ si, y solo si, $I \subseteq K$ como subconjuntos de X.
2. Si $I = (I_t, I_f)$ y $K = (K_t, K_f)$ son interpretaciones 3-valoradas, entonces:
   $I \leq_k K$ si, y solo si, $I_t \subseteq K_t$ y $I_f \subseteq K_f$.
   $I \leq_t K$ si, y solo si, $I_t \subseteq K_t$ y $K_f \subseteq I_f$.

Ejemplo 2.6. Dado el conjunto del ejemplo anterior, consideremos I y K dos interpretaciones 2-valoradas tales que $I = \{p(a), q(a)\}$ y $K = \{p(a), q(a), p(c), q(c)\}$. Entonces, por la Proposición 2.4, $I \leq_t K$ ya que $I \subseteq K$.

Consideremos ahora las siguientes interpretaciones 3-valoradas, J y H. Por un lado, $J = (J_t, J_f)$ donde $J_t = \{p(a), q(b)\}$ y $J_f = \{p(b), q(c)\}$. Por otro lado, $H = (H_t, H_f)$ donde $H_t = \{p(a), q(b), p(c)\}$ y $H_f = \{p(b), q(c)\}$. En este caso particular, por la Proposición 2.4, se cumple tanto $J \leq_k H$ como $J \leq_t H$.

Apéndice: Ordinales

Con el objetivo de hacer la presente memoria lo más autocontenida posible, recopilamos a continuación algunas definiciones y resultados fundamentales sobre la

teoría de *ordinales* que emplearemos a lo largo del trabajo. Cabe aclarar que trabajamos desde un punto de vista más intuitivo que formal y que nuestro objetivo no es, en absoluto, dar una introducción rigurosa al concepto de ordinal en Teoría de Conjuntos.

Definición 2.29 (Buenos órdenes). *Un conjunto $(X, \leq)$ parcialmente ordenado está bien ordenado (o es un buen orden) si cada subconjunto no vacío de X tiene elemento mínimo.*

Por ejemplo, el conjunto de los números naturales con el orden usual $(\mathbb{N}, \leq)$ es un buen orden; mientras que $(\mathbb{Z}, \leq)$ no lo es.

Lema 2.1.

1. Todo conjunto bien ordenado es un orden total.
2. Un conjunto bien ordenado no contiene sucesiones infinitas estrictamente decrecientes.

Demostración.

1. Sean $(X, \leq)$ un conjunto bien ordenado y $a, b \in X$. Por la definición de buen orden, el conjunto no vacío $\{a, b\}$ tiene un menor elemento, digamos a. Entonces, $a \leq b$ y los elementos a y b son comparables.
2. Sea $(X, \leq)$ un conjunto bien ordenado. Por reducción al absurdo, supongamos que $(x_n)_{n \in N}$ es una sucesión estrictamente decreciente en X. Entonces, $\{x_n \mid n \in \mathbb{N}\}$ es un subconjunto no vacío de X que no tiene elemento mínimo. Esto contradice la hipótesis de que $(X, \leq)$ está bien ordenado.

Dados $(X, \leq_x)$ e $(Y, \leq_y)$ dos conjuntos bien ordenados, $f : X \to Y$ se dirá monótona si $a \leq_x b$ implica que $f(a) \leq_y f(b)$ para todo $a, b \in X$. Si f es monótona e inyectiva, f se denomina una *inmersión* de X en Y. Si f es monótona y biyectiva, entonces f es un *isomorfismo de orden* (o simplemente isomorfismo) entre X e Y y, en este caso, X e Y se dirán isomorfos.

Definición 2.30 (Segmento inicial). *Sean $(X, \leq)$ un conjunto bien ordenado y $x_0 \in X$. El conjunto $I = I(x_0) = \{x \in X \mid x \leq x_0\}$ se denomina segmento inicial de X determinado por x_0. Este segmento se dirá propio si $I \neq X$.*

Definición 2.31. *Sean $(X, \leq_x)$ e $(Y, \leq_y)$ dos conjuntos bien ordenados. Escribiremos $X \leq Y$ si X es isomorfo a algún segmento inicial de Y. Escribiremos $X < Y$ si X es isomorfo a algún segmento inicial propio de Y.*

Incluimos sin demostración un resultado fundamental sobre buenos órdenes.

Teorema 2.4. *Sean X e Y dos conjuntos bien ordenados. Se cumple una, y sólo una, de las siguientes afirmaciones:*

1. $X < Y$.
2. $X > Y$.
3. *X e Y son isomorfos.*

A continuación enunciamos un resultado bien conocido, atribuido a E. Zermelo, cuya prueba usa de manera esencial el Axioma de Elección de la Teoría de Conjuntos (de hecho, dicho resultado es formalmente equivalente al Axioma de Elección).

Teorema 2.5 (Teorema del buen orden). *Todo conjunto puede ser bien ordenado. Esto es, para todo conjunto X, existe un buen orden sobre X.*

Este resultado tendrá como consecuencia que cualquier conjunto X es, de hecho, el dominio de algún *ordinal*, atendiendo a la siguiente definición.

Definición 2.32 (Ordinal). *Un ordinal o número ordinal es una clase de equivalencia de un buen orden $(A, \leq)$ bajo la relación de equivalencia "isomorfía de buenos órdenes". Denotaremos por $\#A$ a la clase de equivalencia del buen orden A. Usaremos letras griegas $\alpha, \beta, \ldots$ para denotar ordinales.*

Por ejemplo, la clase formada por todos los buenos órdenes isomorfos a $(\mathbb{N}, \leq)$ es un ordinal, que puede nombrarse por $\omega = \#(\mathbb{N})$.

Los ordinales, a su vez, pueden ser ordenados de la siguiente manera. Sean α y β dos ordinales. Tomemos dos conjuntos A y B tales que $\alpha = \#(A)$ y $\beta = \#(B)$. Entonces,

$$\alpha \leq \beta \quad \text{si y solo si} \quad A < B.$$

Es fácil ver que dicha relación de orden está bien definida: no depende de los representantes de las clases de equivalencia elegidos.

Nótese que, en este contexto, el Teorema 2.4 nos dice que el orden entre los ordinales que acabamos de definir es un orden total. Más áun, también se puede probar que el orden entre ordinales es un buen orden.

En lo que sigue, por abuso de notación, confundiremos un ordinal α con un representante de la clase de equivalencia que lo define (elegido de alguna forma canónica).

Es habitual, y así lo haremos, identificar un ordinal α con el conjunto de todos los ordinales β tales que $\beta < \alpha$. Luego, $\beta \in \alpha$ si, y solo si, $\beta < \alpha$. En particular, cuando en los próximos capítulos hablemos de una función

$$F : X \to \alpha,$$

donde α es un ordinal (por ejemplo, cuando introduzcamos funciones de nivel para programas lógicos), entenderemos que f es una función de X en $\{\beta \mid \beta < \alpha\}$.

Los ordinales se clasifican en tres grupos.

- El vacío es un ordinal: $0 = \#(\emptyset)$.
- Un *ordinal sucesor* es un ordinal α tal que existe el mayor ordinal β con $\beta < \alpha$, y denotaremos $\alpha = \beta + 1$.
- Un ordinal que no es ni 0 ni sucesor es un *ordinal límite*.

Observación 2.8. Un conjunto finito que contiene n elementos puede ordenarse, salvo isomorfismo, esencialmente de una única manera. La notación estándar, junto con los representantes canónicos de las correspondientes clases de equivalencias, para los ordinales finitos son:

$$\begin{aligned} 0 &= \#(\emptyset) \\ 1 &= \#(\{\emptyset\}) \\ 2 &= \#(\{\emptyset, \{\emptyset\}\}) \\ 3 &= \#(\{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}) \\ &\vdots \end{aligned}$$

El primer ordinal infinito es $\omega = \#(\mathbb{N})$ (con el orden usual), que también es el primer ordinal límite.

$$\begin{aligned} \omega &= \#(\{0, 1, 2, \dots\}) \\ \omega + 1 &= \#(\{0, 1, 2, \dots, \omega\}) \\ \omega + 2 &= \#(\{0, 1, 2, \dots, \omega, \omega + 1\}) \\ \omega + 3 &= \#(\{0, 1, 2, \dots, \omega, \omega + 1, \omega + 2\}) \\ &\vdots \\ \omega \cdot 2 &= \#(\{0, 1, 2, \dots, \omega, \omega + 1, \omega + 2, \dots\}) \\ &\vdots \end{aligned}$$

Podemos ya enunciar los resultados técnicos claves que usaremos en el trabajo: los principios de recursión e inducción transfinita. Estos principios pueden verse como extensiones de $\mathbb{N}$ a un buen orden arbitrario de los bien conocidos principios de definición e inducción sobre los números naturales.

Teorema 2.6 (Principio de Inducción Transfinita). *Sea $P(\alpha)$ una propiedad sobre ordinales. Si*

1. *se cumple $P(0)$,*
2. *$P(\alpha)$ implica $P(\alpha+1)$, y,*
3. *para α límite, $P(\alpha)$ es cierta siempre que lo sea $P(\beta)$ para todo $\beta < \alpha$;*

entonces, $P(\alpha)$ se cumple para todo α.

Teorema 2.7 (Principio de Recursión Transfinita). *Sean X un conjunto y G, H funciones. Existe una única función F definida sobre los ordinales tal que:*

1. $F(0) = X$,
2. $F(\alpha+1) = G(F(\alpha))$, *y*
3. $F(\alpha) = H(\{F(\beta) \mid \beta < \alpha\})$; *si α es límite.*

Estos principios serán relevantes en el presente trabajo para el estudio de los puntos fijos de ciertos operadores monótonos sobre el espacio ordenado de las valoraciones de verdad asociados a programas lógicos.

3 Modelos justificados y Modelos estables

El objetivo de este capítulo es introducir los programas lógicos normales, y las nociones fundamentales de la semántica declarativa asociada a estos programas, así como dos de las principales semánticas propuestas para estos programas: los modelos justificados y los modelos estables.

Como punto de partida, estudiamos también la semántica de punto fijo para los programas definidos y el operador de consecuencia inmediata T_P asociado.

3.1 Programas lógicos normales

Comenzamos con la definición del principal objeto de estudio de este trabajo: los programas lógicas normales. De manera intuitiva, un programa lógico normal no es más que un conjunto finito de cláusulas (también llamadas reglas) de un lenguaje de primer orden, que entenderemos tienen como objetivo representar un cierto conocimiento sobre un cierto dominio.

Definición 3.1 (Regla o cláusula). *Sea $\mathcal{L}$ un lenguaje de primer orden. Una regla o cláusula de $\mathcal{L}$ es una fórmula de la forma:*

$$\forall x_1 \dots \forall x_k \, (L_1 \wedge \dots \wedge L_n \rightarrow A),$$

donde A es un átomo de $\mathcal{L}$, $L_1, \dots, L_n$ son literales de $\mathcal{L}$, $x_1, \dots, x_k$ son los símbolos variables que hay en la fórmula y $n, k \in \mathbb{N}$. Normalmente, escribiremos las reglas de la siguiente manera:

$$\forall x_1 \dots \forall x_k \, (A \leftarrow L_1 \wedge \dots \wedge L_n).$$

Más aún, omitiremos los cuantificadores universales y escribiremos (,) en lugar de la conectiva $\wedge$ para expresar la conjunción de los literales. Una cláusula escrita de manera simplificada será por tanto una expresión de la forma:

$$A \leftarrow L_1, ..., L_n,$$

donde $n \geq 0$. El átomo A se denomina la cabeza de la regla y el conjunto de literales $L_1, ..., L_n$ el cuerpo (body, en inglés) de la regla. Obsérvese que n puede tomar el valor 0. Es decir, admitimos reglas sin ningún literal en el cuerpo. Dichas reglas especiales se llamarán hechos y se representarán por $A \leftarrow$ (o, simplemente, A).

Definición 3.2 (Programa lógico normal). *Sea $\mathcal{L}$ un lenguaje de primer orden. Un programa lógico normal (o, simplemente, un programa lógico) P es un conjunto finito de cláusulas de $\mathcal{L}$.*

Destacamos una clase relevante de programas lógicos normales que, como veremos, se comportan especialmente bien a la hora de definir su semántica.

Definición 3.3 (Programa lógico definido). *Sea $\mathcal{L}$ un lenguaje de primer orden. Un programa lógico normal P se dirá definido o positivo si en él no aparece la negación $\neg$. Dicho de otro modo, los cuerpos de todas las reglas de P están formados por literales positivos del lenguaje $\mathcal{L}$.*

Ilustraremos las definiciones anteriores con un par de ejemplos. A menos que se diga lo contrario, usaremos siempre el siguiente convenio a la hora de escribir un programa lógico: las constantes, las funciones y los símbolos de predicado comenzarán con letra minúscula; mientras que los símbolos de variable comenzarán siempre por letra mayúscula.

Ejemplo 3.1 (PIOLÍN). Sean Bob y Piolín un pingüino y un pájaro, respectivamente. El siguiente programa lógico normal representa el conocimiento: todos los pingüinos son pájaros, y todo pájaro que no sea un pingüino puede volar.

```
pinguino(piolin) ←
pajaro(bob) ←
pajaro(X) ← pinguino(X)
vuela(X) ← pajaro(X),¬pinguino(X)
```

Ejemplo 3.2 (GRAFO). Representamos un grafo no dirigido mediante los predicados `nodo/1` y `arco/2`. El siguiente programa lógico definido expresa la relación "hay un camino en el grafo entre los nodos X e Y".

```
nodo(a) ←
nodo(b) ←
nodo(c) ←
arco(a,b) ←
arco(b,c) ←
camino(X,X) ← nodo(X)
camino(X,Y) ← arco(X,Z),camino(Z,Y)
```

Ejemplo 3.3 (PAR). El siguiente programa lógico normal expresa el conocimiento: 0 es un número par y, si un número natural no es par, entonces su sucesor sí lo es.

```
par(a) ←
par(s(X)) ← ¬ par(X)
```

Obsérvese que en este ejemplo s denota un símbolo de función del lenguaje de primer orden subyacente.

Hasta ahora, hemos especificado la sintaxis de los programas lógicos. Tratamos a continuación la *semántica* de dichos programas. Nuestro punto de partida será adaptar las definiciones de preinterpretación e interpretación para un lenguaje de primer orden general que vimos en el Capítulo 2 al presente contexto.

Definición 3.4 (Instancias básicas, $ground(P)$, $B_{P,J}$). *Sea P una programa lógico normal y denotemos por $\mathcal{L}_P$ su lenguaje de primer orden asociado. Sea D un conjunto no vacío.*

1. *Una preinterpretación J para P con dominio D es una preinterpretación para $\mathcal{L}_P$ con dominio D.*
2. *Fijemos una preinterpretación J para P y una asignación de variables θ sobre D. Dada una cláusula C en P de la forma*

$$A \leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m$$

denotamos por $(C\theta)^J$ la cláusula sin variables

$$(A\theta)^J \leftarrow (A_1\theta)^J, \ldots, (A_n\theta)^J, \neg(B_1\theta)^J, \ldots, \neg(B_m\theta)^J$$

y la llamaremos una instancia básica de C.

3. *Denotamos por* $ground_J(P)$ *al conjunto de todas las instancias básicas de las cláusulas de* P *usando asignaciones de variables sobre el dominio* D *asociado a* J*. Nótese que* $ground_J(P)$ *es un conjunto de cláusulas sin variables y que* $ground_J(P)$ *puede ser infinito (si el dominio* D *lo es).*
4. *Denotamos por* $B_{P,J}$ *el conjunto de todas las expresiones de la forma* $p(d_1, \ldots, d_n)$*, donde* p *es un símbolo de predicado n-ario de* $\mathcal{L}_P$ *y* $d_1, \ldots, d_n \in D$*.*

Ejemplo 3.4. Consideremos de nuevo el programa PAR del ejemplo 3.3.

```
par(a) ←
par(s(X)) ← ¬ par(X)
```

El lenguaje asociado es $\mathcal{L}_p = \{a, s/1, par/1\}$. Una preinterpretación J para PAR con dominio $\{1, 2, 3\}$ es, por ejemplo:

$$a^J = 2 \qquad s^J(1) = 2 \quad s^J(2) = 1 \quad s^J(3) = 3$$

En este caso, $B_{P,J} = \{par(1), par(2), par(3)\}$ y $ground_J(P)$ es

```
par(2) ←
par(2) ← ¬ par(1)
par(1) ← ¬ par(2)
par(3) ← ¬ par(3)
```

Aunque nuestra definición permite preinterpretaciones generales, nos centraremos en un tipo particular de estas de espacial interés: las *preinterpretaciones de Herbrand.*

Definición 3.5 (Universo y base de Herbrand, *ground*(P)**).** *Sea* P *un programa lógico normal y denotemos por* $\mathcal{L}_P$ *su lenguaje de primer orden asociado.*

1. *El universo de Herbrand de* P*,* $HU(P)$*, es el conjunto de todos los términos básicos del lenguaje* $\mathcal{L}_P$*.*
2. *La base de Herbrand de* P*,* B_P *o* $HB(P)$*, es el conjunto de todas las expresiones de la forma* $p(d_1, \ldots, d_n)$*, donde* p *es un símbolo de predicado n-ario de* $\mathcal{L}_P$ *y* $d_1, \ldots, d_n \in HU(P)$*.*
3. *La preinterpretación de Herbrand para* P *es una preinterpretación,* J*, con dominio* $HU(P)$ *y que asigna cada término de* $\mathcal{L}_P$ *en sí mismo (visto como elemento del universo de Herbrand). Esto es:*

- c^J *es igual a* $c \in HU(P)$*, para cada constante* c *de* $\mathcal{L}_P$*.*
- f^J *es la función dada por* $f^J(t_1, \ldots, t_n) = f(t_1, \ldots, t_n) \in HU(P)$*, para cada función* f *de* $\mathcal{L}_P$*.*

4. *Denotamos por* $ground(P)$ *al conjunto de todas las instancias básicas de las cláusulas de* P *usando asignaciones de variables sobre* $HU(P)$*.*

Nótese que el universo de Herbrand de un programa P será infinito si el lenguaje de P contiene algún símbolo de función, y que el universo de Herbrand será siempre no vacío (si el lenguaje de P no contuviese ningún símbolo de constante, se añadiría una constante virtual a al lenguaje).

Ejemplo 3.5 (PIOLÍN). Consideremos de nuevo el Ejemplo 3.1

```
pinguino(piolin) ←
pajaro(bob) ←
pajaro(X) ← pinguino(X)
vuela(X) ← pajaro(X),¬pinguino(X)
```

Se tiene que $\mathcal{L}_P = \{piolin, bob, pinguino/1, pajaro/1, vuela/1\}$. Por tanto:

$HU(P) = \{bob, piolin\}$
$B_P = HB(P) = \{pinguino(bob), pinguino(piolin), pajaro(bob), pajaro(piolin), vuela(bob), vuela(piolin)\}$

El conjunto $ground(P)$ consta de las siguientes cláusulas:

```
pinguino(piolin) ←
pajaro(bob) ←
pajaro(bob) ← pinguino(bob)
pajaro(piolin) ← pinguino(piolin)
vuela(bob) ← pajaro(bob),¬pinguino(bob)
vuela(piolin) ← pajaro(piolin),¬pinguino(piolin)
```

Ejemplo 3.6 (PAR). Si consideramos el programa PAR del ejemplo 3.3, se tiene que:

$HU(P) = \{a, s(a), s(s(a)), s(s(s(a))), \ldots\}$
$B_P = HB(P) = \{par(a), par(s(a)), par(s(s(a))), par(s(s(s(a)))), \ldots\}$

El conjunto infinito $ground(P)$ consta de las siguientes cláusulas:

```
par(a) ←
par(s(a)) ← ¬ par(a)
par(s(s(a))) ← ¬ par(s(a))
par(s(s(s(a)))) ← ¬ par(s(s(a)))
    ⋮
```

Enunciamos a continuación las definiciones fundamentales de interpretación y modelo para un programa lógico normal. Lo haremos para una preinterpretación general J ya para una lógica general $\mathcal{T}$ (definida como en el capítulo anterior y con un valor de verdad distinguido $\mathbf{t}$ para denotar *Verdadero*).

Definición 3.6 (Modelo de un programa lógico, Modelos de Herbrand). *Sean P un programa normal, J una preinterpretación para P y $\mathcal{T}$ una lógica.*

1. *Una valoración de verdad o interpretación para P (basada en J y con valores en $\mathcal{T}$) es una aplicación v que asocia a cada elemento de $B_{P,J}$, A, un valor de verdad, $v(A)$.*
2. *Una valoración de verdad o interpretación v se dirá un J-modelo (o, simplemente, modelo) de P si $v(C) = \mathbf{t}$ para toda cláusula C en $ground_J(P)$ (para calcular el valor de $v(C)$ se usarán las definiciones de las conectivas incluidas en la lógica $\mathcal{T}$).*
3. *Una interpretación de Herbrand de P es una interpretación para P basada en la preinterpretación de Herbrand de P.*
4. *Un modelo de Herbrand de P es un modelo de P basado en la preinterpretación de Herbrand de P.*

Como describimos en el capítulo anterior, usaremos a menudo notación conjuntista para representar las valoraciones de verdad y, típicamente, reservaremos el término interpretación para dichas representaciones. Nótese que, en este contexto, las interpretaciones de Herbrand de un programa P basadas en la lógica clásica TWO pueden identificarse con subconjuntos de la base de Herbrand de P, $I \subseteq B_P$. Análogamente, las interpretaciones de Herbrand de un programa P basadas en la lógica $THREE$ pueden identificarse con pares (I, K), donde $I, K \subseteq B_P$ y $K \cap I = \emptyset$.

Ejemplo 3.7. Consideramos el programa lógico P con una sola cláusula

```
p(c) ← p(a), ¬ p(b)
```

Entonces, $UH(P) = \{a, b, c\}$, $B_P = \{p(a), p(b), p(c)\}$ y $ground(P) = P$.

Si trabajamos en la lógica clásica TWO, hay 8 interpretaciones de Herbrand para P (que se corresponden con los distintos subconjuntos de B_P):

$I_1 = \emptyset$
$I_2 = \{p(a)\}$
$I_3 = \{p(b)\}$
$I_4 = \{p(c)\}$
$I_5 = \{p(a), p(b)\}$
$I_6 = \{p(a), p(c)\}$
$I_7 = \{p(b), p(c)\}$
$I_8 = \{p(a), p(b), p(c)\}$

De ellos, todos son modelos del programa excepto I_2.

Si trabajamos en la lógica $THREE$, hay 27 interpretaciones de Herbrand para P:

$I_1 = (\emptyset, \emptyset)$
$I_2 = (\{p(a)\}, \emptyset)$
$I_3 = (\{p(b)\}, \emptyset)$
$I_4 = (\{p(c)\}, \emptyset)$
$I_5 = (\{p(a), p(b)\}, \emptyset)$
$I_6 = (\{p(a), p(c)\}, \emptyset)$
$I_7 = (\{p(b), p(c)\}, \emptyset)$
$I_8 = (\{p(a), p(b), p(c)\}, \emptyset)$
$I_9 = (\{p(a)\}, \{p(b)\})$
$I_{10} = (\{p(a)\}, \{p(c)\})$
$I_{11} = (\{p(a)\}, \{p(b), p(c)\})$
$I_{12} = (\{p(b)\}, \{p(a)\})$
$I_{13} = (\{p(b)\}, \{p(c)\})$
$I_{14} = (\{p(b)\}, \{p(a), p(c)\})$
$I_{15} = (\{p(c)\}, \{p(a)\})$
$I_{16} = (\{p(c)\}, \{p(b)\})$
$I_{17} = (\{p(c)\}, \{p(a), p(b)\})$
$I_{18} = (\{p(a), p(b)\}, \{p(c)\})$
$I_{19} = (\{p(a), p(c)\}, \{p(b)\})$
$I_{20} = (\{p(b), p(c)\}, \{p(a)\})$
$I_{21} = (\emptyset, \{p(a)\})$
$I_{22} = (\emptyset, \{p(b)\})$
$I_{23} = (\emptyset, \{p(c)\})$

$I_{24} = (\emptyset, \{p(a), p(b)\})$
$I_{25} = (\emptyset, \{p(a), p(c)\})$
$I_{26} = (\emptyset, \{p(b), p(c)\})$
$I_{27} = (\emptyset, \{p(a), p(b), p(c)\})$

De ellos, I_{17} e I_2, por ejemplo, son modelos 3-valorados de P, mientras que I_{11} e I_{23} no lo son.

3.2 Programas definidos. El operador de consecuencia inmediata T_P

Como hemos visto en el ejemplo anterior, un programa lógico puede tener una gran cantidad de modelos de Herbrand y, desde luego, no todos ellos corresponden a la intuición del "modelo natural pretendido" del programa. En esta sección veremos que, sin embargo, para los programas definidos sí es posible de manera sencilla elegir un modelo 2-valorado canónico del programa. Este es un resultado bien conocido y puede verse como el punto de partida de cualquier discusión sobre la semántica declarativa de los programas lógicos.

En el resto del capítulo, trabajaremos siempre con interpretaciones basadas en la lógica clásica, $I \subseteq B_{P,J}$, e identificaremos el orden en el espacio de las interpretaciones con la contención conjuntista: $I \leq K$ si, y solo si, $I \subseteq K$.

Definición 3.7 (Modelos Minimales). *Sean P un programa normal y J una preinterpretación. Un J-modelo $I \subseteq B_{P,J}$ de P es minimal si no existe otro J-modelo K de P tal que $K \subset I$.*

Teorema 3.1. *Sean P un programa definido y J una preinterpretación. Entonces, P tiene un único J-modelo minimal, denominado el modelo mínimo de P y denotado por $LM_J(P)$ (por su nombre en inglés).*

Demostración. Puesto que P es un programa definido, la intersección de cualquier colección no vacía de J-modelos de P es, a su vez, un J-modelo de P. Por tanto, basta considerar $LM_J(P) = \cap\{I : I \text{ es un } J\text{-modelo de } P\}$.

Corolario 3.1. Sea P un programa definido. Entonces P tiene un único modelo de Herbrand minimal, denotado por $LM(P)$, que consideraremos su modelo canónico.

Ejemplo 3.8. El programa GRAFO del ejemplo 3.2 es un programa definido. Es fácil comprobar que el menor modelo de Herbrand de GRAFO es

$LM(GRAFO) = \{nodo(a), nodo(b), nodo(c), arco(a,b), arco(b,c), camino(a,a), camino(b,b), camino(c,c), camino(a,b), camino(b,c), camino(a,c)\}$

que coincide con su "modelo natural esperado".

Por tanto, para los programas lógicos definidos, el problema de determinar su modelo canónico tiene una solución simple y clara: el menor modelo de Herbrand. Más aún, esta situación es especialmente satisfactoria porque es posible calcular dicho modelo mínimo $LM(P)$ mediante un proceso iterativo usando el *operador de consecuencia inmediata* asociado a P, que pasamos a definir a continuación. El operador de consecuencia inmediata fue introducido por Kowalski y van Emden (van Emden and Kowalski, 1976)[7] e históricamente fue el primer operador estudiado en relación a la semántica declarativa de los programas lógicos.

Definición 3.8 (Operador de consecuencia inmediata $T_{P,J}$). *Sea P un programa lógico normal y J una preinterpretación para P. El operador de consecuencia inmediata $T_{P,J}$, es una aplicación*

$$T_{P,J} : \mathcal{P}(B_{P,J}) \to \mathcal{P}(B_{P,J})$$

donde, para cada interpretación $I \subseteq B_{P,J}$, se define $T_{P,J}(I) \subseteq B_{P,J}$ como el conjunto de los $A \in B_{P,J}$ para los que hay alguna cláusula $A \leftarrow L_1, ..., L_n$ en $ground_J(P)$ tal que $I(L_1 \wedge ... \wedge L_n) = \mathbf{t}$.

Definimos asimismo
$T^0_{P,J} = \emptyset$, $T^{i+1}_{P,J} = T_{P,J}(T^i_{P,J})$ para $i \geq 0$, y $T_{P,J} \uparrow \omega = \cup\{T^i_{P,J} : i \geq 0\}$.

Cuando la preinterpretación J es clara por el contexto, en particular cuando trabajamos con interpretaciones de Herbrand, omitiremos el subíndice J y escribiremos simplemente T_P.

Proposición 3.1. Sea P un programa normal y J una preinterpretación para P. Los J-modelos de P son exactamente los puntos prefijos del operador $T_{P,J}$.

Demostración. En primer lugar, sea I un J-modelo de P. Sea $A \in T_{P,J}(I)$. Entonces, existe $A \leftarrow L_1, ..., L_n$ en $ground_J(P)$ tal que $I(L_1 \wedge ... \wedge L_n) = \mathbf{t}$. Como I es un J-modelo de P, se tiene que $I(A \leftarrow L_1, ..., L_n) = \mathbf{t}$. Por lo tanto, $I(A) = \mathbf{t}$, es decir, $A \in I$. Entonces queda demostrado que $T_P(I) \subseteq I$, es decir, I es punto prefijo de T_P.

Ahora suponemos $T_{P,J}(I) \subseteq I$. Sea $C \equiv A \leftarrow L_1, ..., L_n$ una cláusula en $ground_J(P)$. Si $I(L_1 \wedge ... \wedge L_n) = \mathbf{f}$, es claro que $I(C) = \mathbf{t}$. Si $I(L_1 \wedge ... \wedge L_n) = \mathbf{t}$ entonces, por definición, $A \in T_{P,J}(I)$. Luego, $A \in I$ y, por tanto, $I(C) = \mathbf{t}$. En todo caso, $I(C) = \mathbf{t}$ y, por tanto, I es un J-modelo de P.

La proposición anterior es válida para cualquier programa normal. Ahora bien, las propiedades más interesantes de T_P aparecen cuando nos restringimos a programas definidos. Denotamos por I_P el espacio de todas la interpretaciones 2-valoradas de P, esto es, $I_P = \mathcal{P}(B_P)$. Recordemos que $(I_P, \subseteq)$ es un orden parcial completo, donde el supremo y el ínfimo corresponden a la unión y la intersección conjuntista, respectivamente.

Teorema 3.2. *Sea P un programa definido y J una preinterpretación fija para P (omitiremos el subíndice J). Se tiene que:*

1. *T_P es un operador continuo en el espacio I_P.*
2. *T_P tiene un menor punto fijo dado por $T_P \uparrow \omega$ y dicho menor punto fijo coincide con el menor modelo de P, $LM(P)$.*

Demostración. (1): En primer lugar, vamos a ver que T_P es monótona. Sean $I, K \in I_P$ tal que $I \subseteq K$. Sea $A \in T_P(I)$. Por definición, existe una cláusula $A \leftarrow body$ en $ground(P)$ tal que $body \subseteq I$. Como $I \subseteq K$, $body \subseteq K$, entonces $A \in T_P(K)$.

Ahora consideramos $\mathcal{I} = \{\mathcal{I}_\lambda : \lambda \in \Lambda\}$, un conjunto dirigido formado por interpretaciones 2-valoradas. Sea $I = sup\ \mathcal{I}$. Nótese que $T_P(\mathcal{I})$ es un conjunto dirigido ya que T_P es monótono y estamos considerando el orden por inclusión de conjuntos.

Solo nos quedaría probar que $T_P(sup\ \mathcal{I}) = sup\ T_P(\mathcal{I})$ para ver que T_P es continua. La primera contención, $sup\ T_P(\mathcal{I}) \subseteq T_P(sup\ \mathcal{I})$, es sencilla y omitimos los detalles. Vamos a demostrar la segunda contención, $T_P(sup\ \mathcal{I}) \subseteq sup\ T_P(\mathcal{I})$. Sea $J = sup\ \mathcal{I}$. Sea $A \in T_P(J)$. Entonces existe una cláusula C de la forma $A \leftarrow A_1, ..., A_n$ en $ground(P)$ tal que $A_1, ..., A_n$ es verdad en J. Por lo tanto, existen $I_{\lambda_1}, ..., I_{\lambda_n}$ en $\mathcal{I}$ tal que $A_i \in I_{\lambda_i}$ para todo $i = 1, ..., n$. Como $\mathcal{I}$ es dirigido, existe $I_\lambda \in \mathcal{I}$ tal que $I_{\lambda_i} \subseteq I_\lambda$ para todo $i = 1, ..., n$. Finalmente, como $I_\lambda(C) = \mathbf{t}$, $A \in T_P(I_\lambda)$, lo que implica que $A \in sup\ T_P(\mathcal{I})$, como queríamos demostrar.

(2): Puesto que $(I_P, \subseteq)$ es un orden parcial ω-completo (de hecho, completo) y T_P es un operador ω-continuo (de hecho, continuo), por el Teorema del punto fijo de Kleene (Teorema 2.1), el operador T_P tiene un menor punto fijo dado por $T_P \uparrow \omega$ que, además, coincide con su menor punto prefijo. Por tanto, el resultado se sigue aplicando la Proposición 3.1

Ejemplo 3.9 (GRAFO). Vamos a calcular el menor punto fijo para el programa GRAFO del ejemplo 3.2.

$T_P^0 = \emptyset$
$T_P^1 = T_P(T_P^0) = \{nodo(a), nodo(b), nodo(c), arco(a, b), arco(b, c)\}$

$T_P^2 = T_P(T_P^1) = T_P^1 \cup \{camino(a,a), camino(b,b), camino(c,c)\}$
$T_P^3 = T_P(T_P^2) = T_P^2 \cup \{camino(a,b), camino(b,c)\}$
$T_P^4 = T_P(T_P^3) = T_P^3 \cup \{camino(a,c)\}$
$T_P^5 = T_P(T_P^4) = T_P^4 = T_P \uparrow \omega$

Entonces T_P^4 es el menor punto fijo para T_P que coincide con $LM(P)$ calculado en el ejemplo anterior, comprobamos así el Teorema 3.2.

Cerramos esta sección dando una caracterización del modelo canónico de un programa definido en términos de una *función de nivel* para el programa.

Definición 3.9 (Función de nivel). *Sea P un programa lógico normal y J una preinterpretación fija para P (omitiremos el subíndice J). Una función de nivel para P es una aplicación*

$$l : B_P \to \alpha$$

donde α es un ordinal. Esto es, a cada átomo A en B_P se le asocia un ordinal $l(A) < \alpha$.

Aunque escribimos la definición con toda generalidad, la usaremos fundamentalmente para la preinterpretación de Herbrand del programa P. La noción de función de nivel para un programa P será muy útil en varias partes de este trabajo. En el futuro, tendremos que extender una función de nivel a literales negativos. Para ello, y salvo que se diga lo contrario, supondremos que $l(\neg A) = l(A)$ para todo átomo A.

Proposición 3.2. Sea P un programa lógico definido y J una preinterpretación fija para P. Sea M un modelo de P. Son equivalentes:

1. M es el menor modelo de P.
2. Existen un ordinal α y una función de nivel para P, $l : B_P \to \alpha$, tales que para cada átomo $A \in M$, existe una cláusula $A \leftarrow L_1, ..., L_n$ en $ground(P)$ con $M(L_1 \wedge ... \wedge L_n) = \mathbf{t}$ y $l(A_i) < l(A)$ para cada cada $i = 1, \ldots, n$.

Demostración. $(1 \Rightarrow 2)$: Supongamos que M es el menor modelo de P. Por el teorema 3.2, $M = T_P \uparrow \omega$. Para construir la función de nivel requerida, tomamos $\alpha = \omega$ y definimos una aplicación $l : B_P \to \omega$ tal que si $A \notin M$, $l(A) = 0$, y si $A \in M$, $l(A) = min\{n | A \in T_P \uparrow (n+1)\}$. Intuitivamente, $l(A)$ "mide" la iteración n del operador de consecuencia inmediata T_P en la que el átomo A aparece por primera vez en el punto fijo de T_P. Puesto que $\emptyset \subseteq T_P \uparrow 1 \subseteq ... \subseteq T_P \uparrow n \subseteq ... \subseteq T_P \uparrow \omega$ y $T_P \uparrow \omega = \bigcup_{m<\omega} T_P \uparrow m$, podemos concluir que l está bien definida y es fácil ver que se cumple (2).

(2 ⇒ 1): En este caso, suponemos (2) y sea l una función de nivel con las propiedades descritas en (2). Por inducción transfinita sobre el ordinal $\beta < \alpha$, es fácil comprobar que si $\beta = l(A)$ y $A \in M$ entonces $A \in T_P \uparrow (l(A) + 1)$. Por tanto, $M \subseteq T_P \uparrow \omega$ y, puesto que $T_P \uparrow \omega$ es el menor modelo de P, se tiene que $M = T_P \uparrow \omega$.

Ejemplo 3.10. Sea P el siguiente programa definido

```
p(a) ←
q(b) ← p(b)
q(a) ← p(a)
r(a) ← p(a),q(a)
```

Entonces, $B_P = HB(P) = \{p(a), q(a), r(a), p(b), q(b), r(b)\}$.
$T_P^0 = \emptyset$, $T_P^1 = \{p(a)\}$, $T_P^2 = \{p(a), q(a)\}$, $T_P^3 = \{p(a), q(a), r(a)\} = T_P \uparrow \omega$.

La función de nivel para P generada como en la prueba de la Proposición 3.2 sería:

$$l(p(a)) = 0, \quad l(q(a)) = 1, \quad l(r(a)) = 2, \quad l(p(b)) = 0, \quad l(q(b)) = 0, \quad l(r(b)) = 0.$$

Es inmediato comprobar que la función l satisface la propiedad requerida en la Proposición 3.2.

3.3 Modelos Justificados

El operador de consecuencia inmediata T_P nos proporciona un solución muy elegante para definir el modelo canónico de un programa definido. Ahora bien, dicho método no puede extenderse, en general, a un programa normal cualquiera pues, en ese caso, T_P no tiene por qué ser monótono.

Ejemplo 3.11. Consideramos el programa normal

```
p(a) → ¬ q(a)
q(a) → ¬ p(a)
```

$T_P^0 = \emptyset$
$T_P^1 = T_P(T_P^0) = \{p(a), q(a)\}$
$T_P^2 = T_P(T_P^1) = \emptyset$

Si quisiéramos seguir calculando la función T_P, nos daríamos cuenta de que no es monótona ya que oscila así, es decir, $T_P^n = \emptyset$ si n es par y $T_P^n = \{p(a), q(a)\}$ si n es impar. También hemos podido comprobar que T_P no tiene ningún punto fijo.

En este apartado, vamos a estudiar los modelos justificados, no es más que un intento de capturar el modelo natural de un programa lógico. La definición de modelo de un programa P no nos basta para conseguir esa intención, sino que tenemos que añadir más requerimientos a esta noción. A continuación, vamos a mostrar un ejemplo claro para entender mejor esta situación.

Ejemplo 3.12. Sea P el programa lógico definido siguiente:

```
p(a) ←
r(a) ←
r(b) ←
p(c) ←
q(X) ← p(X),r(X)
```

Consideremos los modelos de Herbrand de P siguientes:
$M_1 = \{p(a), q(a), r(a), p(b), q(b), r(b), p(c), q(c), r(c)\}$
$M_2 = \{p(a), q(a), r(a), p(b), q(b), r(b), p(c)\}$
$M_3 = \{p(a), q(a), r(a), r(b), p(c)\}$

Está claro que tanto M_1 como M_2 son dos modelos de P. Sin embargo, ninguno de ellos capta el modelo natural. Si observamos M_2, no hay ninguna regla en el programa P que justifique que $p(b)$ esté en el modelo canónico de P. Lo mismo ocurre para M_1 con los átomos $p(b)$ y $r(c)$. Sin embargo, en su modelo canónico M_3, para cada átomo en M_3 podemos encontrar una regla en $ground(P)$ que justifique su presencia en el modelo canónico. Esto motiva la siguiente definición.

En esta sección, trabajaremos siempre con interpretaciones de Herbrand para la lógica clásica TWO.

Definición 3.10 (Interpretación justificada). *Sea P un programa normal. Una interpretación $I \subseteq B_P$ se dirá justificada si para cada $A \in I$ existe alguna cláusula $A \leftarrow L_1, ..., L_n$ en $ground(P)$ tal que $I(L_1 \wedge ... \wedge L_n) = \mathbf{t}$. Un modelo justificado de P es un modelo de P que, además, es una interpretación justificada.*

Proposición 3.3. Sea P un programa normal.

1. Las interpretaciones justificadas de P son, exactamente, los puntos postfijos del operador T_P.
2. Los modelos justificados de P son, exactamente, los puntos fijos de T_P.

Demostración. (1): Vamos a demostrar la primera parte del teorema. En primer lugar, supongamos que I es una interpretación justificada de P y sea $A \in I$. Por definición, existe una cláusula $A \leftarrow body$ en $ground(P)$ tal que $I(body) = \mathbf{t}$. Por lo tanto, $A \in T_P(I)$. Entonces tenemos que $I \subseteq T_P(I)$, es decir, I es un punto postfijo de T_P.

Supongamos ahora que I es un punto postfijo de T_P, $I \subseteq T_P(I)$. Sea $A \in I$. Entonces, $A \in T_P(I)$ y, por definición del operador, existe $A \leftarrow body$ una cláusula en $ground(P)$ tal que $I(body) = \mathbf{t}$. Por lo tanto, I es un modelo justificado de P, como queríamos demostrar.

(2): Finalmente, para la segunda parte del teorema sólo necesitamos aplicar la Proposición 3.1 y el apartado anterior: una interpretación para P es un modelo justificado de P si y sólo si es tanto punto prefijo como punto postfijo de T_P, es decir, si y sólo si es punto fijo de T_P. ∎

Corolario 3.2. Sea P un programa definido. El menor modelo de Herbrand de P es, en particular, un modelo justificado de P.

Demostración. Se sigue del Teorema 3.2 y de la Proposición anterior. ∎

Ejemplo 3.13 (PIOLÍN). Siguiendo con el ejemplo 3.1, un modelo justificado para ese programa sería el siguiente:

$$M_1 = \{pinguino(piolin), pajaro(bob), pajaro(piolin), vuela(bob)\}.$$

Para comprobarlo, basta ver que $T_{PIOLIN}(M) = M$. Por otra parte, la interpretación:

$$M_2 = \{pinguino(piolin), pajaro(bob), pajaro(piolin), pinguino(bob)\}$$

es un modelo de (se cumple $T_{PIOLIN}(M_2) \subseteq M_2$) pero no está justificado. Obsérvese que $pinguino(bob)$ no aparece en la cabeza de ninguna de las cláusulas de $ground$(PIOLÍN).

Ejemplo 3.14. Sea P el siguiente programa lógico normal:

```
p(a) → ¬ q(a)
q(a) → ¬ p(a)
r(a) → ¬ r(a)
```

Entonces, $T_P^0 = \emptyset$, $T_P^1 = \{p(a), q(a), r(a)\}$, $T_P^2 = \emptyset$, $T_P^3 = \{p(a), q(a), r(a)\}$, ... En general, se puede comprobar que T_P no tiene ningún punto fijo (de hecho, $r(a) \in I$ si y solo si $r(a) \notin T_P(I)$) y, por la Proposición anterior, P no tiene modelos justificados. Sin embargo, las interpretaciones $\{q(a), r(a)\}$ y $\{p(a), r(a)\}$ son dos modelos minimales de P.

Ejemplo 3.15. Sea P el programa lógico normal:

```
p(a) → ¬ q(a)
q(a) → ¬ p(a)
r(a) → r(a)
```

Consideremos las interpretaciones $M_1 = \{p(a)\}$, $M_2 = \{q(a)\}$, $M_3 = \{p(a), r(a)\}$ y $M_4 = \{q(a), r(a)\}$. Es fácil comprobar que todas son puntos fijos del operador T_P y, por tanto, todas son modelos justificados de P. Ahora bien, M_1 y M_2 son modelos minimales de P pero M_3 y M_4 no lo son. En particular, obsérvese que un programa puede tener varios modelos justificados y que no todo modelo justificado de un programa es un modelo minimal.

Cerramos esta sección con un resultado que asegura que, para cierto tipo especial de programas lógicos normales, podemos demostrar existencia y unicidad de modelos justificados. Estos son los llamados programas *localmente jerárquicos*. Su definición se basa de nuevo en la existencia de una cierta función de nivel.

Definición 3.11 (Programas localmente jerárquicos, programas acíclicos). *Un programa lógico normal P se dirá localmente jerárquico si existe una función de nivel*

$$l : B_P \to \alpha,$$

para algún ordinal α, tal que para cada cláusula $A \leftarrow L_1, \ldots, L_n$ en $ground(P)$ y para todo $i = 1, \ldots n$ se tiene $l(A) > l(L_i)$.

Si en la definición anterior puede tomarse $\alpha = \omega$, entonces P se dirá un programa acíclico.

Ejemplo 3.16 (PAR). El ejemplo 3.3 es un programa acíclico. Para probarlo tenemos que dar una función de nivel que cumpla las siguientes condiciones:

$$l(par(a)) < l(par(s(a)) < l(par(s(s(a))) = l(par(s^2(a))) < \cdots < l(par(s^n(a))) < \cdots$$

Por tanto, tomamos l tal que $l(par^k(a))) = k$ para todo $k < \omega$ siendo $\omega = \alpha$.

Ejemplo 3.17 (EXISTEPAR). El siguiente programa lógico normal expresa el conocimiento: 0 es un número natural y par; si un número es natural, su sucesor también; si un número no es par, su sucesor sí lo es. Finalmente, si un número es natural y es par, tenemos que existe par.

```
nat(0)←
nat(s(X))← nat(X)
par(0)←
par(s(X))← ¬ par(X)
existePar← nat(X),par(X)
```

Vamos a probar que el programa es localmente jerárquico pero no acíclico. En primer lugar, vamos a calcular B_P y $ground(P)$:

$B_P = \{nat(0), nat(s(0)), nat(s(s(0))), \ldots, par(0), par(s(0)), par(s(s(0))), \ldots\}$
El conjunto $ground(P)$ consta de las siguientes cláusulas:

```
nat(0) ←
nat(s(0)) ← nat(0)
nat(s(s(0))) ← nat(s(0))
nat(s(s(s(0)))) ← nat(s(s(0)))
   ⋮
par(0) ←
par(s(0)) ← ¬ par(0)
par(s(s(0))) ← ¬ par(s(0))
par(s(s(s(0)))) ← ¬ par(s(s(0)))
   ⋮
existePar← nat(0),par(0)
existePar← nat(s(0)),par(s(0))
existePar← nat(s(s(0))),par(s(s(0)))
   ⋮
```

Vamos a buscar una función de nivel l tal que cumpla las siguientes condiciones:

1. $l(nat(0)) < l(nat(s(0)) < l(nat(s^2(0))) < \cdots < l(nat(s^n(0))) < \ldots$

2. $l(par(0)) < l(par(s(0)) < l(par(s^2(0))) < \cdots < l(par(s^n(0))) < \ldots$
3. $l(existePar)$ tiene que ser mayor estricto que $l(nat(0))$, $l(par(0))$, $l(nat(s(0)))$, $l(par(s(0)))$, ... En resumen, $l(existePar)$ tiene que ser mayor que todo lo demás.

Por lo tanto, podríamos tomar l tal que $l(nat(s^k(0))) = l(par(s^k(0))) = k$ para todo $k \in \mathbb{N}$ y $l(existePar) = \omega$. Queda probado que el programa es localmente jerárquico pero no es acíclico.

Nuestro objetivo será aplicar teoremas de punto fijo generalizados para el operador de T_P y así poder obtener existencia y unicidad de modelos justificados. Para ello, será necesario dotar al espacio de las interpretaciones de Herbrand I_P de estructura de espacio métrico.

Sea P un programa lógico acíclico y sea $l : B_P \to \omega$ una función de nivel para P. Definimos una función

$$d_l : I_P \times I_P \to \mathbb{R}$$

como sigue. Sean $I, J \in I_P$ dos interpretaciones de Herbrand para P. Si $I = J$, entonces $d_l(I, J) = 0$. Si $I \neq J$, entonces $d(I, J) = 2^{-n}$ donde I y J difieren en algún átomo A tal que $l(A) = n$ y coinciden en todos los átomos B tales que $l(B) < n$.

Si el programa P es localmente jerárquico pero no acíclico, la construcción es similar pero la distancia d_l no toma valores en $\mathbb{R}$ sino en un espacio métrico formal representado como $\{2^{-\beta} | \beta \leq \alpha\}$ (omitiremos los detalles técnicos).

Teorema 3.3. *Sean P un programa lógico normal y l una función de nivel para P. Supongamos que*

1. *P es acíclico respecto a l, o bien*
2. *P es localmente jerárquico respecto a l.*

Entonces, el operador T_P tiene un único punto fijo y el programa P tiene un único modelo justificado.

Demostración (Esquema). (1): Consideramos el espacio métrico de las interpretaciones de Herbrand (I_P, d_l) definido más arriba. Entonces:

- (I_d, d_l) es un espacio métrico *completo*. Es decir, toda sucesión de Cauchy respecto a la distancia d_l en I_d es convergente en dicho espacio.

- T_P es una contracción con factor de contracción $1/2$.
  Se cumple que si $I_1, I_2 \in I_P$ con $d_l(I_1, I_2) = 2^{-n}$, entonces

$$d_l(T_P(I_1), T_P(I_2)) \leq 2^{-(n+1)} = (1/2) \cdot d_l(I_1, I_2).$$

Podemos pues aplicar el Teorema del punto fijo de Banach.

Teorema 3.4 (Banach). *Sea (X, d) un espacio métrico completo. Sean $0 \leq \lambda < 1$ y $f : X \to X$ una contracción con factor de contracción λ, esto es, $d(f(x), f(y)) \leq \lambda \cdot d(x, y)$ para todo $x, y \in X$. Entonces, f tiene un único punto fijo.*

En consecuencia, el operador T_P tiene un único punto fijo $I \in I_P$. Por la Proposición 3.3, I es el único modelo justificado del programa acíclico P.

(2): La prueba de (2) es similar pero técnicamente más compleja. Ahora el espacio (I_P, d_l) no es necesariamente un espacio métrico completo y es necesario usar una generalización del teorema del punto fijo de Banach a espacios ultramétricos generalizados completos para esferas, el llamado teorema del punto fijo de Prieβ-Crampe y Ribenboim. Omitimos los detalles técnicos.

3.4 Modelos Estables

Una de las desventajas de la semántica de los modelos justificados que acabamos de estudiar es que incluso los programas definidos pueden tener más de un modelo justificado. Basta considerar un ejemplo tan sencillo como el programa definido dado por la regla

```
p(a) ← p(a)
```

Entonces, $M_1 = \emptyset$ y $M_2 = \{p(a)\}$ son ambos modelos justificados de P.

Por tanto, la noción de modelo justificado no nos permite elegir un modelo natural ni en el caso más simple de los programas definidos.

En esta sección, vamos a definir una nueva semántica declarativa para los programas lógicos que solucionará este problema: los modelos estables (también llamados conjuntos de respuestas en el marco de las aplicaciones de la Programación Lógica).

Veremos que un programa definido sí tiene un único modelo estable (que coincide con su modelo canónico) y que, para muchos programas normales con negación, la noción de modelo estable captura de manera adecuada su modelo natural.

La definición de modelo estable se da en dos pasos. Primero definimos el reducto o reducido de un programa P respecto a una interpretación de Herbrand M para P. En esta sección, trabajaremos siempre en el marco de las interpretaciones de Herbrand y la lógica clásica TWO.

Definición 3.12 (Reducto de Gelfond-Lifschitz P_M). *Sean P un programa normal y $M \in I_P$. El reducto o reducido de P respecto a M, denotado por P_M, es un nuevo programa lógico obtenido a partir de $ground(P)$ como sigue:*

1. *Si $A \leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_k$ está en $ground(P)$ y para algún literal negativo $\neg B_i$ se tiene que $B_i \in M$, se elimina la regla completa del programa P_M.*
2. *Si $A \leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_k$ está en $ground(P)$ y para todo literal negativo $\neg B_i$ se tiene que $B_i \notin M$, se añade la regla $A \leftarrow A_1, \ldots, A_n$ al programa P_M.*

En particular, es claro que P_M es un programa definido (se elimina cualquier ocurrencia de la negación de P_M). Por la tanto, P_M tiene un modelo canónico: su menor modelo de Herbrand $LM(P_M) = T_{P_M} \uparrow \omega$, por los resultados de la Sección 3.2.

Definición 3.13 (Modelo estable). *Sean P un programa normal y $M \in I_P$ una interpretación de Herbrand para P. Se dirá que M es un modelo estable del programa P si $LM(P_M) = M$.*

El siguiente ejemplo fue usado por M. Gelfond y V. Lifschitz en su artículo de 1988, (Gelfond and Lifschitz, 1988)[1], donde se introduce la noción de modelo estable por primera vez.

Ejemplo 3.18 (Gelfond&Lifschitz'1988). Sea P el siguiente programa:

```
p(1,2) ←
q(X) ← p(X,Y), ¬ q(Y)
```

Ahora sustituimos la segunda regla por las instancias básicas. Entonces, $ground(P)$ nos quedaría de la siguiente manera:

```
p(1,2) ←
q(1) ← p(1,1), ¬ q(1)
q(1) ← p(1,2), ¬ q(2)
q(2) ← p(2,1), ¬ q(1)
q(2) ← p(2,2), ¬ q(2)
```

Sea $M_1 = \{p(1,2), q(2)\}$. Entonces el reducto P_{M_1} se define de la siguiente manera:

```
p(1,2) ←
q(1) ← p(1,1)
q(2) ← p(2,1)
```

El mínimo modelo de Herbrand de P_{M_1} es $\{p(1,2)\}$. Puesto que es diferente de M_1, se tiene que M_1 no es un modelo estable de P.

Sea ahora $M_2 = \{p(1,2), q(1)\}$. Entonces el reducto P_{M_2} se define como sigue:

```
p(1,2) ←
q(1) ← p(1,2)
q(2) ← p(2,2)
```

En este caso, el mínimo modelo de Herbrand de P_{M_2} es $\{p(1,2), q(1)\}$, que coincide con M_2. Por lo tanto, M_2 es un modelo estable de P.

Observación 3.1. Es interesante observar que la noción de modelo estable también puede capturarse mediante los puntos fijos de un operador.

Definición 3.14 (Operador de Gelfond-Lifschitz). *Sean P un programa normal. El operador de Gelfond-Lifschitz asociado a P, denotado por GL_P, es una aplicación $GL_P : I_P \to I_P$ definida por:*

$$GL_P(I) = LM(P_I) = T_{P_I} \uparrow \omega,$$

donde I una interpretación de Herbrand para P.

Por las definiciones anteriores, I es un modelo estable de P si I es un punto fijo del operador GL_P.

Proposición 3.4. Sea P un programa normal.

1. El operador de Gelfond-Lifschitz GL_P es antimonótono y, en general, no monótono.
2. Sea I una interpretación de Herbrand para P. Entonces, I es un modelo estable de P si, y solo si, $GL_P(I) = I$.

Demostración. (1): Sean P un programa e I, K interpretaciones para P tales que $I \subseteq K$. Entonces, por definición, $P_K \subseteq P_I$, y por inducción se prueba que $T_{P_K} \uparrow n \subseteq T_{P_I} \uparrow n$ para todo $n \in \mathbb{N}$. Luego, $GL_P(K) = T_{P_K} \uparrow \omega \subseteq T_{P_I} \uparrow \omega = GL_P(I)$. Por lo tanto, GL_P es antimonótona.

Para demostrar que es no monótona, vamos a utilizar el siguiente programa:

```
p← p
p← ¬ p
```

Si $I = \emptyset$, entonces $GL_P(I) = \{p\}$. Por otro lado, si tomásemos $I = \{p\}$, $GL_P(I) = \emptyset$. Por lo tanto, GL_P es, en general, no monótono.

(2): Inmediato por la definición de modelo estable. ∎

Incluimos más ejemplos de este importante concepto.

Ejemplo 3.19. Consideremos el siguiente programa P:

```
hombre(miguel) ←
casado(X) ← hombre(X), ¬ soltero(X)
soltero(X) ← hombre(X), ¬ casado(X)
```

Calculemos $ground(P)$:

```
hombre(miguel) ←
casado(miguel)← hombre(miguel),¬ soltero(miguel)
soltero(miguel) ← hombre(miguel), ¬ casado(miguel)
```

Tomamos $M_1 = \{hombre(miguel), casado(miguel)\}$. Entonces el reducto P_{M_1} quedaría de la siguiente manera:

```
hombre(miguel) ←
casado(miguel)← hombre(miguel)
```

En este caso, el mínimo modelo de Herbrand es $\{hombre(miguel), casado(miguel)\}$ que coincide con M_1, por lo tanto, M_1 es estable.

Si tomásemos, por ejemplo, $M_2 = \{hombre(miguel)\}$, entonces el reducto P_{M_2} quedaría de la siguiente manera:

```
hombre(miguel) ←
casado(miguel)← hombre(miguel)
soltero(miguel)← hombre(miguel)
```

En este caso, $LM(P_{M_2}) = \{hombre(miguel), casado(miguel), soltero(miguel)\}$ que no coincide con M_2, por lo tanto, M_2 no es estable.

Nótese que el programa tiene un segundo modelo estable
$M_3 = \{hombre(miguel), soltero(miguel)\}$.

Ejemplo 3.20. Consideremos el programa P formado por una única regla

```
p(a) ← ¬ p(a)
```

Entonces, P no tiene modelos estables.
Sea M una interpretación para P tal que $p(a) \notin M$, i.e. $M(p(a)) = \mathbf{f}$. Entonces $M(\neg p(a)) = \mathbf{t}$ y $p(a)$ está en el reducto P_M. Por tanto $M \neq LM(P_M)$ porque $p(a) \notin M$ y M no es un modelo estable de P. Sea ahora N una interpretación tal que $p(a) \in N$, i.e. $N(p(a)) = \mathbf{t}$. Entonces, P_N es vacío y $LM(P_N) = \emptyset \neq N$. Luego, N tampoco es un modelo estable de P.

Pasamos ahora a estudiar las principales propiedades teóricas de los modelos estables.

Teorema 3.5. *Sea P un programa lógico normal. Un modelo estable de P es, en particular, un modelo de Herbrand minimal de P.*

Demostración. Sea M un modelo estable de P. Esto significa que M es una interpretación de Herbrand de P y que $M = LM(P_M)$.

En primer lugar, hay que demostrar que M es un modelo de P. Sea R una regla (=cláusula) en $ground(P)$. Si R contiene algún literal $\neg B$ con $B \in M$, entonces el cuerpo R es falso según M y, por tanto, $M(R) = \mathbf{t}$. En caso contrario, R aporta al reducto P_M una nueva regla S obtenida eliminando en R los literales negativos que haya su cuerpo. Obviamente S es una regla de P_M y, como M es modelo minimal de P_M, se tiene que $M(S) = \mathbf{t}$. En consecuencia, también se tiene que $M(R) = \mathbf{t}$ (nótese que el cuerpo de S está contenido en el cuerpo de R).

Ahora tendríamos que demostrar que M es un modelo *minimal* de P. Sea una interpretación N tal que N es modelo de P y $N \subseteq M$. Vamos a demostrar que N es modelo de P_M. Tomamos cualquier regla S de P_M. Sabemos que S se obtiene a partir de otra regla R de $ground(P)$ eliminando los literales negativos de su cuerpo. Por ejemplo, sea $\neg B$ uno de esos literales. Sabemos que $B \notin M$ y, por tanto, tampoco estará en N. Puesto que N es un modelo de P, se tiene que $N(R) = \mathbf{t}$ y, por tanto, $N(S) = \mathbf{t}$ (pues los literales negativos de R son verdaderos en N). Por lo tanto, N es modelo de P_M. Puesto que $M = LM(P_M)$, $N = M$, como queríamos demostrar. ∎

Corolario 3.3. Un programa definido P tiene un único modelo estable, el cual coincide con su menor modelo de Herbrand.

Demostración. En primer lugar, recordemos que un programa lógico definido o positivo es aquel programa en el que no hay literales negativos. En este caso, el reducto P_M sería igual que $ground(P)$ para cualquier interpretación M. Si M es estable, $M = LM(P_M) = LM(P)$ y, por el teorema 3.1, todo programa lógico positivo tiene un único modelo minimal. ∎

Nuestro siguiente objetivo es demostrar que todo modelo estable de un programa P es, en particular, una interpretación justificada del programa. Para ello caracterizaremos la noción de modelo estable en términos de funciones de nivel, y esto nos será útil para probar propiedades de los modelos estables.

Definición 3.15 (Bien-justificada). *Una interpretación I para un programa P se denomina bien-justificada si para algún ordinal α existe una función de nivel*

$$l : B_P \to \alpha$$

tal que, para cada $A \in I$, existe C en $ground(P)$ de la forma $A \leftarrow A_1, ..., A_n, \neg B_1, ..., \neg B_k$ de manera que el cuerpo de C es verdad en I y $l(A_i) < l(A)$ para $i = 1, ..., n$.

Un modelo bien-justificado de P es un modelo de P que, además, es una interpretación bien-justificada de P.

Proposición 3.5. Sean P un programa normal y M una interpretación de Herbrand para P. Son equivalentes:

1. M es un modelo estable de P.
2. M es un modelo bien-justificado de P.

Demostración. [1 $\Rightarrow$ 2] Supongamos que M es un modelo estable de P. Entonces, $GL_P(M) = T_{P_M} \uparrow \omega = M$, es decir, M es el menor modelo del reducto P_M. Por una parte, M es un modelo de P. Por otra parte, P_M es un programa definido y, por la prueba de la Proposición 3.2, M está bien justificado con respecto a la función de nivel l dada por

$$l(A) = min\{n | A \in T_{P_M} \uparrow (n+1)\}$$

para cada $A \in M$ (y $l(A) = 0$ si $A \notin M$).

[2 $\Rightarrow$ 1] Sea M un modelo bien justificado. Entonces para algún ordinal α, existe una función de nivel $l : B_P \leftarrow \alpha$ tal que, para cada $A \in I$, existe $C \in ground(P)$ de la forma $A \leftarrow A_1, ..., A_n, \neg B_1, ..., \neg B_k$ de manera que el cuerpo de C es verdad en M y $l(A_i) < l(A)$ para $i = 1, ..., n$. A partir de aquí tenemos que para todo $A \in M$, existe $A \leftarrow A_1, ..., A_n$ en P_M tal que $M(body) = \mathbf{t}$ y $l(A_i) < l(A)$ para $i = 1, ..., n$. Luego, por la Proposición 3.2, M es el menor modelo para P_M, esto es, $GL_P(M) = T_{P_M} \uparrow \omega = M$. Por lo tanto, M es un modelo estable, como queríamos demostrar. ∎

Teorema 3.6. *Todo modelo estable de un programa normal P es un modelo justificado de P.*

Demostración. Se sigue directamente de la Proposición anterior porque toda interpretación bien-justificada es, en particular, una interpretación justificada. ∎

Observación 3.2. Hemos demostrado pues que todo modelo estable de un programa normal P es un modelo minimal y justificado de P. Sin embargo, el recíproco no es cierto, como muestra el siguiente ejemplo. Sea P el programa:

```
p(a) ← ¬ p(a)
p(a) ← p(a)
```

Entonces, $M = \{p(a)\}$ es un modelo minimal y justificado de P. Sin embargo, M no es un modelo estable de P. Obsérvese que el reducto P_M es

```
p(a) ← p(a)
```

cuyo menor modelo de Herbrand es $\emptyset$, que no coincide con M.

Ejemplo 3.21 (PIOLÍN 2). Consideremos el siguiente programa que resulta de añadirle al de Piolín la última cláusula.

```
pinguino(piolin)←
pajaro(bob)←
pajaro(X)← pinguino(X)
vuela(X)← pajaro(X),¬ pinguino(X)
pinguino(bob)← pinguino(bob)
```

El siguiente conjunto M es un modelo justificado para el primer ejemplo de Piolín 3.1 y lo es también para éste.
$M = \{pinguino(piolin), pajaro(bob), pajaro(piolin), vuela(bob)\}$
Además, tiene otro modelo justificado que es:
$M' = \{pinguino(piolin), pinguino(bob), pajaro(piolin), pajaro(bob)\}$.

M es un modelo bien justificado para este programa pero M' no lo es.

Ejemplo 3.22 (PIOLÍN 3). Consideremos el siguiente programa:

```
aguila(piolin)← ¬ pinguino(piolin)
pinguino(piolin)← ¬ aguila(piolin)
pajaro(X)← aguila(X)
pajaro(X)← pinguino(X)
vuela(X)← pajaro(X),¬ pinguino(X)
```

Este programa tiene dos modelos bien justificados:
$M_1 = \{aguila(piolin), pajaro(piolin), vuela(piolin)\}$
$M_2 = \{pinguino(piolin), pajaro(piolin)\}$

4 | Modelos de Fitting

En el capítulo anterior, hemos estudiado el concepto de modelo estable, que es uno de los más importantes en la semántica formal de los programas lógicos normales. Dicho concepto resulta satisfactorio pues, en muchos casos, captura nuestra intuición sobre el modelo natural de un programa lógico y, por otra parte, la noción de modelo estable tiene buenas propiedades matemáticas. Sin embargo, un programa normal puede tener más de un modelo estable, hecho que supone una debilidad.

Sería deseable disponer de una semántica declarativa que nos permitiese asociar de manera natural a cada programa lógico normal un *único* modelo canónico. Una manera de conseguir este objetivo es apoyarnos en la lógica 3-valorada $THREE$, en lugar de en la lógica clásica TWO.

En este capítulo, estudiaremos una nueva noción de semántica de un programa normal para elegir exactamente un único modelo: los llamados *modelos de Fitting* (también conocidos bajo el nombre de *semántica de Kripke-Kleene*), introducidos en (Fitting, 1985)[4]. Para ello, haremos uso de la lógica fuerte 3-valorada de Kleene y del orden del conocimiento, $\leq_k$, ambos definidos en el Capítulo 2.

En particular, en este capítulo trabajaremos con interpretaciones de Herbrand 3-valoradas ($\mathbf{t}, \mathbf{f}, \mathbf{u}$) y, por regla general, usaremos conjuntos signados para representar dichas interpretaciones 3-valoradas.

Nuestro punto de partida es la definición de un nuevo operador asociado a un programa lógico normal. Pero, en esta ocasión, dicho operador actuará sobre el espacio de las interpretaciones 3-valoradas provisto del orden del conocimiento $\leq_k$.

4.1 El operador de Fitting Φ_P

Dado un programa lógico normal P, denotaremos por $I_{P,3}$ al espacio de las interpretaciones de Herbrand 3-valoradas para P. Dotaremos a $I_{P,3}$ del orden del conocimiento $\leq_k$. Esto es, si $I = (I_t, I_f)$ y $K = (K_t, K_f)$ son dos elementos de $I_{P,3}$, entonces $I \leq_k K$ si, y solo si, $I_t \subseteq K_t$ e $I_f \subseteq K_f$ (véase la Proposición 2.4). Se recuerda que $(I_{P,3}, \leq_k)$ no es un retículo completo (carece de elemento máximo) pero sí es un semi-retículo superior completo.

La definición del operador de Fitting se da en dos pasos.

Definición 4.1 (Operador de Fitting). *Sea P un programa lógico normal. Definiremos el operador de Fitting asociado a P, Φ_P, a partir de otros dos operadores.*

En primer lugar, definimos

$$T'_P : I_{P,3} \to B_P$$

donde $T'_P(I)$ es el conjunto formado por todos los átomos $A \in B_P$ para los que existe al menos una cláusula de la forma $A \leftarrow body$ en $ground(P)$ tal que $I(body) = \mathbf{t}$ (con respecto a la lógica 3-valorada $THREE$).

Por otro lado, definimos

$$F_P : I_{P,3} \to B_P$$

donde $F_P(I)$ es el conjunto de todos átomos $A \in B_P$ tales que para todas las cláusulas de la forma $A \leftarrow body$ en $ground(P)$, se tiene que $I(body) = \mathbf{f}$ (con respecto a la lógica 3-valorada $THREE$).

Finalmente, definimos Φ_P como sigue:

$$\Phi_P : I_{P,3} \to I_{P,3}$$

$$\Phi_P(I) = T'_P(I) \cup \neg F_P(I)$$

donde $\neg F_P(I) = \{\neg A \mid A \in F_P(I)\}$ y usamos por tanto conjuntos signados para representar las interpretaciones de Herbrand 3-valoradas.

Observación 4.1. Es importante tener en cuenta que, para cualquier interpretación $I \in I_{P,3}$, se tiene que $A \in \Phi_P(I)$ si A es un hecho de $ground(P)$, mientras que $\neg A \in \Phi_P(I)$ si no hay ninguna cláusula en $ground(P)$ cuya cabeza sea el átomo A.

Ilustremos la definición anterior con un ejemplo.

Ejemplo 4.1. [PIOLÍN] Volviendo al ejemplo 3.1, el conjunto $ground(P)$ consta de las clásulas

```
pinguino(piolin) ←
pajaro(bob) ←
pajaro(bob) ← pinguino(bob)
pajaro(piolin) ← pinguino(piolin)
vuela(bob) ← pajaro(bob),¬pinguino(bob)
vuela(piolin) ← pajaro(piolin),¬pinguino(piolin)
```

Vamos a calcular $\Phi_P(I)$ tomando $I = \emptyset$.

$T'_P(\emptyset) = \{pinguino(piolin), pajaro(bob)\}$
$F_P(\emptyset) = \{pinguino(bob)\}$
$\Phi_P(\emptyset) = \{pinguino(piolin), pajaro(bob), \neg pinguino(bob)\}$

Calculemos ahora $\Phi_P(\Phi_P(\emptyset))$.

$T'_P(\Phi_P(\emptyset)) = \{pinguino(piolin), pajaro(bob), pajaro(piolin), vuela(bob)\}$
$F_P(\Phi_P(\emptyset)) = \{pinguino(bob), vuela(piolin)\}$
$\Phi_P(\Phi_P(\emptyset)) = \{pinguino(piolin), pajaro(bob), pajaro(piolin), vuela(bob),$
$\neg pinguino(bob), \neg vuela(piolin)\}$

Es fácil comprobar que $\Phi_P(\Phi_P(\Phi_P(\emptyset))) = \Phi_P(\Phi_P(\emptyset))$. Esto es, hemos alcanzado un punto fijo del operador de Fitting Φ_P. Obsérvese además que, en este caso, ningún átomo tiene el valor indeterminado **u** en el punto fijo y, por tanto, $\Phi_P(\Phi_P(\emptyset))$ puede verse como una interpretación clásica 2-valorada. Además, el punto fijo es un modelo de P.

Ejemplo 4.2. Consideremos el programa P

```
p(b) ← ¬ p(a)
p(a) ← ¬ p(b)
```

Calculemos $\Phi_P(\emptyset)$.

$T'_P(\emptyset) = \emptyset \qquad F_P(\emptyset) = \emptyset \qquad \Phi_P(\emptyset) = \emptyset$

Luego hemos alcanzado un punto fijo de ϕ_P: la interpretación vacía, que asigna tanto a $p(a)$ como a $p(b)$ el valor de verdad indeterminado **u** de la lógica trivalorada $THREE$.

En este caso, el punto fijo no puede verse como una interpretación clásica, pero sí es un modelo de P en el sentido de la lógica $THREE$.

En los dos ejemplos anteriores, los puntos fijos calculados para Φ_P resultan ser modelos (en la lógica fuerte de Kleene) del correspondiente programa P. Como muestra la siguiente proposición, este hecho no es casualidad.

Proposición 4.1. Sea P un programa lógico normal. Los modelos 3-valorados para P son, exactamente, los puntos prefijos de Φ_P con respecto al orden de la verdad, $\leq_t$.

Demostración. Sea M una interpretación de Herbrand 3-valorada de P.

En primer lugar, vamos a suponer que M es un punto prefijo de Φ_P respecto a $\leq_t$, es decir, $\Phi_P(M) \leq_t M$. Hay que probar que M es un modelo 3-valorado para P. Podemos escribir $\Phi_P(M) = (I_1, I_2)$ y $M = (K_1, K_2)$ (primero colocamos los átomos verdaderos y después los falsos, los indeterminados no aparecen). Sea $A \leftarrow bod\,y$ una claúsula en $ground(P)$.

Caso 1: Supongamos que $\Phi_P(M)(A) = \mathbf{u}$. Entonces, $A \notin I_1$ y $A \notin I_2$ y, puesto que $\Phi_P(M) \leq_t M$, $I_1 \subseteq K_1$ y $K_2 \subseteq I_2$. Luego, de $A \notin I_2$ se sigue que $A \notin K_2$ y, por tanto, $M(A) \neq \mathbf{f}$. Entonces, $M(A) = \mathbf{u}$ o $M(A) = \mathbf{t}$. Puesto $\Phi_P(M)(A) \neq \mathbf{t}$, no puede haber ninguna cláusula en $ground(P)$ de la forma $A \leftarrow bod\,y$ tal que $M(bod\,y) = \mathbf{t}$. Entonces, $M(bod\,y) = \mathbf{f}$ o $M(bod\,y) = \mathbf{u}$. A partir de la definición de la función de verdad de la implicación para la lógica fuerte de Kleene (ver Capítulo 2), se tiene que $A \leftarrow bod\,y$ es verdad en M.

Caso 2: Supongamos ahora que $\Phi_P(M)(A) = \mathbf{t}$. Entonces, $A \in I_1$, y puesto $I_1 \subseteq K_1$, $A \in K_1$ también. Por tanto, $M(A) = \mathbf{t}$. En consecuencia, independientemente de que $bod\,y$ sea verdadero, falso o indefinido en M, $M(A) \geq_t M(bod\,y)$ y la cláusula $A \leftarrow bod\,y$ es verdad en M.

Caso 3: Por último, supongamos que $\Phi_P(M)(A) = \mathbf{f}$. Por definición, $M(bod\,y) = \mathbf{f}$ para todas las cláusulas de la forma $A \leftarrow bod\,y$ en $ground(P)$. Luego, independientemente de que A sea verdadero, falso o indefinido en M, se tiene que $M(A) \geq_t M(bod\,y)$ y la cláusula $A \leftarrow bod\,y$ es verdad en M.

La implicación opuesta se demuestra de manera similar y omitimos los detalles. ∎

De la Proposición anterior, inmediatamente se sigue que

Corolario 4.1. Sea P un programa lógico normal. Los puntos fijos del operador de Fitting Φ_P son modelos de Herbrand 3-valorados de P.

El propósito de los dos siguientes ejemplos es mostrar que ninguna de las dos implicaciones de la Proposición anterior es cierta si consideramos el orden del conocimiento $\leq_k$ en lugar del orden de la verdad $\leq_t$.

Ejemplo 4.3. Consideremos el siguiente programa P:

```
p← ¬ q
p← ¬ r
q← q
r← r
```

Entonces, $B_P = \{p, q, r\}$ y $P = ground(P)$. Consideramos una interpretación de Herbrand M tal que

$$M(p) = \mathbf{f},\ M(q) = \mathbf{u},\ M(r) = \mathbf{t},$$

que podríamos escribir también como $M = (\{r\}, \{p\})$ o como $M = \{r, \neg p\}$ si usamos un conjunto signado. Vamos a calcular $\Phi_P(M)$:

$T'_P(M) = \{r\}$
$F_P(M) = \{\}$

Entonces $\Phi_P(M) = (\{r\}, \{\})$. Podemos observar que $\Phi_P(M) \leq_k M$ (pues $\{r\} \subseteq \{r\}$ y $\{\} \subseteq \{p\}$) y, por tanto, M es un punto prefijo del operador de Fitting respecto al orden $\leq_k$. Sin embargo, aplicando directamente las definiciones, M no es modelo del programa P, pues $M(\neg q \to p) = \mathbf{f}$.

Ejemplo 4.4. Consideremos ahora el siguiente programa P:

```
p← ¬ q
q← ¬ p
r← r
```

Tomemos M tal que $M(p) = \mathbf{t}$, $M(q) = \mathbf{u}$, $M(r) = \mathbf{t}$ (esto es, $M = \{p, r\}$ como conjunto signado). Es fácil comprobar que M es un modelo de P. Vamos a calcular ahora $\Phi_P(M)$:

$T'_P(M) = \{r\}$
$F_P(M) = \{q\}$

Entonces $\Phi_P(M) = \{r, \neg q\}$. Por lo tanto, M no es un punto prefijo del operador de Fitting respecto al orden $\leq_k$, ya que $\{q\} \not\subseteq \{\}$.

La siguiente proposición es de vital importancia pues garantizará la existencia de un menor punto fijo del operador de Fitting Φ_P para cualquier programa normal P.

Proposición 4.2. Sea P un programa lógico normal. Entonces Φ_P es un operador monótono en el espacio $I_{P,3}$ con el orden del conocimiento, $\leq_k$.

Demostración. Sean $M, N \in I_{P,3}$ con $M \leq_k N$. Tenemos que demostrar que $\Phi_P(M) \leq_k \Phi_P(N)$. Representamos las interpretaciones 3-valoradas como conjuntos signados y, por tanto, el orden $\leq_k$ corresponde a la inclusión conjuntista $\subseteq$.

Supongamos primero que el átomo A está en $\Phi_P(M)$. Entonces, $A \in T'_P(M)$ y, por tanto, existe alguna cláusula $A \leftarrow body$ en $ground(P)$ tal que $M(body) = \mathbf{t}$. En consecuencia, todo literal L_i en $body$ debe ser verdadero en M y por tanto pertenece al conjunto signado M. Puesto que $M \subseteq N$, todo literal L_i en $body$ es también verdadero en N y $N(body) = \mathbf{t}$. Por tanto, $A \in T'_P(N) \subseteq \Phi_P(N)$.

Supongamos ahora que el átomo negado $\neg A$ está en $\Phi_P(M)$. Entonces, $A \in F_P(M)$ y, por definición del operador de Fitting, $M(body) = \mathbf{f}$ para todas las cláusulas básicas de la forma $A \leftarrow body$ en $ground(P)$. Sea una tal cláusula. Al menos un literal $L_i \in body$ es falso en M. Puesto que $M \subseteq N$, dicho literal L_i es también falso en N y, por tanto, $N(body) = \mathbf{f}$. Luego, $A \in F_P(N)$ y $\neg A \in \Phi_P(N)$.

El espacio $I_{P,3}$ con el orden del conocimiento $\leq_k$ no es un retículo completo (carece de elemento máximo) y, por tanto, no podemos aplicar el teorema del punto fijo Knaster-Tarski en toda su generalidad (el cual garantiza la existencia de tanto un menor como un mayor punto fijo). Ahora bien, el espacio $I_{P,3}$ sí es un semi-retículo superior completo y, como se desprende de la prueba dada del teorema de Knaster-Tarski en el capítulo 2 (Teorema 2.2), dicha estructura es suficiente para garantizar la existencia de un *menor punto fijo* de un operador monótono (aunque no garantiza la existencia de un mayor punto fijo). Y dicho menor punto fijo puede obtenerse mediante potencias ordinales a partir de la interpretación 3-valorada vacía.

Toda esta discusión justifica la siguiente definición, fundamental para el contenido del presente capítulo.

Definición 4.2 (Modelos de Fitting). *Sea P un programa lógico normal. El operador de Fitting asociado a P, Φ_P, tiene un menor punto fijo, el cual es una potencia ordinal $\Phi_P \uparrow \alpha$, para algún ordinal α. Dicho menor punto fijo se denomina modelo de Fitting para P (o también modelo de Kripke-Kleene de P).*

Obsérvese que la Proposición 4.2 garantiza la existencia del modelo de Fitting para cualquier programa normal P y la Proposición 4.1 garantiza que el modelo de Fitting es, realmente, un modelo 3-valorado de P.

En el Ejemplo 4.1 vimos que el programa PIOLÍN posee un modelo de Fitting total (que puede interpretarse como un modelo clásico). En el siguiente ejemplo se muestra que añadir una cláusula aparentemente no significativa como

`pinguino(bob)` ← `pinguino(bob)`

puede variar completamente el cálculo del modelo de Fitting.

Ejemplo 4.5 (PIOLÍN 2). Consideremos el segundo programa de Piolín (Ejemplo 3.21) en el que hemos añadido la cláusula `pinguino(bob)` ← `pinguino(bob)`.

Sea $B_P = \{pinguino(bob), pinguino(piolin), pajaro(bob), pajaro(piolin), vuela(bob), vuela(piolin)\}$. $Ground(P)$ está formado por las cláusulas:

```
pinguino(piolin) ←
pajaro(bob) ←
pajaro(bob) ← pinguino(bob)
pajaro(piolin) ← pinguino(piolin)
vuela(bob) ← pajaro(bob),¬pinguino(bob)
vuela(piolin) ← pajaro(piolin),¬pinguino(piolin)
pinguino(bob) ← pinguino(bob)
```

Calculemos el modelo de Fitting de P. Empezamos con la interpretación trivaluada $I = \emptyset$ (vista como un conjunto signado).

$T'_P(\emptyset) = \{pinguino(piolin), pajaro(bob)\}$
$F_P(\emptyset) = \{\}$
$\Phi_P(\emptyset) = \{pinguino(piolin), pajaro(bob)\}$

$T'_P(\Phi_P(\emptyset)) = \{pinguino(piolin), pajaro(bob), pajaro(piolin)\}$
$F_P(\Phi_P(\emptyset)) = \{vuela(piolin)\}$
$\Phi_P(\Phi_P(\emptyset)) = \{pinguino(piolin), pajaro(bob), pajaro(piolin), \neg vuela(piolin)\} = M$

Es fácil comprobar que $\Phi_P(M) = M$. Luego, M es el modelo de Fitting de P. En este caso, el valor de verdad del átomo $pinguino(bob)$ es indeterminado y, por tanto, el modelo de Fitting no define un modelo clásico.

Ejemplo 4.6 (PIOLÍN 3). Consideremos el programa 3.22. Tenemos que tener en cuenta, antes de nada, el conjunto B_P y $ground(P)$:
$B_P = \{aguila(piolin), pajaro(piolin), pinguino(piolin), vuela(piolin)\}$ y $ground(P)$ está formado por las siguientes cláusulas:

```
aguila(piolin) ← ¬ pinguino(piolin)
pinguino(piolin) ← ¬ aguila(piolin)
pajaro(piolin) ← aguila(piolin)
pajaro(piolin) ← pinguino(piolin)
vuela(piolin) ← pajaro(piolin), ¬ pinguino(piolin)
```

Para calcular Φ_P, necesitamos primero T'_P y F_P. Empezamos con la interpretación trivaluada $I = \emptyset$.

$T'_P(\emptyset) = \emptyset$
$F_P(\emptyset) = \emptyset$
$\Phi_P(\emptyset) = T'_P(\emptyset) \cup \neg F_P(\emptyset) = \emptyset = I$

Por tanto, hemos encontrado el punto fijo del operador de Fitting en su primer paso. En consecuencia, el modelo de Fitting de este programa es el conjunto signado $\emptyset$ o, equivalentemente, la interpretación $I = (\emptyset, \emptyset)$ o la valoración de verdad que asocia a todos los átomos el valor indeterminado.

Nótese que el programa posee dos modelos estables debido a la simetría entre los átomos $aguila(piolin)$ y $pinguino(piolin)$. Típicamente, esta situación se traduce en modelos de Fitting no totales, donde una parte de la base B_P recibe el valor de verdad indeterminado.

El propósito del siguiente ejemplo es mostrar que el operador de Fitting Φ_P es monótono pero, en general, no es ω-continuo.

Ejemplo 4.7. Consideremos el siguiente programa P formado por las siguientes cláusulas:

```
p(s(X)) ← p(X)
q ← p(X)
```

Lo primero que tenemos que hacer es calcular $ground(P)$. Antes, es interesante observar que, ya que el programa contiene símbolos de función, el universo de Herbrand es infinito y, por tanto, la base de Herbrand también.

$HU(P) = \{0, s(0), s(s(0)), s(s(s(0))), ...\}$
$B_P = \{q, p(0), p(s(0)), p(s(s(0))), ...\}$

El conjunto $ground(P)$ es también infinito:

$$
\begin{aligned}
&\texttt{p(s(0))} \leftarrow \texttt{p(0)}\\
&\texttt{p(s(s(0)))} \leftarrow \texttt{p(s(0))}\\
&\texttt{p(s(s(s(0))))} \leftarrow \texttt{p(s(s((0))))}\\
&\vdots\\
&\texttt{q} \leftarrow \texttt{p(0)}\\
&\texttt{q} \leftarrow \texttt{p(s(0))}\\
&\texttt{q} \leftarrow \texttt{p(s(0))}\\
&\vdots
\end{aligned}
$$

Vamos a empezar a iterar el operador de Fitting para este programa:

$\Phi_P \uparrow 0 = \emptyset$
$\Phi_P \uparrow 1 = \Phi_P(\emptyset) = \{\neg p(0)\}$
$\Phi_P \uparrow 2 = \Phi_P(\Phi_P \uparrow 1) = \{\neg p(0), \neg p(s(0))\}$
$\vdots$
$\Phi_P \uparrow \omega = \bigcup_{\alpha<\omega} \Phi_P \uparrow \alpha = \{\neg p(0), \neg p(s(0)), \neg p(s(s(0))), \neg p(s(s(s(0)))), ...\}$

$\Phi_P \uparrow \omega$ aún no es punto fijo del operador de Fitting, necesitamos aplicar el operador un pasos más para obtenerlo:
$\Phi_P \uparrow \omega + 1 = \Phi_P(\Phi_P \uparrow \omega) = \{\neg p(0), \neg p(s(0)), \neg p(s(s(0))), \neg p(s(s(s(0)))), ..., \neg q\}$

Con este programa, observamos que el operador de Fitting no es, en general, ω-continuo (si lo fuese, por el teorema de punto fijo de Kleene alcanzaría su menor punto fijo en, a lo más, ω pasos) y, por tanto, para obtener el menor punto fijo del operador no siempre basta con iterar ω veces. En este caso, hemos iterado $\omega + 1$ veces el operador Φ_P. Pero el comportamiento puede ser mucho peor: $\Phi_P \uparrow \alpha$ puede necesitar hasta ω_1^{CK} pasos para alcanzar un punto fijo, donde ω_1^{CK} (=ordinal de Church-Kleene) es

un ordinal numerable muy grande; de hecho, es el menor ordinal que no puede ser descrito de manera computable.

4.2 Modelos de Fitting totales

Como hemos visto en los ejemplos anteriores, el modelo de Fitting asociado a un programa lógico P puede dejar ciertos átomos de la base de Herbrand de P con el valor de verdad indeterminado o bien decidir todo átomo de B_P. Cuando esto segundo ocurre, el comportamiento del operador de Fitting es especialmente bueno porque el modelo de Fitting obtenido puede verse como un modelo clásico del programa. De hecho, como veremos a continuación, esta situación es especialmente satisfactoria.

Definición 4.3. *Sean P un programa lógico normal y $v : B_P \to THREE$ una interpretación 3-valorada. Diremos que v es una interpretación total si $v(A) \neq \boldsymbol{u}$ para todo átomo $A \in B_P$. En caso contrario, v se dirá una interpretación parcial.*

Obsérvese que las interpretaciones totales son los elementos maximales del espacio $I_{P,3}$ con el orden del conocimiento $\leq_k$, y que existe una correspondencia biyectiva entre interpretaciones 3-valoradas maximales y modelos clásicos 2-valorados.

Proposición 4.3. Sea P un programa lógico normal. Entonces, se tiene que $\Phi_P(I)^+ \subseteq T_P(I^+) \subseteq B_P \setminus \Phi_P(I)^-$ para toda interpretación 3-valorada de P, $I \in I_{P,3}$. En consecuencia, el operador Fitting transforma interpretaciones totales en interpretaciones totales y coincide con el operador de consecuencia inmediata T_P sobre interpretaciones totales.

Demostración. Sea $I = I^+ \cup \neg I^-$ una interpretación 3-valorada (vista como un conjunto signado). Vamos a demostrar la primera inclusión. Sea $A \in \Phi_P(I)^+$. Existe una cláusula C en $ground(P)$ de la forma $A \leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_k$ tal que el cuerpo es verdad en I. Entonces, $A_i \in I^+$ y $B_j \in I^-$ y, por tanto, $A_i \in I^+$ y $B_j \notin I^+$ para todo $i = 1, \ldots, n$ y para todo $j = 1, \ldots, k$. Luego, el cuerpo de C es verdad en la interpretación 2-valorada I^+, lo que implica que $A \in T_P(I^+)$.

Para la segunda inclusión, recordemos que $A \in \Phi_P(I)^-$ si, y sólo si, para todas las cláusulas de la forma $A \leftarrow body$ en $ground(P)$, $I(body) = \mathbf{t}$. En este caso, se tiene que uno de los literales del cuerpo es falso, es decir, o bien algún $A_i \in I^-$ o bien algún $B_j \in I^+$. Luego, que es algún $A_i \notin I^+$ o bien algún $B_j \in I^+$. Por tanto, tenemos también que el cuerpo es falso según la interpretación 2-valorada I^+, lo que

implica que $A \notin T_P(I^+)$. Entonces, $\Phi_P(I)^- \subseteq B_P \setminus T_P(I^+)$, que es lo que queríamos demostrar.

Por último, en el caso en el que I fuera total, tendríamos que $B_P \setminus \Phi_P(I)^- = \Phi_P(I)^+ = T'_P(I) = T_P(I^+)$.

Como consecuencia inmediata de la Proposición anterior, obtenemos que los modelos de Fitting totales son siempre justificados.

Corolario 4.2. Sea P un programa normal. Si el modelo de Fitting de P es una interpretación total, entonces dicho modelo de Fitting es un modelo justificado de P.

Demostración. Sea M un tal modelo de Fitting total. Por la Proposición anterior, $\Phi_P(M)^+ = T_P(I^+)$ y, por tanto, I^+ es un punto fijo del operador de consecuencia inmediata T_P. Pero vimos en el capítulo anterior que los puntos fijos de T_P son, exactamente, los modelos justificados de P.

De hecho, incluso más es cierto. Aunque no incluimos su prueba, que es técnicamente más compleja, se tiene el siguiente resultado.

Teorema 4.1. *Sea P un programa normal con un modelo de Fitting total. Entonces, P tiene un único modelo estable y dicho único modelo estable coincide con el modelo de Fitting de P.*

Si retomamos el ejemplo inicial de este capítulo (Ejemplo 4.1), podemos ver ahora que no era accidente el hecho de que el modelo de Fitting total del programa PIOLÍN allí obtenido coincida con su único modelo estable.

Cerramos este capítulo con la observación de que el recíproco del Teorema anterior no es cierto. Un programa con un único modelo estable no tiene por qué tener un modelo de Fitting total.

Ejemplo 4.8. Sea P el siguiente programa:

```
p(a) ← ¬ q(a)
q(a) ← ¬ p(a)
p(a) ← ¬ p(a)
```

Es fácil comprobar que $\{p(a)\}$ es el único modelo estable del programa P. Sin embargo,

$\Phi_P \uparrow 0 = \emptyset$
$\Phi_P \uparrow 1 = \Phi_P(\emptyset) = \emptyset.$

Por lo tanto, el modelo de Fitting de este programa es la interpretación 3-valorada vacía $I = \emptyset$, que, por supuesto, no es una interpretación total.

5 Programas localmente estratificados. Modelos Perfectos

En el capítulo anterior, usando modelos 3-valorados en lugar de modelos clásicos, pudimos asociar a cada programa lógico normal P un *único* modelo canónico (el menor punto fijo del operador de Fitting asociado), evitando de esta manera la ambigüedad presente en la semántica de los programas normales basada en los modelos estables, donde un programa normal puede tener varios estables. Aunque, y como punto débil de la semántica basada en modelos de Fitting, no todo programa normal posee un modelo de Fitting *total*.

En el presente capítulo, se propone una manera alternativa de afrontar el problema de la ambigüedad en la semántica de los programas normales, sin tener que abandonar los modelos de Herbrand clásicos basados en la lógica TWO. En lugar de variar la noción de modelo para conseguir la unicidad, impondremos ciertas restricciones sintácticas a los programas lógicos de manera que un programa que satisfaga estas restricciones tenga asociado un único modelo canónico. Dichas condiciones sintácticas dan lugar a la importante clase de los *programas localmente estratificados*. Asociada a estos programas, introducimos también en este capítulo la semántica de los *modelos perfectos*.

El principal resultado que presentaremos en este capítulo es el teorema que dice que todo programa lógico normal localmente estratificado posee un *único modelo estable*. Luego para esa clase de programas lógicos el principal punto débil de la semántica de los modelos estables desaparece.

5.1 Programas localmente estratificados

Comenzamos con la definición de la clase de programas normales que será de nuestro interés.

Definición 5.1 (Programas localmente estratificados). *Un programa lógico P se dirá localmente estratificado si existe una función de nivel $l : B_P \to \alpha$, para algún ordinal α, tal que para cada cláusula*

$$A \leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m$$

en $ground(P)$ se cumplen las siguientes condiciones

(S1) $l(A) \geq l(A_i)$ *para* $i = 1, \ldots, n$.
(S2) $l(A) > l(B_i)$ *para* $i = 1, \ldots, m$.

Definición 5.2 (Programas estratificados). *Un programa lógico P se dirá estratificado si es localmente estratificado y, además, para todo par de átomos $A, B \in B_P$ con el mismo símbolo de predicado se tiene que $l(A) = l(B)$, donde l es la función de nivel respecto a la cual el programa es localmente estratificado.*

Nótese que para un programa estratificado P siempre podemos considerar una función de nivel que tome valores en un ordinal *finito* (P está formado por un conjunto finito de cláusulas y solo pueden aparecer un número finito de símbolos de predicado en su lenguaje asociado). En cambio, un programa localmente estratificado P puede necesitar, en general, ordinales infinitos para sus funciones de nivel (aunque el lenguaje asociado a P es finito, la base de Herbrand B_P es infinita si el programa contiene símbolos de función).

Ejemplo 5.1. Sea P el siguiente programa lógico normal. Nuestro objetivo es demostrar que P es un programa localmente estratificado dando una función de nivel para $ground(P)$.

```
seta(a)←
seta(b)←
gris(a)← ¬ rojo(a)
rojo(b)← ¬ gris(b)
venenosa(X)← seta(X),rojo(X)
comestible(X)← seta(X),¬ venenosa(X)
```

En primer lugar, necesitamos calcular $ground(P)$:

```
seta(a)←
seta(b)←
gris(a)← ¬ rojo(a)
rojo(b)← ¬ gris(b)
venenosa(a)← seta(a),rojo(a)
venenosa(b)← seta(b),rojo(b)
comestible(a)← seta(a),¬ venenosa(a)
comestible(b)← seta(b),¬ venenosa(b)
```

Sea B_P el conjunto de todos los átomos sin variables
B_P={$seta(a)$, $seta(b)$, $rojo(a)$, $rojo(b)$, $gris(a)$, $gris(b)$, $venenosa(a)$, $venenosa(b)$, $comestible(a)$, $comestible(b)$}

El siguiente paso será calcular la función de nivel l:
$l(seta(a)) = v_1$, $l(seta(b)) = v_2$, $l(rojo(a)) = v_3$, $l(rojo(b)) = v_4$, $l(gris(a)) = v_5$, $l(gris(b)) = v_6$, $l(venenosa(a)) = v_7$, $l(venenosa(b)) = v_8$, $l(comestible(a)) = v_9$, $l(comestible(b)) = v_{10}$.

Usando la Definición 5.1, obtenemos un sistema de inecuaciones:
$v_3 < v_5$
$v_6 < v_4$
$v_1 \leq v_7$ y $v_3 \leq v_7$
$v_2 \leq v_8$ y $v_4 \leq v_8$
$v_1 \leq v_9$ y $v_7 < v_9$
$v_2 \leq v_{10}$ y $v_8 < v_{10}$

Una solución de este sistema de inecuaciones viene dado, por ejemplo, por:
$v_1 = v_2 = 0$ (ni $seta(a)$ ni $seta(b)$ aparecen en la cabeza de ninguna regla que no sea un hecho)
$v_3 = v_6 = 0$ (ni $rojo(a)$ ni $gris(b)$ aparecen en la cabeza de ninguna regla)
$v_4 = v_5 = 1$
$v_7 = v_8 = 2$
$v_9 = v_{10} = 3$

En resumen, la función de nivel l para P sería así: $l(seta(a)) = 0$, $l(seta(b)) = 0$,

$l(rojo(a)) = 0$, $l(rojo(b)) = 1$, $l(gris(a)) = 1$, $l(gris(b)) = 0$, $l(venenosa(a)) = 2$, $l(venenosa(b)) = 2$, $l(comestible(a)) = 3$, $l(comestible(b)) = 3$.

Por lo tanto, $\boldsymbol{P}$ es un programa localmente estratificado. Ahora bien, el programa $\boldsymbol{P}$ no es estratificado. Obsérvese que no puede ocurrir que $l(rojo(a)) = l(rojo(b))$ y $l(gris(a)) = l(gris(b))$.

Ahora vamos a ver una serie de ejemplos sencillos para terminar de entender bien estas definiciones.

Ejemplo 5.2. Sea $\boldsymbol{P}$ el siguiente programa:

```
p(a)← ¬q(a)
q(a)← ¬ p(a)
r(X)← p(X)
```

Si intentamos buscar una función de nivel, tendría que cumplir que $l(p(a)) > l(q(a))$ y que $l(q(a)) > l(p(a))$, y obviamente, no es posible. Por lo tanto, $\boldsymbol{P}$ es un programa no localmente estratificado.

Ejemplo 5.3. Sea $\boldsymbol{P}$ el siguiente programa:

```
p(a)←
q(a)← p(a)
r(a)← p(a),q(a)
```

Este programa es claramente localmente estratificado (de hecho, estratificado) tomando, por ejemplo, como función de nivel $l(p(a)) = 0$, $l(q(a)) = 1$ $l(r(a)) = 2$. Más aún, puesto que no aparece la negación en $\boldsymbol{P}$, podríamos simplemente tomar como función de nivel $l(p(a)) = l(q(a)) = l(r(a)) = 0$.

El propósito de este sencillo ejemplo es destacar que todo programa definido es localmente estratificado (de hecho, estratificado).

Ejemplo 5.4 (PAR). Sea $\boldsymbol{PAR}$ el siguiente programa.

```
par(a)←
par(s(X))← ¬ par(X)
```

En primer lugar, vamos a calcular $ground(PAR)$:

```
par(a)←
par(s(a))← ¬ par(a)
par(s(s(a)))← ¬ par(s(a))
par(s(s(s(a))))← ¬ par(s(s(a)))
   ⋮
```

Estudiemos si es localmente estratificado. Para ello, vamos a calcular una función de nivel que cumpla las condiciones necesarias. Sea la función $l : B_P \to \omega$ tal que $l(par(s^n(a))) = n$. Es fácil comprobar que dicha función de nivel hace al programa PAR un programa localmente estratificado y que, además, en este caso el uso del ordinal infinito ω en la función de nivel es necesario. En particular, P no es estratificado.

Ejemplo 5.5. Sea P el siguiente programa.

```
par(a)←
par(s(X))← ¬ par(X)
q(a)← par(X)
r(a)← ¬ q(a)
```

Vamos a comprobar que es localmente estratificado y que necesita una función de nivel con ordinal α infinito y, en este caso, con $\alpha \geq \omega+2$. En primer lugar, necesitamos $ground(P)$:

```
par(a)←
par(s(a))← ¬ par(a)
par(s(s(a)))← ¬ par(s(a))
   ⋮
q(a)← par(a)
q(a)← par(s(a))
q(a)← par(s(s(a)))
   ⋮
r(a)← ¬ q(a)
```

Es fácil comprobar que P necesita, como poco, una función de nivel $l : B_P \to \omega + 2$ dada por $l(par(s^n(a))) = n$, $l(q(a)) = \omega$, $l(r(a)) = \omega + 1$.

Los programas estratificados son particularmente relevantes desde el punto procedural (como veremos en la Sección 5.2). Sin embargo, desde el punto de vista de la semántica declarativa, resulta más relevante la noción de programa localmente estratificado. De hecho, en el trabajo (Przymusinski, 1988)[8] se introduce la noción de programa localmente estratificado y, a la par, se introduce una semántica declarativa para dichos programas que definimos a continuación: la semántica de los *modelos perfectos*.

Durante esta sección, y las dos restantes, trabajamos en la lógica clásica TWO y podemos identificar pues las interpretaciones I para un programa P con subconjuntos de su base de Herbrand B_P, $I \subseteq B_P$.

Definición 5.3 (Modelos perfectos). *Sea P un programa normal localmente estratificado y sea l su función de nivel asociada.*

1. *Sean M y N dos modelos distintos de P. Diremos que N es preferible a M si, para cualquier átomo básico A en $N - M$, existe un átomo básico B en $M - N$ tal que $l(A) > l(B)$.*
2. *Un modelo M de P se dirá perfecto si no hay modelos de P que sean preferibles a M.*

Ejemplo 5.6 (PIOLÍN 2). Volviendo al programa del ejemplo 3.21, vamos a ver que es localmente estratificado. Para ello, vamos a calcular la función de nivel l:
$l(pinguino(piolin)) = v_1$, $l(pinguino(bob)) = v_2$, $l(pajaro(piolin)) = v_3$,
$l(pajaro(bob)) = v_4$, $l(vuela(piolin)) = v_5$,$l(vuela(bob)) = v_6$

Como hicimos en el ejemplo 5.1, llegamos al siguiente sistema de inecuaciones:
$v_1 \leq v_3$.
$v_2 \leq v_4$.
$v_1 < v_5$ y $v_3 \leq v_5$
$v_2 < v_6$ y $v_4 \leq v_6$
$v_2 \leq v_2$.

Una de las soluciones sería la siguiente:
$l(pinguino(piolin)) = 0$, $l(pinguino(bob)) = 0$, $l(pajaro(piolin)) = 1$,
$l(pajaro(bob)) = 1$, $l(vuela(piolin)) = 2$, $l(vuela(bob)) = 2$

Como ya tenemos la función de nivel que cumple todas las condiciones, el programa es localmente estratificado. Podemos observar también que, por la definición 5.2,

el programa es estratificado.

Dados los dos modelos M y M' del ejemplo 3.21, M es preferible a M':
$M - M' = \{vuela(bob)\}, v_6 = 2$
$M - M' = \{pinguino(bob)\}, v_2 = 0$
De hecho, puede probarse que M es un modelo perfecto del programa PIOLÍN2.

Nótese que, en principio, la definición de modelo perfecto para un programa P depende de la función de nivel l que hayamos elegido para certificar que P es localmente estratificado. Sin embargo, en (Przymusinski, 1988)[8] se demuestra el siguiente resultado sobre la existencia de los modelos perfectos que, en particular, demuestra que la noción de modelo perfecto no depende de la función de nivel l elegida.

Teorema 5.1 (Przymusinski, 1988). *Sea P un programa normal localmente estratificado. Entonces P tiene un único modelo perfecto. Además, dicho modelo es independiente de la función de nivel con respecto a la cual el programa P es localmente estratificado.*

Por tanto, la semántica de los modelos perfectos consigue el objetivo fundamental de asociar a cada programa localmente estratificado un único modelo canónico, que puede considerarse pues el *significado natural* del programa lógico. Además, dicho modelo canónico coincide con el modelo esperado para programas definidos.

Lema 5.1. Sea P un programa normal localmente estratificado. Todo modelo perfecto de P es un modelo minimal de P.

Demostración. Sea M un modelo perfecto de P. Por reducción al absurdo, supongamos que existe $N \subset M$ tal que N es modelo de P. Puesto que M es perfecto, N no es preferible a M. En particular, debe existir algún átomo A en $N - M$, lo cual contradice el hecho de que $N \subseteq M$.

Corolario 5.1. Sea P un programa definido. Entonces, P tiene un único modelo perfecto, el cual coincide con su menor modelo de Herbrand, $T_P \uparrow \omega$.

Demostración. Es claro que todo programa definido está locamente estratificado y que podemos elegir una función de nivel l que asocie a todo átomo básico el valor 0 (la conectiva *not* no aparece en el programa). Es inmediato comprobar que el menor modelo de Herbrand de P satisface la definición de modelo perfecto y, por el lema anterior, este es el único modelo perfecto de P.

5.2 Programas estratificados. Interpretación procedural

En este apartado, vamos a estudiar una definición distinta pero equivalente de programa estratificado. En este caso, vamos a verlo desde un enfoque más procedimental: en los programas estratificados podemos encontrar un orden de evaluación para las reglas del programa de tal manera que el valor de los literales negativos puede ser predeterminado.

Definición 5.4 (Grafo de dependencia). *Sea P un programa lógico normal. Su grafo de dependencia, $dep(P)$, consta de un conjunto V de nodos y un conjunto E de arcos definidos como sigue.*

- *V es el conjunto de todos los símbolos de predicado del lenguaje del programa.*
- *E está formado por arcos $p \to q$ donde p, q son símbolos de predicado de P. Un arco $p \to q$ aparecerá en E por cada cláusula C de P tal que el cuerpo de C contiene algún literal L_i con símbolo de predicado q y la cabeza de C es un átomo con símbolo de predicado p. Si el literal L_i es un literal negativo, el arco se marcará con la etiqueta $*$, esto es, $(p \to^* q)$.*

Observación 5.1. Intuitivamente, dado un programa P, el predicado q será evaluado antes que p si en el grafo de dependencia hay un camino de p a q

$$p = p_0 \to p_1 \to \ldots \to p_{n-1} \to p_n = q$$

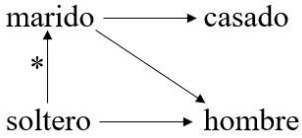
tal que para algún i, $0 \leq i \leq n$, el arco $p_i \to p_{i+1}$ está marcado con $*$.

En el siguiente ejemplo, vamos a ver un caso sencillo de un grafo de dependencia.

Ejemplo 5.7. Dado el siguiente programa:

```
hombre(miguel)←
marido(X)← hombre(X),casado(X)
soltero(X)← hombre(X),¬ marido(X)
```

Su grafo de dependencia es el siguiente:



En este ejemplo, *marido* y *casado* deben ser evaluados antes que *soltero* ya que encontramos el siguiente camino en el grafo $soltero \rightarrow^* marido \rightarrow casado$.

Definición 5.5 (Estratificación). *Sea P un programa normal y sea $pred(P)$ el conjunto de todos los símbolos de predicado que hay en el lenguaje de P. Una estratificación de P es una partición de $pred(P)$ en n conjuntos disjuntos y no vacíos, $\Sigma = \{S_i : 1 \leq i \leq n\}$, tal que:*

- *Si $p \in S_i$, $q \in S_j$ y $p \rightarrow q$ está en $dep(P)$, entonces $i \geq j$.*
- *Si $p \in S_i$, $q \in S_j$ y $p \rightarrow^* q$ está en $dep(P)$, entonces $i > j$.*

Un programa P es estratificado si tiene alguna estratificación Σ. Los conjuntos S_i, $1 \leq i \leq n$, de la estratificación se denominan estratos.

Una estratificación Σ determina un orden de evaluación para los predicados del programa. Siguiendo dicho orden, es posible calcular un modelo del programa mediante una serie de modelos mínimos iterados.

Definición 5.6 (Modelos mínimos iterados). *Sean P un programa lógico normal y Σ una estratificación de P, $\Sigma = \{S_1, ..., S_k\}$, con $k \geq 1$. Definimos P_{S_i} como el conjunto de las cláusulas de P tales que su cabeza está formada por un átomo cuyo símbolo de predicado está en S_i. Definimos también el conjunto $B^*_{P_{S_i}} = \bigcup_{j \leq i} \{p(t) \in B_P : p \in S_j\}$. Por último, definimos los modelos mínimos iterados $M_i \subseteq B_P$, con $1 \leq i \leq k$, de la siguiente manera:*

1. *M_1 es el menor modelo de Herbrand de P_{S_1}.*
2. *Si $i > 1$, M_i es el menor subconjunto M de B_P tal que:*
   - *M es un modelo de Herbrand de P_{S_i}.*
   - *$M \cap B^*_{P_{S_{i-1}}} = M_{i-1} \cap B^*_{P_{S_{i-1}}}$.*

El modelo mínimo iterado de P es M_k y escribiremos $M_{P,\Sigma}$ para denotarlo.

Ejemplo 5.8. Volviendo con el programa del ejemplo 5.7 y teniendo en cuenta su grafo de dependencia, vamos a buscar una estratificación de P.

Teniendo en cuenta la Definición 5.5, al haber una arista marcada con $*$ entre *soltero* y *marido*, *soltero* tiene que estar en un conjunto S_i tal que $i > j$ donde j es el subíndice del conjunto S_j con $marido \in S_j$. Por otro lado, podemos observar en el grafo que no hay ninguna arista entre *casado* y *hombre*, pero sí están las siguientes aristas: $soltero \to hombre$, $marido \to hombre$ y $marido \to casado$. Por lo tanto, una estratificación podría quedar así:
$S_1 = \{hombre, casado\}$, $S_2 = \{marido\}$, $S_3 = \{soltero\}$.

Por la definición anterior, tenemos que $P_{S_1} = \{$`hombre(miguel)` $\leftarrow\}$ y M_1 es el menor modelo de P_{S_1}, $M_1 = \{hombre(miguel)\} \subseteq B_P$.
Por otro lado, $P_2 = \{$`marido(X)` $\leftarrow$ `hombre(X),casado(X)`$\}$ y un modelo de P_{S_2} sería $M_2 = \{hombre(miguel)\}$. Puesto que $B^*_{P_{S_1}} = \{hombre(miguel), casado(miguel)\}$ y $M_2 = M_1$, es sencillo comprobar que $M_2 \cap B^*_{P_{S_1}} = M_1 \cap B^*_{P_{S_1}}$.
Por lo tanto, M_2 es el menor modelo que cumple estas propiedades.
Por último, $P_{S_3} = \{$`soltero(X)` $\leftarrow$ `hombre(x),`$\neg$`marido(X)`$\}$ y un modelo para P_{S_3} sería $M_3 = \{hombre(miguel), soltero(miguel)\}$. Puesto que se cumple que $B^*_{P_{S_2}} = \{hombre(miguel), casado(miguel), marido(miguel)\}$, podemos comprobar que $M_3 \cap B^*_{P_{S_2}} = M_2 \cap B^*_{P_{S_2}}$. Por tanto, M_3 es el mínimo modelo de P_{S_3} y $M_{P,\Sigma} = M_3$.

Observación 5.2. Es importante observar que la estratificación no es única. En el ejemplo anterior, podríamos haber dado una estratificación diferente, por ejemplo: $S_1 = \{hombre, casado, marido\}$ y $S_2 = \{soltero\}$. Sin embargo, se cumple el siguiente teorema.

Teorema 5.2. *Dados P un programa estratificado y Σ, Σ' dos estratificaciones de P, se tiene que $M_{P,\Sigma} = M_{P,\Sigma'}$. Esto es, el menor modelo iterado de P no depende de la estratificación elegida. Además, dicho menor modelo iterado de P coincide con el modelo perfecto de P.*

Demostración. Una prueba de este hecho puede encontrarse en *Apt, K., Blair, H., Walker, A.: Towards a Theory of Declarative Knowledge, pp. 89-148* en *Foundations of Deductive Databases and Logic Programming, 1988 Elsevier.*

Observación 5.3. Como dijimos al inicio de esta sección, las definiciones que hemos dado de *programa estratificado* son equivalentes. Vamos a comprobarlo de una manera sencilla.

Dado un programa estratificado P siguiendo la definición original, la cuestión es cómo podemos construir los estratos S_i a los que hace referencia la nueva definición.

Partiríamos de una función de nivel l que cumpliera todas las condiciones necesarias para que P fuera un programa estratificado. Simplemente construimos la estratificación Σ empezando por el conjunto S_1 donde estarían los símbolos de predicado del lenguaje de P cuyo valor en la función de nivel fuera el mínimo. Para continuar, el conjunto S_2 estaría formado por los símbolos de predicado cuyo valor en la función de nivel fuera el consecutivo según l, es decir, el primer valor mayor que el considerado en el caso anterior. Y así sucesivamente tendríamos los conjuntos S_1,..., S_n que forman la estratificación Σ.

Por otro lado, dado un programa estratificado P siguiendo la segunda definición con estratos S_1,...,S_n , ¿cómo podemos construir la función de nivel a la que hace referencia la definición original? Para definirla, daríamos el valor 0 a aquellos átomos de B_P con símbolo de predicado en el conjunto S_1, el valor 1 a aquellos que estuvieran en S_2, y así sucesivamente.

Ejemplo 5.9. Consideramos una vez más el programa PIOLÍN2 del ejemplo 3.21. En el ejemplo 5.6 vimos que era un programa estratificado según la definición original. Vamos a construir una estratificación de P para probar así que es estratificado según la nueva definición. Puesto que $l(pinguino(piolin)) = 0$ y $l(pinguino(bob)) = 0$, $S_1 = \{pinguino\}$. Por otro lado, $S_2 = \{pajaro\}$ ya que $l(pajaro(piolin)) = 1$ y $l(pajaro(bob)) = 1$. Finalmente, $S_3 = \{vuela\}$ ya que $l(vuela(piolin)) = 2$ y $l(vuela(bob)) = 2$. Tomemos $\Sigma = \{S_1, S_2, S_3\}$.

De hecho, ahora podemos justificar que el modelo

$$M = \{pinguino(piolin), pajaro(piolin), pajaro(bob), vuela(bob)\}$$

considerado en el ejemplo 5.6 es el modelo perfecto del programa PIOLÍN2, pues coincide con su menor modelo iterado $M_{PIOLIN2,\Sigma}$.

Ejemplo 5.10. Volviendo al programa del ejemplo 5.7 y tomando la estratificación que hemos buscado en el ejemplo 5.8, vamos a calcular una función de nivel l que cumpla las condiciones para que sea programa estratificado según la definición original. Damos el valor 0 a los átomos con predicados en S_1, el valor 1 a los átomos con predicados en S_2 y el valor 2 a los átomos con predicados en S_3. La función de nivel queda de la siguiente manera: $l(hombre(miguel)) = 0$, $l(casado(miguel)) = 0$, $l(marido(miguel)) = 1$, $l(soltero(miguel)) = 2$.

5.3 Programas localmente estratificados. Existencia de modelo único

El objetivo de esta sección es presentar las ideas más relevantes de la demostración del teorema principal de este capítulo: "todo programa localmente estratificado tiene un único modelo estable".

En primer lugar, usaremos el concepto de completación de punto fijo de un programa P, que no es más que una transformación del programa basada en la noción de despliegue que explicaremos formalmente a continuación.

Definición 5.7 (Semi-interpretación). *Una semi-interpretación es un conjunto de cláusulas de la forma* $A \leftarrow \neg B_1, \ldots, \neg B_m$, *donde* A *y* B_i *son átomos básicos para todo* $i = 1, \ldots, m$.

Definición 5.8 (El operador T'_P). *Sean* P *un programa lógico normal y* Q *una semi-interpretación. Definimos un nuevo operador* T'_P[1] *que actúa sobre semi-interpretaciones como sigue:*
$T'_P(Q)$ *es la semi-interpretación formada por el conjunto de todas las cláusulas*

$$A \leftarrow bod\, y_1, \ldots, bod\, y_n, \neg B_1, \ldots, \neg B_m$$

para las que existen una cláusula $A \leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m$ *en* $ground(P)$ *y cláusulas* $A_i \leftarrow bod\, y_i$ *en* Q *donde* $i = 1, \ldots, n$.

Nótese que se permite tomar $n = 0$ *o* $m = 0$ *en la presente definición.*

Podemos dotar al conjunto de las semi-interpretaciones de una estructura de orden parcial considerando que $Q \leq R$ si, y solo si, $Q \subseteq R$ vistos como conjuntos de cláusulas. El conjunto de las semi-interpretaciones es un orden parcial completo con respecto al orden de la inclusión de conjuntos.

La siguiente propiedad de operador T'_P será crucial.

Proposición 5.1. Sea P un programa lógico normal. El operador T'_P es continuo en el espacio de todas las semi-interpretaciones.

[1] Siguiendo la literatura del tema, hemos usado el mismo símbolo T'_P para denotar tanto el operador entre semi-interpretaciones aquí definido como la parte positiva del operador de Fitting que estudiamos en el Capítulo 4. Puesto que ambos opradores actúan sobre espacios distintos, creemos que no producirá confusión al lector.

Demostración. En primer lugar, vamos a probar que T'_P es monótono. Sean Q y R dos semi-interpretaciones tal que $Q \subseteq R$. Sea $A \leftarrow bod\,y$ en $T'_P(Q)$. Vamos a probar que también está en $T'_P(R)$. Hay que distinguir dos casos:

1. Si $A \leftarrow bod\,y$ no resulta de ningún despliegue, entonces está contenida en P y, por lo tanto, pertenece también a $T'_P(R)$.
2. Si $A \leftarrow bod\,y$ viene del despliegue de alguna cláusula $A \leftarrow bod\,y_0$ en P con algunas cláusulas $B_i \leftarrow bod\,y_i$ en Q, entonces $B_i \leftarrow bod\,y_i$ pertenece a R para todo i. Como tenemos que existe $A \leftarrow bod\,y_0$ en P, $A \leftarrow bod\,y$ pertenece también a $T'_P(R)$.

Por último, tenemos que probar que T'_P conserva supremos. Para ello, vamos a considerar una familia dirigida de semi-interpretaciones $\mathcal{Q}$ tal que $\mathcal{Q} = \{Q_\lambda : \lambda \in \Lambda\}$. Sea $R = sup\ \mathcal{Q}$. Entonces $T'_P(\mathcal{Q})$ es un conjunto dirigido ya que T'_P es monótona y estamos considerando el orden por inclusión de conjuntos y por tanto tiene supremo. Nos quedaría probar que $T'_P(R) = sup\ T'_P(\mathcal{Q})$.

La primera contención, $sup\ T'_P(\mathcal{Q}) \subseteq T'_P(R)$, es sencilla y se cumple para cualquier función monótona f. Omitimos los detalles.

Ahora, vamos a demostrar la segunda contención. Suponemos que $A \leftarrow bod\,y$ pertenece a $T'_P(R)$. Si esta cláusula no viene de ningún despliegue, está contenida en P y por tanto, en $T'_P(\mathcal{Q})$. En caso contrario, proviene del despliegue de alguna cláusula $A \leftarrow bod\,y_0$ en P con alguna $B_i \leftarrow bod\,y_i$ en R. Entonces existe λ tal que todos los $B_i \leftarrow bod\,y_i$ están contenidos en Q_λ. Por lo tanto, $A \leftarrow bod\,y$ está contenida en $T'_P(Q_\lambda) \subseteq T'_P(\mathcal{Q})$, como queríamos demostrar. ∎

Puesto que T'_P es un operador continuo actuando sobre un orden parcial completo, podemos aplicar el teorema de punto fijo de Kleene. Esto justifica la siguiente definición.

Definición 5.9 (Completación de punto fijo). *Sea P un programa lógico normal. Definimos la completación de punto fijo de P, $fix(P)$, de la siguiente manera:*

$$fix(P) = T'_P \uparrow \omega.$$

Ejemplo 5.11 (PIOLÍN 2). Sea P el programa lógico del ejemplo 3.21. Vamos a calcular su completación de punto fijo $fix(P)$. Aplicando la definición anterior, se tiene que:

$T'_P \uparrow 0 = \emptyset$
$T'_P \uparrow 1 = \{pinguino(piolin) \leftarrow, pajaro(bob) \leftarrow\}$

$T'_P \uparrow 2 = T'_P \uparrow 1 \cup \{pajaro(piolin), vuela(bob) \leftarrow \neg pinguino(bob)\}$
$T'_P \uparrow 3 = T'_P \uparrow 2 \cup \{vuela(piolin) \leftarrow \neg pinguino(piolin)\}$
Finalmente, $fix(P) = T'_P \uparrow 3$.

Luego, la completación de punto fijo del programa PIOLÍN2 es el conjunto de clásulas

```
pinguino(piolin)←
pajaro(bob)←
pajaro(piolin)←
vuela(bob)← ¬ pinguino(bob)
vuela(piolin)← ¬ pinguino(piolin)
```

La importancia de la completación de punto fijo de un programa lógico P radica en el hecho de que los modelos estables de P son, exactamente, los modelos justificados de $fix(P)$. Más aún, se tiene que (recuérdese que GL_P denota el operador de Gelfond-Lifschitz y que T_P denota el operador de consecuencia inmediata):

Teorema 5.3. *Sea P un programa lógico normal y sea I una interpretación de Herbrand para P. Entonces, $GL_P(I) = T_{fix(P)}(I)$.*

Demostración. En primer lugar, vamos a demostrar $GL_P(I) \subseteq T_{fix(P)}(I)$. Recordando que $GL_P(I) = T_{P_I} \uparrow \omega$, vamos a probar esta contención por inducción en las potencias de T_{P_I}.

- Si $k = 0$, $T_{P_I} \uparrow 0 = \emptyset$.
- Supongamos ahora que para todo $A \in T_{P_I} \uparrow k$, existe una cláusula de la forma $A \leftarrow body$ en $fix(P)$ tal que $I(body) = \mathbf{t}$. Sea $A \in T_{P_I} \uparrow k+1$). Entonces existe una cláusula $A \leftarrow A_1, \ldots, A_k$ en P_I tal que $A_1, \ldots, A_k \in T_{P_I} \uparrow k$ y, por definición del reducto P_I, existe una cláusula $A \leftarrow body_1, \ldots, body_k, \neg B_1, \ldots, \neg B_m$ en $ground(P)$ con $B_1, \ldots, B_m \notin I$. Por hipótesis de inducción, existe una cláusula $A_i \leftarrow body_i$ en $fix(P)$ con $I(body_i) = \mathbf{t}$ para cada $i = 1, \ldots, k$ y entonces $A_i \in T_{fix(P)}(I)$. Por definición de T'_P, $A \leftarrow body_1, \ldots, body_k, \neg B_1, \ldots, \neg B_m$ pertenece a $fix(P)$. Finalmente, como $I(body_i) = \mathbf{t}$ y $B_1, \ldots, B_m \notin I$, $A \in T_{fix(P)}(I)$, como queríamos demostrar.

A continuación, vamos a demostrar la otra contención, $T_{fix(P)}(I) \subseteq GL_P(I)$. Sea $A \in T_{fix(P)}(I)$. Vamos a probar por inducción en k que $T_{T'_P \uparrow k}(I) \subseteq GL_P(I)$ para todo $k \in \mathbb{N}$.

- Si $k = 0$, $T_{T'_P \uparrow 0}(I) = \emptyset$.
- Suponemos que $T_{T'_P \uparrow k}(I) \subseteq GL_P(I)$. Tenemos que demostrar $T_{T'_P \uparrow k+1}(I) \subseteq GL_P(I)$. Sea $A \in T_{T'_P \uparrow (k+1)}(I)\ T_{T'_P \uparrow k}(I)$. Entonces, existe una cláusula $A \leftarrow bod\,y_1, \ldots, bod\,y_n, \neg B_1, \ldots, \neg B_m$ en $T'_P \uparrow (k+1)(I)$ cuyo cuerpo es verdad en I. Por consiguiente, $B_1, \ldots, B_m \notin I$, y existe una cláusula $A_i \leftarrow bod\,y_i$ en $T'_P \uparrow k$ tal que $I(bod\,y_i) = \mathbf{t}$ para cada $i = 1, \ldots, n$. Por tanto, $A_i \in T_{T'_P \uparrow k}(I) \subseteq GL_P(I)$. Como $B_1, \ldots, B_m \notin I$ y por definición de T'_P, existe una cláusula $A \leftarrow A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m$ en $ground(P)$ tal que $A \leftarrow A_1, \ldots, A_n$ pertenece a P_I. Por último, como $A_i \in GL_P(I)$ para todo $i = 1, \ldots, n$, $A \in GL_P(I)$, y ya se tiene que $T_{fix(P)}(I) = T_{T'_P \uparrow (k+1)}(I) \subseteq GL_P(I)$, como queríamos demostrar.

Esto concluye la prueba del teorema. ∎

Como una consecuencia inmediata obtenemos:

Corolario 5.2. Sea P un programa lógico normal. Los modelos estables de P son, exactamente, los modelos justificados de $fix(P)$.

Demostración. Se sigue del Teorema anterior porque, como consecuencia de la Proposición 3.3, los puntos fijos del operador $T_{fix(P)}$ son los modelos justificados del programa $fix(P)$. ∎

Estamos preparados ya para enunciar el principal teorema de este capítulo. Con el trabajo desarrollado previamente, su demostración se reduce a poner juntas las piezas necesarias.

Teorema 5.4. *Sea P un programa lógico normal localmente estratificado. Entonces, el operador de Gelfond-Lifschitz GL_P tiene un único punto fijo y, por tanto, P tiene un único modelo estable.*

Demostración. Obsérvese que si P es localmente estratificado con respecto a la función de nivel l, entonces $fix(P)$ es localmente jerárquico (recuérdese la Definición 3.11) respecto a l. Puesto que $fix(P)$ es localmente jerárquico, por el Teorema 3.3, se tiene que el operador $T_{fix(P)}$ tiene un único punto fijo y el programa $fix(P)$ tiene un único modelo justificado. El resultado ahora se sigue del Teorema 5.3 anterior. ∎

Aunque no incluimos su prueba (dicha prueba conllevaría desarrollar una nueva semántica para los programas lógicos, los *modelos bien-fundados*), en (van Gelder, Ross, Schlipf 1988)[6] se demuestra (una versión más general) del siguiente resultado.

Teorema 5.5. *Sea P un programa lógico normal localmente estratificado. Si P tiene un modelo perfecto, entonces ese modelo es su único modelo estable.*

Luego, junto con el Teorema 5.1, obtenemos

Corolario 5.3. Sea P un programa lógico normal localmente estratificado. Entonces, P tiene un único modelo estable y un único modelo perfecto y ambos modelos coinciden.

Puesto que para los programas estratificados es posible calcular su modelo perfecto mediante su menor modelo iterado, obtenemos también

Corolario 5.4. Sea P un programa lógico normal estratificado. Entonces, P tiene un único modelo estable y coincide con su menor modelo iterado.

6 | Conclusiones

En primer lugar, cabe destacar que la semántica de los programas lógicos definidos no presenta ninguna dificultad. Está claramente aceptada para ellos la semántica de punto fijo basada en el operador de consecuencia inmediata T_P.

Por otro lado, a lo largo del trabajo hemos comprobado que establecer el significado de la negación en Programación Lógica es un tema mucho más delicado y, de hecho, no hay una única semántica privilegiada que pueda considerarse estándar para los programas lógicos con negación. En el campo de la Programación Lógica se ha dedicado un gran esfuerzo en intentar generalizar los teoremas de punto fijo para el operador de consecuencia inmediata a otros operadores asociados a programas con negación, así como al estudio de las relaciones entre las distintas semánticas propuestas. Pero ninguna de ellas podría destacarse como la semántica *definitiva* de los programas normales.

Sin embargo, a pesar de que no haya una semántica que cumpla todos los propósitos para estos programas, merece la pena destacar que sí hay una clase de programas lógicos con negación privilegiados: los programas localmente estratificados. A cada programa de este tipo se les puede asociar un único modelo canónico que refleje su significado natural.

Por último, la realización de este trabajo me ha permitido introducirme en ciertas técnicas formales con las que no había tenido ningún contacto durante los estudios del Grado. Principalmente, el uso de lógicas no clásicas multivaluadas y el uso de ordinales y construcciones transfinitas basadas en ellos.

Bibliografía

[1] M. Gelfond and V. Lifschitz. (1988) The stable model semantics for logic programming In: Proceedings of the Fifth International Conference on Logic Programming (ICLP), 1070-1080.

[2] T. Eiter, G. Ianni and T. Krennwallner. (2009) Answer Set Programming: A primer.

[3] P. Hitzler and A. Seda. (2011) Mathematical aspects of logic programming semantics. Chapman y Hall/CRC.

[4] M. Fitting. (1985) Fixpoint Semantics for Logic Programming A Survey.

[5] P. Rondogiannis and William W. Wadge. (2005) Minimum Model Semantics for Logic Programs With Negation-As-Failure.

[6] A. Van Gelder, K. Ross and J. S. Schlipf. (1988) Unfounded sets and well-founded semantics for general logic programs, Proc. Seventh Symp. on Principles of Database Systems, 221-230.

[7] M. H. van Emden and R. A. Kowalski. (1976) The semantics of predicate logic as a programming language. Journal of the ACM, 23(4):733-742.

[8] T. C. Przymusinski. (1988) On the declarative semantics of deductive databases and logic programs. In Minker, J., editor, Foundations of Deductive Databases and Logic Programming, 193-216. Morgan Kaufmann Publishers, Los Altos, CA.

[9] R. A. Kowalski. (1974) Predicate logic as a programming language. In Rosenfeld, J.L., editor, Proceedings IFIP'74, Stockholm, Sweden, August, 1974, 569-574. North-Holland, Amsterdam.

[10] J. A. Robinson. (1965) A machine-oriented logic based on the resolution principle. Journal of the ACM, 12(1):23-41.