Capítulo 1

Marco teórico

En este capítulo se presentan las bases conceptuales de la programación lógica además de algunas propiedades de conjuntos parcialmente ordenados y de la programación de orden parcial.

1.1 Fundamentos de la Programación Lógica.

Los programas lógicos son un subconjunto de la lógica de primer orden. Para definir los programas lógicos, se dará un breve panorama sobre la sintaxis y la semántica de la lógica de primer orden. Además se presentarán algunos conceptos básicos, los cuales son necesarios para los fundamentos teóricos de la programación lógica.

La lógica de primer orden tiene dos conceptos fundamentales: la sintaxis y la semántica. El aspecto sintáctico son las fórmulas bien formadas admitidas por la gramática de un lenguaje formal. La semántica se refiere al significado relacionado a las fórmulas bien formadas.

La teoría de primer orden consiste de un alfabeto, un lenguaje de primer orden, un conjunto de axiomas y un conjunto de reglas de inferencia. El lenguaje de primer orden consiste de fórmulas bien formadas. Los axiomas son un subconjunto designado de fórmulas bien formadas. Los axiomas y las reglas de inferencia son usadas para derivar los teoremas de la teoría.

La sintaxis de la lógica de primer orden está basada en un alfabeto, definido de la siguiente manera:

Definición 1.1.1 ([5]) *Un alfabeto consiste de siete clases de símbolos:*

1. *Variables, son denotadas por las letras mayúsculas U, V, W, X, Y y Z.*
2. *Símbolos de funciones, denotadas por una secuencia finita de las letras minúsculas f, g, h.*

3. *Símbolos de predicados, denotados por una secuencia finita de letras minúsculas como por ejemplo p, q, r, path, top.*

4. *Constantes proposicionales, Verdadero y Falso.*

5. *Conectivos lógicos:* $\sim$*(negación),* $\wedge$ *(conjunción),* $\vee$ *(disyunción),* $\rightarrow$ *(implicación) y* $\leftrightarrow$ *(equivalencia).*

6. *Cuantificadores:* $\exists$*(existencia o cuantificador existencial) y* $\forall$ *(para todo o cuantificador universal).*

7. *Símbolos de puntuación, "(", ")" y ",".*

La jerarquía de precedencia es la siguiente:

$$\sim, \forall, \exists$$
$$\vee$$
$$\wedge$$
$$\rightarrow, \leftrightarrow$$

Las clases de símbolos de la 4 a la 7 en la definición anterior son las mismas para cada lenguaje mientras que, los símbolos de funciones y los símbolos de predicados únicamente determinan el alfabeto de cada lenguaje. Hay un número natural asociado con cada símbolo de función y símbolo de predicado, llamado aridad[1] del símbolo. Si el símbolo tiene aridad n, entonces este es llamado un símbolo de aridad n, y si es de aridad 0 es llamado símbolo constante. Las constantes son denotadas por una secuencia finita de letras minúsculas a, b, c, d. Un símbolo de predicado de aridad 0 es llamado símbolo proposicional.

Otras definiciones importantes en la programación lógica son:

Definición 1.1.2 ([5]) *Un término es definido inductivamente por:*

- *Una variable es un término.*
- *Una constante es un término.*
- *Sea f un símbolo de función con aridad n y sean* $t_1,\ldots,t_n$ *términos, entonces* $f(t_1,..,t_n)$ *es un término.*

Definición 1.1.3 ([5]) *Una fórmula bien formada es definida inductivamente por :*

[1]Aridad : Número de argumentos de un símbolo de función o símbolo de predicado.

- *Si p es un símbolo de predicado con aridad n y t_1,..,t_n términos, entonces p(t_1,...,t_n) es una fórmula (llamada fórmula atómica, o simplemente átomo).*
- *Verdadero y falso son fórmulas.*
- *Si F y G son fórmulas, entonces ($\sim$F), (F $\wedge$ G), (F $\vee$ G), (F $\rightarrow$ G) y (F $\leftrightarrow$G) son fórmulas.*
- *Si F es una fórmula y x es una variable, entonces ($\forall$x F) y ($\exists$x F) son fórmulas.*

A menudo será conveniente escribir (F $\rightarrow$G) como (G $\leftarrow$ F).

Un ejemplo de una fórmula bien formada es: $\forall$x $\exists$y (p(x,y) $\rightarrow$q(x))

Definición 1.1.4 ([5]) *El lenguaje de primer orden dado por un alfabeto, consiste de un conjunto de todas las fórmulas construídas de los símbolos del alfabeto.*

Definición 1.1.5 ([5]) *Una literal es un átomo (A) o la negación de un átomo ($\sim$A). Una literal positiva es un átomo. Una literal negativa es la negación de un átomo.*

Definición 1.1.6 ([5]) *Un término ground es un término que no contiene variables. Similarmente, un átomo ground es un átomo que no contiene variables.*

Definición 1.1.7 ([5]) *Una fórmula cerrada es una fórmula que no tiene ocurrencias libres de alguna variable.*

La semántica de una teoría de primer orden provee el significado de la teoría basada en alguna interpretación. Las interpretaciones proveen un significado específico a los símbolos del lenguaje y son usados para proveer significado al conjunto de fórmulas bien formadas.

Algunos conceptos importantes en la teoría de semánticas son pre-interpretación, interpretación y modelo los cuales se definen a continuación.

Definición 1.1.8 ([5]) *Una pre-interpretación de un lenguaje L de primer orden consiste en lo siguiente:*

- *Un conjunto D no-vacío, llamado dominio de la pre-interpretación.*
- *Para cada constante en L, se le asigna un elemento en D.*
- *Para cada símbolo de función con aridad n en L, se le asigna un mapeo de D^n a D.*

Definición 1.1.9 ([5]) *Una interpretación I de un lenguaje de primer orden L, consiste de una pre-interpretación J con dominio D en L junto con lo siguiente: para cada símbolo de predicado de ariadad n en L, se le asocia un mapeo de D^n en {Verdadero, Falso} (o equivalente, a una relación sobre D^n.)*

Definición 1.1.10 ([5]) *Sea I una interpretación para un lenguaje de primer orden L y W una fórmula en L.*

Se dice que W es satisfactible en I si $\exists(W)$ *es verdadero con respecto a I.*

Se dice que W es válida en I si $\forall(W)$ *es verdadero con respecto a I.*

Se dice que W es insatisfactible en I si $\exists$ *(W) es falso con respecto a I.*

Se dice que W es no válida en I si $\forall(W)$ *es falso con respecto a I.*

Definición 1.1.11 ([5]) *Sea I una interpretación de un lenguaje de primer orden L y F una fórmula cerrada de L. Entonces I es un modelo para F, si F evalúa verdadero en I.*

En otras palabras, *un modelo M* para una teoría T (conjunto de fórmulas) es una interpretación I de T tal que, toda formula $f \in T$ evalúa cierto en I.

Definición 1.1.12 ([5]) *Sea T una teoría y* α *una fórmula cerrada, entonces definimos que* α *es una consecuencia lógica de T, si para toda interpretación I tal que I es modelo de T, entonces I también es modelo de* α. *Se representa como:* $T \models \alpha$.

Ejemplo 1.1 *Considere los siguientes predicados:*

P(x) : donde x es hombre.

Q(x) : donde x es mortal.

a : Sócrates.

Sea T el siguiente conjunto de fórmulas:

$P(a)$

$\forall\, x(P(x) \rightarrow Q(x))$

En este modelo es cierto que Sócrates es mortal, por lo tanto $T \models Q(a)$.

La sustitución de variables por términos en una fórmula, juega un papel importante en la teoría de programas lógicos.

Definición 1.1.13 ([5]) *Una sustitución* θ *es un conjunto finito de la forma* $\{v_1 \leftarrow t_1, ..., v_n \leftarrow t_n\}$, *donde cada* v_i *es una variable, cada* t_i *es un término distinto de* v_i *y las variables* $v_1,...,v_n$ *son distintas, donde* θ *es llamada sustitución ground si las* t_i *son términos ground.*

El proceso de encontrar una sustitución es llamado unificación para los átomos. La unificación está basada en la sustitución, además de tener una gran importancia en la teoría de la programación lógica. Algunos conceptos importantes de la unificación son:

Definición 1.1.14 ([5]) *Dos átomos (o expresiones) A y B son unificables si existe una sustitución θ tal que $A\theta = B\theta$. La sustitución θ es llamada unificador para A y para B. Este es llamado un unificador más general (mgu) si para cada unificador η para A y para B existe una sustitución γ tales que $\eta = \theta\gamma$.*

Definición 1.1.15 *Match es un caso especial de la unificación, donde uno de los átomos es ground (no contiene variable) y el otro no.*

Un ejemplo de *match* es el siguiente:

Ejemplo 1.2 *Consideremos los siguientes átomos P(a, b) y P(x, y), es claro que el primer átomo es ground y el otro no, entonces para que unifiquen se tiene el unificador siguiente:*

$$\theta \leftarrow \{x \leftarrow a, y \leftarrow b\}$$

1.2 Conjuntos parcialmente ordenados.

La teoría de relaciones es fundamental en la comprensión de la programación de orden parcial presentada mas adelante. En esta sección se presentarán los conceptos fundamentales de la teoría de relaciones y retículos[2].

Frecuentemente queremos comparar o contrastar varios elementos de un conjunto, quizá para acomodarlos en un orden apropiado para agrupar aquéllos con propiedades similares. La estructura matemática para describir este tipo de organización de conjuntos es la teoría de relaciones.

Para ilustrar lo anterior, supongamos que hay una manera de comparar los elementos de un conjunto, de modo que siempre que se den dos elementos del conjunto sabemos como compararlos, veamos el siguiente ejemplo.

Ejemplo 1.3

1. *Estamos acostumbrados a comparar números reales. Por ejemplo: 3 es menor que 5, -1 es menor que 4 y -1 es mayor que -3. Comparamos dos números diciendo cual es menor y cuál es mayor.*
2. *Si el conjunto S tiene índices en P o N de manera que elementos distintos tienen índices distintos, entonces podemos comparar dos elementos de S observando cuál de ellos tiene menor índice.*

Si los elementos del conjunto se pueden comparar como en los incisos 1) y 2), se dice que el conjunto está *ordenado*, y a la especificación de cómo comparar un elemento de otro se le llama *relación de orden* en el conjunto. Para decir algo útil sobre conjuntos ordenados, debemos dar una definición más precisa. Primero hay que notar que, en muchos conjuntos que surgen naturalmente sabemos como comparar algunos elementos con otros, pero también hay parejas que no son comparables veáse el siguiente ejemplo.

[2]Del inglés lattices.

Ejemplo 1.4 *1. Podemos comparar dos números en {1,2,3,...,73} en caso de que uno sea factor de otro. Entonces podemos comparar 6 y 72 y también 3 y 6 sin embargo, no podemos comparar 6 y 8 ya que ni 8 es factor de 6 ni 6 es factor de 8.*

2. Podemos comparar dos subconjuntos de un conjunto S, si uno está contenido en otro. Si S tiene más de un elemento, entonces tiene subconjuntos incomparables. Por ejemplo, si s_1 y s_2 están en S y $s_1 \neq s_2$, entonces los conjuntos $\{s_1\}$ y $\{s_2\}$ no son comparables.

Los conjuntos donde las relaciones de comparación permite la posibilidad de que dos elementos no sean comparables como los del ejemplo 1.4, se llaman *conjuntos parcialmente ordenados*, y forman una clase muy importante que definiremos ahora.

Definición 1.2.1 ([15]) *Supongamos que nos dan una relación en un conjunto S de manera que para cada par $\langle x, y\rangle$ en $S \times S$ sabemos cuando x está relacionada con y. Escribimos $x \preceq y$ si y sólo si x está relacionada con y. La relación se llama un orden parcial en S si satisface lo siguiente:*

(R) $s \preceq s$ para toda s en S;

(AS) $s \preceq t$ y $t \preceq s$ implica $s = t$;

(T) $s \preceq t$ y $t \preceq u$ implica $s \preceq u$.

Las condiciones (R), (AS) y (T) son las propiedades *reflexiva, antisimétrica* y *transitiva* respectivamente, y se dice que $\preceq$ es reflexiva, antisimétrica o transitiva si las satisface. Si $\preceq$ es un orden parcial en S, se dice que $(S, \preceq)$ es un conjunto parcialmente ordenado, o para abreviar, un *copo*. Se utilizará la notación " $\preceq$ "como símbolo para un orden parcial en general.

En el ejemplo 1.4 las relaciones que se presentan se sobreentiende que son "*es un factor de*" y "*es subconjunto de*". También podríamos haber considerado las relaciones "*es un múltiplo de*" y "*contiene*", ya que estas relaciones dan la misma información comparativa que las elegidas.

Definición 1.2.2 ([15]) *Dado un orden parcial $\preceq$ en un conjunto S, podemos definir la relación $\prec$ en S como :*

$x \prec y$ si y sólo si $x \preceq y$ y $x \neq y$

Por ejemplo: si $\preceq$ es la inclusión $\subseteq$, entonces $A \prec B$ significa que A es un subconjunto propio de B, es decir $A \subset B$. La relación $\prec$ satisface:

$(AR) s \prec s$ es falso para toda s en S;

(T) $s \prec t$ y $t \prec u$ implica $s \prec u$.

Una relación que satisface (AR) es *antireflexiva*. Una relación que satisface (AR) y (T) es un *orden parcial estricto*. Cada orden parcial en S genera un orden parcial estricto y recíprocamente si $\prec$ es un orden parcial estricto en S, entonces la relación $\preceq$ definida por:

$x \preceq y$ si y sólo si $x \prec y$ o $x = y$

es un orden parcial en S. Dependiendo del problema en particular de que se trate, se trabaja con el orden parcial o con el orden parcial estricto asociado. Generalmente se trabajará con orden parcial.

Definición 1.2.3 ([15]) *Si $(P, \preceq)$ es un copo, decimos que $x \in P$ es máximal si no existen $y \in P$ tal que $x < y$ y llamamos a x minimal si no existe $y \in P$ tal que $y < x$.*

Si S es un subconjunto de un copo $(P, \preceq)$, puede suceder que S tenga un elemento M tal que $s \preceq M$ para toda s en S. Un elemento M con esta propiedad se llama *el mayor elemento* del S o *el máximo* de S y se denota $máx(S)$. De manera similar, si S tiene un elemento m tal que $m \preceq s$ para toda s en S, entonces m es *el emento menor* del conjunto o el *mínimo* de S y se denota por $mín(S)$.

A pesar de que un subconjunto S de un copo $(P, \preceq)$ no tenga elemento máximo, puede existir un elemento x en P tal que $s \preceq x$ para toda s en S. Un elemento que satisfaga esto se llama *cota superior* para S en P. Si x es una cota superior de S en P tal que $x \preceq y$ para toda y cota superior de S en P, entonces x es llamada *mínima cota superior* (least upper bound) o *supremo* de S en P y la denotamos $lub(S) = x$. De manera similar un elemento z en P tal que $z \preceq s$ para toda s en S es una *cota inferior* para S en P. Una cota inferior z tal que $w \preceq z$ para toda cota inferior w de S en P se llama *máxima cota inferior* (greatest lower bound) o *ínfimo* de S en P y la denotamos $glb(S) = z$. Por la propiedad *antisimétria* (AS), un subconjunto de P no puede tener dos mínimas cotas superiores distintas o dos máximas cotas inferiores distintas.

Definición 1.2.4 ([2]) *Sea P un conjunto parcialmente ordenado no vacío.*

(i) *Si para todo $x, y \in P$ existen el $lub\{x, y\}$ y $glb\{x, y\}$ entonces P es llamado retículo.*

(ii) *Si para todo $S \subset P$ existen el lub S y glb S entonces P es llamado retículo completo.*

Algunas propiedades del *ínfimo* (glb) y *supremo* (lub) de un conjunto ordenado y de un *retículo* son las siguientes.

Lema 1.2.1 ([2]) *Sea P un conjunto ordenado y $S, T \subseteq P$, asumimos que lub(S), lub(T), glb(S) y glb(T) existen en P. Entonces:*

(i) Para todo $s \in S$, $s \leq lub(S)$ y $s \geq glb(S)$.

(ii) Sea $x \in P$; entonces $x \leq glb(S)$ si y sólo si $x \leq s$ para toda $s \in S$.

(iii) Sea $x \in P$; entonces $x \geq lub(S)$ si y sólo si $x \geq s$ para toda $s \in S$.

(iv) $lub(S) \leq glb(T)$ si y sólo si $s \leq t$ pata toda $s \in S$ y toda $t \in T$.

(v) Si $S \subseteq T$, entonces $lub(S) \leq lub(T)$ y $glb(S) \geq glb(T)$.

Lema 1.2.2 ([2]) *Sea P un retículo y $S, T \in P$, asumimos que $lub(S)$, $lub(T)$, $glb(S)$ y $glb(T)$ existen en P. Entonces $lub(S \cup T) = lub\{lub(S),\ lub(T)\}$ y $glb(S \cup T) = \{glb(S), glb(T)\}$*

Lema 1.2.3 ([2]) *Sea P un retículo y $a, b \in P$. Entonces las siguientes relaciones son equivalentes:*

(i) $a \leq b$.

(ii)$lub(a, b) = b$.

(iii) $glb(a, b) = a$.

Un caso especial de los copos es la existencia de un orden total o lineal definido acontinuación.

Definición 1.2.5 ([15]) *Sea S un conjunto, un orden parcial se llama orden total o lineal si para cada elección de s y t en S, $s \preceq t$ o $t \preceq s$. Entonces llamaremos a S un conjunto totalmente ordenado o conjunto linealmente ordenado.*

Ejemplo 1.5 *Ejemplos de conjuntos totalmente ordenados:*

a) El conjunto $\mathbb{R}$ (números reales) con el orden usual $\leq$ es un conjunto totalmente ordenado.

b) Las listas de nombres en un directorio telefónico o de palabras en un diccionario, son conjuntos totalmente ordenados si definimos $w_1 \leq w_2$ como $w_1 = w_2$ o w_1 aparece antes de w_2.

Todo subcopo de un conjunto totalmente ordenado a su vez es un conjunto totalmente ordenado. Por ejemplo los copos $(\mathbb{Z}, \leq)$ y $(\mathbb{Q}, \leq)$ son subcopos de $(\mathbb{R}, \leq)$ y están totalmente ordenados por el orden que heredan de $\mathbb{R}$.

Definición 1.2.6 *Una función f es monótona si $X \leq Y$ implica que $f(X) \leq f(Y)$. Una función f es inflationary si $X \leq f(X)$. Una función f es deflationary si $f(x) \leq x$.*

1.3 Programación de orden parcial

En esta sección se presentarán algunos conceptos importantes de la programación de orden parcial como es la sintaxis de los programas y algunos conceptos de la semántica declarativa.

1.3.1 Sintaxis

Un programa P es una pareja $< PO, PA >$ donde PO es un conjunto de *cláusulas de orden parcial* y DA es un conjunto de *cláusulas datalog*. Las cláusualas de orden parcial son de la forma:

$$f(terms) \geq expression \text{ :- } p_1(terms), \ldots, p_k(terms)$$
$$f(terms) \leq expression \text{ :- } p_1(terms), \ldots, p_k(terms)$$

donde cada variable en *expression* también ocurre en el cuerpo de las cláusulas, $p_1(terms)$, ..., $p_k(terms)$ donde $k \geq 0$ y cada predicado p_i es definido por un *programa datalog* o en otra palabras por un conjunto de *cláusulas datalog*. Cuando $k = 0$, tendremos clásulas incondicionales de la forma:

$$f(terms) \geq expression$$
$$f(terms) \leq expression$$

La sintaxis de *terms* y *expression* está dada por la siguiente gramática:

term ::= *variable* | *constant* | *c*(*terms*)
terms ::= *term* | *term* , *terms*
expression ::= *term* | *c*(*exprs*) | *f*(*exprs*)
exprs ::= *expression* | *expression* , *exprs*

El símbolo c es un estándar para cualquier constructor, mientras que f es un estándar para símbolos no constructores también llamados símbolos definidos por el usuario. La sintaxis de las *cláusulas datalog* está determinada por la siguiente gramática:

rule ::= *atom* | *atom* :- *body*
body ::= *literal* | *literal*, *body*
literal ::= ∼ *atom* | *atom*
atom ::= *p*(*terms*)

En la programación de orden parcial se tiene la convención de iniciar los símbolos de constantes con letras minúsculas y los símbolos de las variables con letras mayúsculas. Cabe destacar que las cláusulas de orden parcial costituyen un paradigma de programación funcional y por consiguiente todas las llamadas a las funciones se realizan con argumentos ground. También se hace uso del término *consulta*, con esto nos referimos a un llamado a una función. La sintaxis de una consulta es de la siguiente forma:

$$f(ground - term)$$

En el caso general, múltiples cláusulas de orden parcial son usadas para definir a una función. Para ilustrar un poco la forma en que se modela un programa en este paradigma veamos los siguientes ejemplos:

Ejemplo 1.6 (Distancia más corta en un grafo) *La formulación del problema de encontrar la distancia más corta en un grafo es una de las más elegantes e ilustrativas del uso de cláusulas de orden parcial.*

```
distancia(X,Y) ≤ C1 :- arco(X,Y,C1).
distancia(X,Y) ≤ C2 + distancia(Z,Y) :- arco(X,Z,C2).
```

La relación `arco(X,Y,C1)` *significa que existe un arco del vértice* `X` *al vértice* `Y` *con una distancia* `C1`. *Observe que la función* `distancia(X,Y)` *está definida mediante la relación* $\leq$. *Esto quiere decir informalmente que la* `distancia(X,Y) = mínimo{C1, C2 + distancia(Z,Y)}`.

Otro interesante ejemplo, es la formulación del problema de la mochila.

Ejemplo 1.7 (Problema de la mochila) *El planteamiento del problema de la mochila es el siguiente: Se tienen n objetos con las ganancias* p_i *y los costos* w_i, *para* $1 \leq i \leq n$, *y una capacidad de almacenamiento de* M. *Se quiere maximizar* $\sum x_i p_i$, *sujeto a* $\sum x_i w_i \leq M$, *y* $x_i \in \{0,1\}$, $1 \leq i \leq n$. *El valor de uno en* x_i *indica que el objeto i se mete a la mochila teniendo una ganancia de* p_i, *y la capacidad de la mochila sera* $M - w_i$ *y el valor de cero en* x_i *que no se mete a la mochila el objeto i. La formulación de este problema con cláuslas de orden parcial es la siguiente:*

$\texttt{kn}_{01}(\texttt{0},\texttt{M}) \geq \texttt{0}.$
$\texttt{kn}_{01}(\texttt{I},\texttt{M}) \geq \texttt{kn}_{01}(\texttt{I}-1,\texttt{M}) :- \texttt{I} \geq 1.$
$\texttt{kn}_{01}(\texttt{I},\texttt{M}) \geq \texttt{kn}_{01}(\texttt{I}-1,\texttt{M}-\texttt{c(I)}) + \texttt{p(I)}$ `:- I` $\geq$ `1, c(I)` $\leq$ `M.`

La función `c(I)` *define el costo del objeto* `I`, *la función* `p(I)` *define la ganacia del objeto* `I`. *Note que la función* $\texttt{kn}_{01}$ *está definida mediante la relación* $\geq$ *que informalmente nos indica que* $\texttt{kn}_{01}$`(I,M) = máximo { 0,` $\texttt{kn}_{01}$`(I-1,M),` $\texttt{kn}_{01}$`(I-1, M - c(I)) + p(I) }`.

1.3.2 Semántica declarativa

Una parte importante en todo lenguaje formal es su semántica declarativa, es por esto que se presentarán algunos conceptos de ésta.

Para la presentación de la semántica declarativa, primero definiremos la noción de *reducir un programa con respecto a su sección de cláusulas datalog.* La reducción consta de dos pasos. El primero, consiste en reducir el programa datalog en una base de datos extencional (EDB)[3]. Para esto es necesario que el programa datalog sea estratificado para asegurar que el proceso de reducción termina en una EDB. En este primer paso, se hace uso del algoritmo

[3]Una *base de datos extencional (EDB)* es un conjunto de átomos ground.

presentado en [1]. Este proceso tiene un comportamiento polinomial con respecto al tamaño de la EDB. Otro posible algoritmo para reducir el programa datalog es el presentado en [9]. En el segundo paso, se reduce el programa original con respecto a la EDB obtenida en el primer paso.

Cabe hacer mención que los algoritmos a utilizar en el primer paso se basan principalmente en reglas de trasformación de programas lógicos como las que se definen en [13]; esto es gracias a que su tiempo de cálculo es polinomial. Una descripción de cómo utilizar reglas de transformación en programas datalog es presentada en [8]. De hecho existe una gran variedad de aplicaciones alrededor de la trasformación de programas lógicos, como las que se discuten en [12].

Definición 1.3.1 *La reducción de un programa P con respecto a una EDB E, es denotada por R(P) y definida como sigue:*

$R(P) = \{(g(t) \leq e)\theta \mid g(t) \leq e$ `:-` $e_1, .., e_n \in P, \theta$ *es una sustitución tal que cada* $(e_i\theta) \in E\}$

Ejemplo 1.8 *Consideremos nuevamente el programa del ejemplo 1.6, con la siguiente EDB:*

```
arco(a,b,2).
arco(a,c,3).
```

Entonces reduciendo la función `distancia` *con respecto al predicado* `arco` *obtenemos el siguiente programa:*

```
distancia(a,b) ≤ 2.
distancia(a,c) ≤ 3.
distancia(a,Y) ≤ 2 + distancia(b,Y).
distancia(a,Y) ≤ 3 + distancia(c,Y).
```

Ahora se definirá la clase de programas legales para la semántica. En la siguiente definición se asume que el programa ya fue reducido con respecto a su EDB.

Definición 1.3.2 *Un programa P es generalmente estratificado si existe una función, nivel* : $F \rightarrow N$, *que mapea desde F que es el conjunto de símbolos de las funciones definidas por el usuario (no constructores) en P a (subconjunto finito de) los números naturales N tal que:*

(i) Toda cláusula de la forma $f(term_1) \geq term_2$ es válida. Toda cláusula de esta forma es llamada cláusula S-S (S-S es una abreviación de stronly-stratified).

(ii) Para cláusulas de la forma

$f(term) \geq g(expre)$

donde f y g son funciones definidas por el usuario, $nivel(f)$ es mayor o igual al $nivel(g)$ y $nivel(f)$ es mayor que $nivel(h)$, donde h es un símbolo de función definida por el usuario que ocurre en expre. También este tipo de cláusulas son llamadas cláusulas S-S.

(iii) Para cláusulas de la forma

$f(term) \geq m(g(expre))$

donde m es una función monótona, nivel(f) es mayor que nivel(m), nivel(f) es mayor o igual a nivel(g) y nivel(f) es mayor que nivel(h), donde h es un símbolo de función definida por el usuario que ocurre en expre. Las cláusulas de esta forma son llamadas cláusulas G-S (G-S es una abreviación de general-stratified).

(iv) Toda cláusula de otra forma es no válida.

Aunque un programa puede tener diferentes niveles de mapeo, se selecciona un conjunto imagen que parta del número 1. En la anterior definición, note que las funciones f y g no son necesariamente diferentes. También la dependencia de funciones no monótonas ocurre sólo con funciones de bajo nivel. Observe que se puede tener mayor libertad en la definición anterior. Partiendo de la idea que la composición de funciones monótonas es monótona entonces la función m puede ser reemplazada por una composición de funciones monótonas.

Para la presentación de la semántica operacional en el siguiente capítulo se asume que el programa ya fue reducido con respecto a su sección de cláusulas datalog y su EDB y además que toda cláusula está en un *formato plano*. Para la presentación del formato plano veamos primero un ejemplo y posteriormente daremos la definición formal.

Ejemplo 1.9 *Asumimos que f, g, h y k son funciones definidas por el usuario y las restantes funciones son constructores, el formato plano de la cláusula.*

$$f(c(X,Y)) \leq c1(c2(g(c3(X)), k(d1(h(Y,1)))))$$

es el siguiente:

$$f(c(X,Y)) \leq c1(c2(Y1,Y3)) : -g(c3(X)) = Y1, h(Y,1) = Y2, k(d1(Y2)) = Y3.$$

En la cláusula anterior, seguimos la convención de Prolog usando $: -$ para $'si'$ y comas para la conjunción.

Definición 1.3.3 *La forma general de una cláusula en forma plana es:*

$$Cabeza : -Cuerpo$$

donde la Cabeza es de la forma $f(t) \leq u$, y t y u son términos, y el Cuerpo es de la forma $E_1, ..., E_n$. Cada E_i es de la forma $f_i(t_i) = y_i$, donde cada f_i es un símbolo de una función definida por el usuario, cada y_i es una nueva variable no presentada en la Cabeza, y cada t_i es un término que es equivalente al argumento de f_i, en la cláusula original sin formato plano. Cada fórmula $f_i(t_i) = y_i$ en el Cuerpo es llamada meta básica, y una lista de metas básicas es llamada una secuencia de metas básicas.

El orden en que las metas básicas son listadas en el lado derecho de una cláusula en formato plano es: el más a la izquierda es el más interno $(leftmost - innermost)$.

Semántica basada en modelos

La semántica declarativa de la programación de orden parcial se basa en el concepto de *modelo*, es por esto, la necesidad de esta sección. Para la presentación se utilizará la interpretación de Herbrand, donde el universo de Herbrand de un programa P sólo consiste de términos ground, y es representado como U_p. La base de Herbrand B_p de un programa P consiste de igualdades ground de la forma $f(t) = u$, donde f es una función definida por el usuario, t es un término ground y u es un término ground perteneciente a algún dominio(retículo completo). Dados dos términos ground t_1 y t_2, se escribe $t_1 \leq t_2$ para denotar que t_1 es menor o igual a t_1 en algun retículo.

La presentación de la semántica es sin referencia a un retículo específico tal como conjuntos, números, etc. Esto permite una presentación enfocada a nuestro objetivo principal, que son las cláusulas de orden parcial sin perder objetividad en la presentación.

Asumimos que toda interpretación I siempre incluye la igualdad y desigualdad de átomos de la forma $t_1 = t_2$ y $t_1 \leq t_2$, de acuerdo con la interpretación fija de estos en el programa. También, en toda interpretación I, f es interpretada como una función total, es decir:

(i) $(\forall t \in U_p)(\ \exists u \in U_p)\ f(t) = u$ *; y*

(ii) $f(t) = t_1 \in I$ *y* $f(t) = t_2 \in I \Rightarrow t_1 = t_2$.

Definición 1.3.4 ([7]) *Sea P un programa. Una interpretación M es un modelo de P, denotado por $M \vDash P$, si para toda instancia ground, $f(t) \leq t_1 : -E_1, ..., E_k$, de una cláusula $\leq$ en P y si $\{E_1, ..., E_k\} \subseteq M$ entonces existe un átomo $f(t) = u \in M$ y $u \leq t_1$.*

Se motivará brevemente el enfoque de semánticas basadas en modelos. Básicamente, se definirá la semántica de una función llamada $f(t)$, donde t es un término ground, para ser el *lub* de todos los términos definidos para $f(t)$ en los diferentes modelos de Herbrand para f. Para ver la necesidad de tomar los *lubs*, considere el siguiente programa trivial P:

$f(x) \leq 1$

Asumamos que el dominio resultante para f es el retículo de números totalmente ordenados, $N : 0 \leq 1 \leq 2...\top$(donde $\top$ denota el elemeto mayor del retículo). Cada modelo de P interpreta a f como una función constante.

f(x) = 0, para toda $x \in U_p$

f(x) = 1, para toda $x \in U_p$

El modelo asignado a la función f, es $f(x) = 1$ para toda $x \in U_p$, note que este modelo no es obtenido por la intersección clásica de conjuntos ($\cap$) de todos los modelos, pero en vez se usa el *lub* de los términos definidos para $f(x)$ en los diferentes modelos. En el ejemplo anterior, el *lub* es el *máximo*. Una importante propiedad en la programación de orden parcial es garantizada por el teorema siguente:

Teorema 1.3.1 ([7]) *Todo programa de orden parcial es consistente.*

Pero sintácticamente no todo programa bien formado tiene un significado bien definido. La circularidad en funciones definidas es permisible, a lo largo de su ocurrencia mediante funciones monótonas. Consideremos el siguiente programa (tomado de [7]) donde *not* es el operador de negación común, y como sabemos *not* no es una función monótona con respecto al retículo boleano $false \leq true$ y $not(false) = true$ y $not(true) = false$.

$a \geq not(b)$

$b \geq not(a)$

Este programa tiene tres modelos:

$\{a = true, b = true\}$

$\{a = true, b = false\}$

$\{a = false, b = true\}$

Sin embargo, tomando el glb[4] de los términos definidos para a y b respectivamente en los tres modelos obtenemos:

$\{a = false, b = false\}$

que claramente no es un modelo. Se puede observar que las funciones no monótonas no son permisibles cuando estas son definiciones circulares mediante funciones. Esto motiva a considerar programas de orden parcial estratificados.

Definición 1.3.5 ([7]) *Sea P un programa generalmente estratificado. Se define P_k como aquellas cláusulas de P para las cuales el símbolo de la función del lado izquierdo de la cláusula tiene un nivel $\leq k$.*

Definición 1.3.6 ([7]) *Dadas dos interpretaciones I y J para un programa P, se define $I \sqsubseteq J$ si $\forall f(t) = t_1 \in I$ existe $f(t) = t_2 \in J$ tal que $t_1 \leq t_2$. Se dice que $I = J$ si $I \sqsubseteq J$ y $J \sqsubseteq I$*

La construcción de la semántica basada en modelos de un programa generalmente estratificado se realizará nivel por nivel. Por este motivo se define una interpretación limitada por algún nivel.

Definición 1.3.7 ([7]) *Para cada interpretación I, $I_k := \{A : A \in I \wedge nivel(A) \leq k\}$*

Definición 1.3.8 ([7]) *Dadas dos interpretaciones I y J de un programa P, $I \sqcup J := \{f(t) = lub(u, u') \quad : \quad f(t) = u \in I, \quad f(t) = u' \in J, \; f$ un símbolo de función de P, $t \in U_P \}$*

[4]En este caso calculamos el *glb* de los términos porque el programa está definido por cláusulas $\geq$ que es el caso dual de las cláusulas $\leq$.

Definición 1.3.9 ([7]) *Para cualquier conjunto X de interpretaciones, $\sqcup X$ es la natural generalización de la anterior definición.*

Proposición 1.3.1 ([7]) *Sea X un conjunto de modelos para un programa P con nivel j tal que para cualquier $I \in X$ y $J \in X$, $I_{j-1} = J_{j-1}$. Entonces $\sqcup X$ es también un modelo.*

Definición 1.3.10 ([7]) *Sea P un programa con j niveles, se define la semántica basada en modelos de P como:*

for $j = 1$, $\mathcal{M}(P_1) := \sqcup\{M : M \models P_1\}$, *y*

for $j > 1$, $\mathcal{M}(P_j) := \sqcup\{M : M_{j-1} = \mathcal{M}(P_{j-1})$ *y* $M \models P_j\}$.

Definición 1.3.11 ([7]) *Sea P un programa con j niveles y G un conjunto de secuencia de metas, se dice que la sustitución θ es una respuesta correcta para G si*

$$\mathcal{M}(P_j) \models \bigwedge_{g \text{ es una meta basica en } G\theta} g$$

La noción de sustitución máxima es necesaria en este enfoque como se verá más adelante. En la siguiente definición, se usa la función $Vars(\theta)$ para hacer referencia al dominio de la sustitución θ. Por ejemplo $Vars(\{X \leftarrow 1, Y \leftarrow 2\}) = \{X, Y\}$.

Definición 1.3.12 (Sustitución Máxima [7]) *Sea θ_1 y θ_2 sustituciones. Se define $\theta_1 \leq \theta_2$ como $Vars(\theta_1) = Vars(\theta_1)$ y $\forall X$, si $X \leftarrow s_1 \in \theta$ entonces existe un s_2 tal que $X \leftarrow s_2 \in \theta \wedge s_1 \leq s_2$. Dada una sustitución, decimos que una sustitución es máximal si es el elemento máximo en el conjunto ordenado por $\leq$ como se definió anteriormente.*