Lógica Computacional
Nota 06. Lógica de predicados.*

Noé Salomón Hernández S.

1. La necesidad de un lenguaje más expresivo

Aunque la lógica proposicional puede lidiar satisfactoriamente con componentes de las oraciones como '*no*', '*y*', '*o*' y '*si ... entonces*' los aspectos lógicos de los lenguajes naturales y artificiales van mucho más allá que ellos. Hay limitaciones de la lógica proposicional con respecto a modificadores del lenguaje como '*existe...*', '*todos...*', '*cada...*', y '*solo...*'. El deseo de expresar oraciones declarativas más detalladas conduce al diseño de la *lógica de predicados*.

Considere la oración declarativa

Todo estudiante es más joven que algún instructor.

En la lógica proposicional, identificamos esta afirmación con la variable proposicional p. Esto falla en reflejar la estructura lógica más fina de la oración anterior. Se trata de *ser un estudiante*, *ser un instructor*, y de *ser más joven que alguien más*. Estas son propiedades de algún tipo, nos gustaría tener un mecanismo para expresarlas junto con sus relaciones lógicas y dependencias.

Usamos *predicados* para tal fin. Por ejemplo, $S(antonio)$ denota que Antonio es un estudiante, mientras que $I(pablo)$ dice que Pablo es un instructor. Así, $Y(antonio, pablo)$ indica que Antonio es más joven que Pablo. Los símbolos S, I e Y son llamados predicados.

Estos predicados no son suficientes. No queremos escribir todas las instancias de $S(\cdot)$ donde $\cdot$ es reemplazado por cada uno de los nombres de los estudiantes. Por lo tanto, ocupamos el concepto de *variable*, como $x, y, z, \ldots$, las cuales pueden ser consideradas como *parámetros* o *posiciones* para valores concretos. De este modo tenemos

$$\begin{aligned} S(x): &\quad x \text{ es un estudiante} \\ I(y): &\quad y \text{ es un instructor} \\ Y(x,y): &\quad x \text{ es más joven que } y. \end{aligned}$$

Las variables son simplemente posiciones para objetos concretos. La disponibilidad de variables no es suficiente para capturar la esencia de la oración con la que iniciamos. Necesitamos transmitir el significado de '**Todo** *estudiante* x *es más joven que* **algún** *instructor* y', para ello introducimos los *cuantificadores* $\forall$ (para todo) y $\exists$ (existe), los cuales siempre vienen pegados a una variable, como en $\forall x$ (para toda x) o en $\exists z$ (existe z o hay z). La oración inicial podemos escribirla simbólicamente como

$$\forall x \Big(S(x) \to \big(\exists y \ (I(y) \wedge Y(x,y)) \big) \Big).$$

*El material aquí presentado se basa el libro de Huth y Ryan, *Logic in Computer Science*; en las notas del prof. Francisco Hernández Q.; y en las notas del prof. Favio Miranda.

Diferentes predicados pueden tener diferente número de argumentos. Los predicados con cualquier número finito de argumentos son posibles en la lógica de predicados.

Otro ejemplo es la oración

No todas las aves vuelan.

Para ella escogemos los predicados:

$B(x)$: x es un ave
$F(y)$: x vuela.

'No todas las aves vuelan' puede ser codificado como

$$\neg\Big(\forall x\ \big(B(x) \to F(x)\big)\Big)$$

También es correcta la codificación:

$$\exists x\ \big(B(x) \wedge \neg F(x)\big)$$

Explicaremos más adelante porque fórmulas como las dos anteriores son equivalentes *semánticamente*.

La lógica de predicados extiende a la lógica proposicional con *símbolos de funciones*. Considere la oración declarativa:

Cada niño es más joven que su mamá.

Usando lógica de predicados lo anterior se expresa como

$$\forall x\ \big(C(x) \to \exists y\ (M(y,x) \wedge Y(x,y))\big)$$

donde $C(x)$ significa que x es un niño, $M(y,x)$ que y es la mamá de x, y $Y(x,y)$ que x es más joven que y. Otra formulación es,

$$\forall x\ \forall y\ \big(C(x) \wedge M(y,x) \to Y(x,y)\big)$$

Para todo niño x y *toda* mamá y de ese niño, x es más joven que y. Hay que notar que la implicación anterior es trivialmente verdadera para toda aquella persona que no es mamá del niño en cuestión porque el antecedente es falso, incluso si nadie cumple con ser su mamá. Cuando se trata de toda mama de ese niño el antecedente es verdadero y el consecuente también lo debe ser. No es elegante decir 'toda mamá de x', ya que sabemos que cada individuo tiene sólo una mamá biológica. Esto se ejemplifica aun más en la siguiente oración:

Antonio y Pablo tienen la misma abuela materna.

lo cual se expresa como sigue, teniendo en cuenta que a es por Antonio y p por Pablo, y M es el predicado binario recién definido,

$$\forall x\ \forall y\ \forall u\ \forall v\ \big(M(x,y) \wedge M(y,a) \wedge M(u,v) \wedge M(v,p) \to x = u\big).$$

Si y y v son las mamás de Antonio y Pablo, respectivamente, y x y u son las mamás de ellas, i.e. abuelas maternas de Antonio y Pablo, respectivamente, entonces x y u son la misma persona.

Hemos usado un predicado especial en la lógica de predicados, *igualdad* (=). Se escribe entre dos argumentos, $x = y$ en lugar de $= (x, y)$. Otra forma de expresar esta oración es,

$$\exists x\ \exists y\ \exists u\ \exists v\ \big(M(x,y) \wedge M(y,a) \wedge M(u,v) \wedge M(v,p) \wedge x = u\big)$$

Los símbolos de función nos brindan una forma de evitar esta pobre codificación, ya que nos permiten representar a la mamá de y de un modo más directo. En lugar de escribir $M(x, y)$, escribimos $m(y)$. Usando la función m, la dos oraciones anteriores se simplifican dando como resultado:

$$\forall x\big(C(x) \to Y(x, m(x))\big)$$

todo niño es más joven que su mamá. La representación de que Antonio y Pablo tienen la misma abuela materna queda

$$m(m(a)) = m(m(p)).$$

Los símbolos de función pueden ser utilizados sólo en situaciones en la cuales queremos denotar un único objeto. Así, no podemos tener un símbolo de función $b(\cdot)$ para 'hermano'. Puede que no tenga sentido hablar del hermano de x, ya que x puede no tener o tener más de uno. 'Hermano' debe codificarse como un predicado binario.

Si Bárbara tiene varios hermanos, la afirmación 'a Ana le gusta el hermano de Bárbara' es ambigua. Puede suceder que a Ana le guste uno de los hermanos de Bárbara, lo cual se escribe como:

$$\exists x\ \big(B(x,b) \wedge L(a,x)\big)$$

donde B y L significan 'es hermano de' y 'le gusta', y a y b significan Ana y Bárbara. Si a Ana le gustan todos los hermanos de Bárbara, lo escribimos como:

$$\forall x\ \big(B(x,b) \to L(a,x)\big)$$

Funciones con cero argumentos se llaman *constantes*: a y b arriba son contantes para Ana y Bárbara, respectivamente. Si el dominio hace referencia a estudiantes y la calificación que reciben, se podría tener una función binaria $g(\cdot, \cdot)$ que toma dos argumentos; $g(x, y)$ se refiere a la calificación del estudiante x en el curso y.

2. Lógica de predicados como un lenguaje formal

Hay dos clases de elementos que forman parte de la lógica de predicados. La primera clase denota a los objetos de los que hablamos: individuos como a y b, como las variables x y v, o como los que resultan al aplicar funciones. Las expresiones en la lógica de predicados que denotan objetos se llaman *términos*. El universo o dominio de discurso es el conjunto de objetos que se consideran en el razonamiento.

La otra clase de elementos denota valores de verdad; expresiones en esta clase se llaman *fórmulas*: $Y(x, m(x))$ es una fórmula, con x y $m(x)$ términos. Dentro de las fórmulas tenemos a los predicados que representan a las propiedades o relaciones de los objetos del dominio de discurso. Así un mismo predicado, digamos $Y(x, y)$, está representando un número potencialmente infinito de fórmulas a las que se les asigna un valor de verdad, uno por cada posible combinación de x y y.

Un vocabulario para predicados consiste de dos conjuntos: (i) un conjunto de símbolos de predicado $\mathcal{P}$ y (ii) un conjunto de símbolos de función $\mathcal{F}$. Cada símbolo de predicado y cada símbolo de función vienen con una aridad escrita como superíndice. Las constantes se pueden

considerar como funciones que no toman ningún argumento. Por lo tanto, las constantes son parte del conjunto $\mathcal{F}$ junto con las funciones "reales" que sí toman argumentos. De ahora en adelante estipularemos a las constantes como funciones de aridad 0, que llamaremos *nulas*.

2.1. Términos

Los términos constan de variables, constantes y símbolos de función.

Definición 2.1 Los términos se definen como sigue.

- Una variable es un término.
- Si $c \in \mathcal{F}$ es una función nula, entonces c es un término.
- Si $t_1, \ldots, t_n$ son términos y $f \in \mathcal{F}$ tiene aridad $n > 0$, entonces $f(t_1, \ldots, t_n)$ es un término.
- Nada más es un término.

La gramática en BNF para la definición anterior es:

$$t ::= x \mid c \mid f(t, \ldots, t)$$

donde x toma valores de un conjunto de variables var, c de los símbolos de función nulos $\mathcal{F}$, y f sobre aquellos elementos de $\mathcal{F}$ con aridad $n > 0$.

2.2. Fórmulas

La elección de los conjuntos $\mathcal{P}$ y $\mathcal{F}$ para los símbolos de predicado y función, respectivamente, se toma de acuerdo a lo que uno desea describir.

Definimos el conjunto de fórmulas sobre $(\mathcal{F}, \mathcal{P})$ inductivamente a través de la siguiente gramática en BNF:

$$\varphi ::= P(t_1, t_2, \ldots, t_n) \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \mid (\varphi \leftrightarrow \varphi) \mid (\forall x\, \varphi) \mid (\exists x\, \varphi)$$

donde $P \in \mathcal{P}$ es un símbolo de predicado con aridad $n \geq 1$, t_i son términos sobre $\mathcal{F}$ y x es una variable.

Las fórmulas de la lógica de predicados pueden representarse por árboles de análisis sintáctico. Por ejemplo, el árbol de análisis sintáctico en la Figura 1 representa la fórmula $\forall x\ ((P(x) \to Q(x)) \wedge S(x, y))$.

Convención Por conveniencia, retenemos la precedencia acordada para la lógica proposicional y agregamos la referente a $\forall x$ y $\exists x$ como sigue:

- $\neg$, $\forall x$ y $\exists x$ tienen la precedencia más alta;
- luego $\wedge$ y $\vee$;
- luego $\to$, que es asociativo a la derecha;
- por último, $\leftrightarrow$.

Ejercicios 2.2 Considere los símbolos de constante a y b; los de función $f^{(1)}$ y $g^{(2)}$; y los de predicado $P^{(1)}$, $R^{(2)}$ y $Q^{(3)}$, donde el superíndice en los símbolos de función y de predicado indica el número de argumentos que reciben. Diga si las siguientes expresiones son fórmulas de la lógica de predicados:

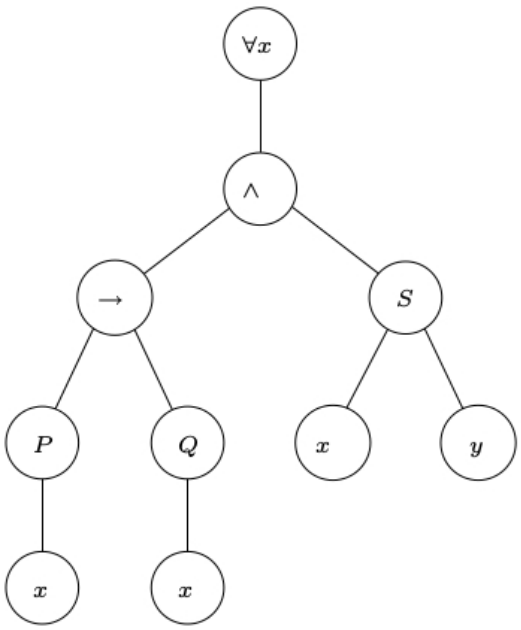

Figura 1: Árbol de análisis sintáctico para una fórmula de la lógica de predicados.

- $Q(a)$,
- $P(y)$,
- $Q(x, P(a), b)$,
- $\neg R(x, a)$,
- $R\big(a, g(a, f(a))\big)$,
- $\forall x\ \neg P(x)$,
- $\exists a\ R(a, a)$,
- $\forall R(x, a)$,
- $\exists x\ \big(Q(x, f(x), b) \rightarrow \forall x\ R(a, x)\big)$,
- $a \rightarrow P(b)$,
- $R(x, b) \vee \neg \exists y\ g(y, b)$,
- $\forall x \exists y \big(R(x, y) \rightarrow R(y, x)\big)$,

2.3. Especificación formal

Algunas frases del español no se pueden especificar de una manera completamente fiel a la lógica de predicados. Se presentan a continuación una serie de recomendaciones a tomar en cuenta durante el proceso de especificación.

- Se especifican o traducen oraciones declarativas o proposiciones.
- 'y' se traduce como $\wedge$, también 'pero'.
- La disyunción es incluyente.
- Frases como *para todos*, *para cualquier*, *todos*, *cualquiera*, *los*, *las*, etc. se traducen usando el cuantificador universal $\forall$.
- Frases como *para algún*, *existe un*, *alguno*, *alguna*, *uno*, *una*, etc. se traducen usando el cuantificador existencia $\exists$. Aunque hay excepciones, por ejemplo, *si alguien se avienta del*

bungee se arriesga se puede interpretar como *cualquiera que se avienta del bungee se arriesga*, su especificación es $\forall x\big(B(x) \to A(x)\big)$.

- Es común que una especificación compuesta que involucre cuantificadores pueda formarse como alguno de los cuatro juicios aristotélicos:

  Universal afirmativo. *Todo S es P* se traduce como $\forall x\big(S(x) \to P(x)\big)$.

  Existencial afirmativo. *Algún S es P* se traduce como $\exists x\big(S(x) \wedge P(x)\big)$.

  Universal negativo. *Ningún S es P* se traduce como $\forall x\big(S(x) \to \neg P(x)\big)$.

  Existencial negativo. *Algún S no es P* se traduce como $\exists x\big(S(x) \wedge \neg P(x)\big)$.

  Considerando a los predicados S y P como conjuntos, los juicios aristotélicos anteriores pueden interpretarse del siguiente modo:

  Universal afirmativo. *Todo S es P* puede entenderse como $S \subseteq P$.

  Existencial afirmativo. *Algún S es P* puede entenderse como $S \cap P \neq \varnothing$.

  Universal negativo. *Ningún S es P* puede entenderse como $S \subseteq \overline{P}$.

  Existencial negativo. *Algún S no es P* puede entenderse como $S \cap \overline{P} \neq \varnothing$.

- Las variables no se mencionan en español. $\forall x\big(S(x) \to T(x)\big)$ debe escribirse en español como *Cualquier superhéroe viste un traje* y no como *para cualquier x, si x es superhéroe, entonces x viste un traje.*

- $\forall x(\varphi \to \psi)$ y $\exists x(\varphi \wedge \psi)$ son útiles y comunes. Menos comunes son $\forall x(\varphi \wedge \psi)$, $\forall x(\varphi \vee \psi)$, y $\exists x(\varphi \vee \psi)$.

- $\exists x(\varphi \to \psi)$ es muy raro que aparezca como una traducción del español. Bajo ese patrón la fórmula $\exists x\big(B(x) \to \forall y B(y)\big)$ es un teorema para un universo no vacío de individuos, como se verá en la Nota 08. Este teorema tiene a menudo la siguiente interpretación: *hay alguien que si bebe, entonces todos beben.* Sin embargo, es difícil que se busque tener una situación así al momento de especificar una fórmula en la lógica de predicados.

- El hecho que se usen dos o más variables distintas no implica que éstas representen a objetos distintos del universo. Para especificar dos objetos distintos no es suficiente con variables distintas. Las fórmulas $\exists x P(x)$ y $\exists x \exists y(P(x) \wedge P(y))$ expresan lo mismo, algo cumple P. Se debe agregar explícitamente que x y y tienen la propiedad de ser distintos, con el predicado $x \neq y$.

- En la fórmula $\forall x \exists y\, \varphi$ la variable y depende del valor de x; por ejemplo, en el cálculo diferencial e integral se define el límite de la función $f(x)$, cuando x tiende a c, como L si y solo si $\forall \varepsilon > 0\, \exists \delta > 0\ (0 < |x - c| < \delta) \to (|f(x) - L| < \varepsilon)$, para $f(x) = 3x - 1$, $c = 1$ y $L = 2$, la definición se cumple tomando $\delta = \varepsilon/3$ para cualquier ε.

  Esta dependencia no está presente en la fórmula $\exists y \forall x\, \psi$; por ejemplo, en los números naturales se tiene que $\exists n \forall m\ (n \leq m)$, con $n = 0$ se satisface la fórmula.

Ejemplo 2.3 Considere las constantes m, a y p para representar a la primera persona del singular, a Amélie y a Pablo, respectivamente. Así m, a y p son términos. Escogemos $\{F^{(2)}, M^{(2)}, H^{(2)}, S^{(2)}, B^{(2)}\}$ como el conjunto de predicados con significado:

$F(x,y)$:	x es padre de y	$S(x,y)$:	x es la hermana de y
$M(x,y)$:	x es madre de y	$B(x,y)$:	x es el hermano de y.
$H(x,y)$:	x es el cónyuge de y		

Traduzca las siguientes oraciones a lógica de primer orden.

1. *Todo hijo de mi padre es mi hermano.*

   a) Representamos a *mi padre* como una función, para esto $f^{(1)}$ representa a la función que regresa el padre del único argumento que recibe. Así tenemos,

   $$\forall x\big(F(f(m),x) \to B(x,m) \vee S(x,m)\big)$$

   b) Representamos padre exclusivamente como un predicado. La codificación para la oración es

   $$\forall x\forall y\big(F(x,m) \wedge F(x,y) \to B(y,m) \vee S(y,m)\big)$$

2. *Todos tienen una madre.*

   $$\forall x\exists y M(y,x)$$

3. *Cualquier persona que tenga una madre tiene un padre.*

   $$\forall x\big(\exists y M(y,x) \to \exists z F(z,x)\big)$$

4. *Ninguna abuela es el padre de todo el mundo.*

   a) Representando padre ($f^{(1)}$) y madre ($m^{(1)}$) como función.

   $$\forall x\Big(\neg\forall w\big(F(m(m(x)),w)\big) \wedge \neg\forall w\big(F(m(f(x)),w)\big)\Big)$$

   b) Representando padre y madre como predicado.

   $$\forall x\forall y\forall z\Big(\big(F(y,z) \vee M(y,z)\big) \wedge M(x,y) \to \neg\forall w F(x,w)\Big)$$

   Lo que es equivalente a

   $$\forall x\forall y\forall z\Big(\big(F(y,z) \wedge M(x,y) \to \neg\forall w F(x,w)\big) \wedge \big(M(y,z) \wedge M(x,y) \to \neg\forall w F(x,w)\big)\Big)$$

5. *Pablo es el cuñado de Amélie.*

   $$\exists x\big(S(x,a) \wedge H(p,x)\big) \vee \exists y\big(H(y,a) \wedge B(p,y)\big)$$

6. *Todos los tíos de Amélie y de Pablo son hermanos.*

   a) Representando padre ($f^{(1)}$) y madre ($m^{(1)}$) como función.

   $$\forall x\forall w\Big(\big(B(x,f(a)) \vee B(x,m(a))\big) \wedge \big(B(w,f(p)) \vee B(w,m(p))\big) \to B(x,w)\Big)$$

   b) Representando padre y madre como predicado.

   $$\forall x\forall y\forall w\forall z\Big(\big(F(y,a) \vee M(y,a)\big) \wedge B(x,y) \wedge \big(F(z,p) \vee M(z,p)\big) \wedge B(w,z) \to B(x,w)\Big)$$

7. *Todos los tíos en común de Amélie y de Pablo son padres.*

   a) Representando padre(f) y madre(m) como función.

$$\forall x\Big(\big(B(x,f(a))\vee B(x,m(a))\big)\wedge\big(B(x,f(p))\vee B(x,m(p))\big)\rightarrow\exists wF(x,w)\Big)$$

   b) Representando padre y madre como predicado.

$$\forall x\forall y\forall z\Big(\big(F(y,a)\vee M(y,a)\big)\wedge B(x,y)\wedge\big(F(z,p)\vee M(z,p)\big)\wedge B(x,z)\rightarrow\exists wF(x,w)\Big)$$

Las especificaciones formales requieren de *conocimiento del dominio específico*. Expertos en el tema no hacen explícito todo el conocimiento, así que un especificador puede ignorar importantes restricciones para un modelo. Por ejemplo, algunas especificaciones arriba expuestas pueden parecer correctas, pero ¿qué sucede si los argumentos del predicado B son iguales? Si las relaciones familiares no son del domino común, entonces el especificador puede no darse cuenta que un hombre no puede ser su propio hermano. Así, algunas fórmulas de este ejemplo serían incorrectas.

Ejercicio

¿Cuál es la formalización más adecuada para la oración *Todas la mujeres son rockeras, con la posible excepción de Adela*, dado el siguiente diccionario? Diccionario: a: Adela, $M(x)$: x es mujer, y $D(x)$: x es rockera.

1. $\forall x(M(x)\rightarrow D(x))\wedge\neg D(a)$
2. $\forall x(M(x)\wedge a\neq x\rightarrow D(x))$
3. $\forall x(D(x)\rightarrow M(x)\wedge\ a\neq x)\wedge\neg D(a)$
4. $\forall x((M(x)\wedge a\neq x)\equiv D(x))$
5. $\forall x(D(x)\rightarrow M(x)\wedge a\neq x)$

La opción más cercana a transmitir la idea presentada es la número 2. Esta declaración afirma que toda persona que es mujer, exceptuando Adela, es rockera, pero no tenemos información contundente respecto a que si Adela es rockera o no, así que *posiblemente* ella lo sea.

Es interesante notar que si buscamos la formalización para *Todas la mujeres son rockeras, con la certera excepción de Adela*, tendríamos lo siguiente

$$\forall x(M(x)\wedge a\neq x\rightarrow D(x))\wedge\neg D(a)$$

2.4. Inducción estructural para la lógica de predicados

Los principios de inducción para términos y fórmulas se enuncian a continuación.

Definición 2.4 (Principio de inducción estructural para términos) Sea P una propiedad acerca de términos. Para demostrar que P es válida para todos los términos, llevamos a cabo los siguientes pasos:

Base de inducción: Se demuestra que

- Si x es una variable, entonces P se cumple para x.
- Si c es una constante, entonces P se cumple para c.

Paso inductivo: Suponiendo que P se cumple para los términos $t_1, \ldots t_n$, hay que demostrar que si $f \in \mathcal{F}$ es un símbolo de función de aridad $n > 0$, entonces $f(t_1, \ldots t_n)$ cumple P.

Definición 2.5 (Principio de inducción estructural para fórmulas) Sea P una propiedad acerca de fórmulas. Para demostrar que P es válida para todas las fórmulas, llevamos a cabo los siguientes pasos:

Base de inducción: Demostrar que si $P \in \mathcal{P}$ es un símbolo de predicado con aridad $n > 0$ y $t_1, \ldots t_n$ son términos sobre $\mathcal{F}$, entonces $P(t_1, \ldots t_n)$ cumple con P.

Paso inductivo: Suponiendo que P se cumple para las fórmulas φ y ψ, hay que demostrar que

- $(\neg\varphi)$ cumple P.
- $(\varphi \bowtie \psi)$ cumple P, donde $\bowtie \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.
- $\forall x\varphi$ y $\exists x\varphi$ cumplen P.

2.5. Variables libres y ligadas

Definición 2.6 Sea φ una fórmula de la lógica de predicados. Una presencia de x en φ es **libre** si es un nodo hoja del árbol de análisis sintáctico correspondiente a φ tal que no hay trayectoria ascendente de ese nodo x a un nodo $\forall x$ o $\exists x$. De otro modo, esa presencia de x está **ligada**. El alcance de $\forall x$ en $\forall x\ \varphi$ es φ —a excepción de cualquier subfórmula de φ que esté cuantificando a x por su cuenta propia. Análogamente se define el alcance de $\exists x$ en $\exists x\ \varphi$.

Si x figura en φ, x está ligada si y sólo si está dentro del alcance de algún $\exists x$ o $\forall x$.

Definición 2.7 Sea φ una fórmula. El conjunto de variables libres de φ se denota $FV(\varphi)$, donde $FV(\varphi) = \{x \mid x \text{ es una variable que figura libre en } \varphi\}$.

El papel que desempeña una variable libre es la de un *parámetro*, que representa un elemento arbitrario o genérico del dominio. Este uso de las variables libres es evidente en las reglas de deducción natural que se estudian en la siguiente nota. Además, una fórmula con variables libres es verdadera si en lugar de tales variables se considera cualquier elemento del dominio, y la fórmula que resulta es verdadera. La semántica de la lógica de predicados se estudia a detalle en la Nota 8.

Definición 2.8 Se tienen los siguientes conceptos:

- Un *término cerrado* es un término que no contiene ninguna variable.
- Una *fórmula atómica cerrada* es una fórmula atómica cuyos términos son todos cerrados.
- Una *literal cerrada* es una fórmula atómica cerrada o la negación de una.
- Una *fórmula es cerrada* si no tiene variables libres. Una fórmula cerrada también se conoce como enunciado.
- Una *fórmula cerrada sin cuantificadores* es una fórmula libre de cuantificadores, cuyas fórmulas atómicas son cerradas.

Ejercicios 2.9 Dibuje el árbol de análisis sintáctico para las siguientes fórmulas. Indique cuales son fórmulas cerradas y cuales son abiertas.

- $\forall x\, P(a)$,
- $\forall x \exists y\, (Q(x,y) \vee R(x,y))$
- $P(x) \to \exists x\, Q(x, f(x))$
- $\forall x\, \big(\neg \forall y\, Q(x,y) \wedge R(x,y)\big)$

2.6. Sustitución

La noción de sustitución en la lógica de predicados es más complicada debido a la existencia de términos y de variables ligadas. Las sustituciones deben respetar el ligado de variables.

2.6.1. Sustitución en términos

La sustitución de la variable x por el término t en un término t', denotada $t'[x := t]$, se define recursivamente sobre la estructura de t' como sigue,

$$\begin{aligned} c[x := t] &= c \\ x[x := t] &= t \\ y[x := t] &= y \quad \text{si } x \neq y \\ f(t_1, \ldots, t_n)[x := t] &= f(t_1[x := t], \ldots, t_n[x := t]) \end{aligned}$$

2.6.2. Sustitución en fórmulas

La sustitución en fórmulas necesita cuidados especiales debido a la presencia de variables ligadas mediante cuantificadores. Deben evitarse las siguientes situaciones:

- **Sustitución de variables ligadas.** La sustitución en fórmulas **no** debe ser textual ya que las variables ligadas no pueden sustituirse. La *solución* consiste en definir la sustitución únicamente sobre variables libres.
- **Captura de variables libres.** Considere la fórmula $\forall x\, \varphi$, la cual es cierta siempre y cuando φ sea cierta para cualquier valor de x. Dado $\forall x\, \varphi$, nos gustaría llegar a la verdad de cualquier sustitución del estilo $\varphi[x := t]$. Tomemos ahora $\forall x \exists y\, (x \neq y)$, que indica que para cualquier individuo existe otro distinto a él, esto es cierto si el universo consta de al menos dos objetos. Sin embargo, al sustituir x por y sin mayor cuidado tenemos:

  $$\begin{aligned} \big(\exists y\, (x \neq y)\big)[x := y] &= \exists y \big((x \neq y)[x := y]\big) \\ &= \exists y\, (y \neq y) \end{aligned}$$

  lo cual es absurdo. El problema es que la presencia libre de x en $\exists y\, (x \neq y)$ se ligó al sustituirla por y. Las posiciones que correspondan a una variable libre representan a objetos particulares y el permitir que se liguen al realizar una sustitución modificará el significado intensional de la fórmula.

  Para *solucionar* este problema, la sustitución en una fórmula se define renombrando las variables ligadas, puesto que los nombres de las variables ligadas no importan; por ejemplo, $\forall x P(x)$ y $\forall y P(y)$ significan lo mismo, que todos los objetos cumplen el predicado P.

  Para realizar la sustitución $\big(\exists y\, (x \neq y)\big)[x := y]$, se renombraría la variable ligada y, digamos $\big(\exists z\, (x \neq z)\big)[x := y]$, y ahora sí y sustituiría a x, lo que resulta en $\exists z\, (y \neq z)$.

Definición 2.10 Decimos que dos fórmulas φ y ψ son α-equivalentes, lo cual se escribe $\varphi \sim_\alpha \psi$ si y sólo si φ y ψ difieren únicamente en los nombres de sus variables ligadas.

Las fórmulas α-equivalentes son lógicamente equivalentes, de modo que son intercambiables en cualquier contexto. El concepto de α-equivalencia se emplea implícitamente en la siguiente definición de sustitución para fórmulas.

La sustitución de x por el término t en una fórmula φ, $\varphi[x := t]$, de define recursivamente sobre la estructura de φ como,

$$
\begin{aligned}
P(t_1, \ldots, t_n)[x := t] &= P(t_1[x := t], \ldots, t_n[x := t]) \\
(\neg\varphi)[x := t] &= \neg(\varphi[x := t]) \\
(\varphi_1 \wedge \varphi_2)[x := t] &= \varphi_1[x := t] \wedge \varphi_2[x := t] \\
(\varphi_1 \vee \varphi_2)[x := t] &= \varphi_1[x := t] \vee \varphi_2[x := t] \\
(\varphi_1 \to \varphi_2)[x := t] &= \varphi_1[x := t] \to \varphi_2[x := t] \\
(\varphi_1 \leftrightarrow \varphi_2)[x := t] &= \varphi_1[x := t] \leftrightarrow \varphi_2[x := t] \\
(\forall x\ \varphi)[x := t] &= \forall x\ \varphi \\
(\forall y\ \varphi)[x := t] &= \forall y\ (\varphi[x := t]) \quad \text{si } y \notin t \\
(\forall y\ \varphi)[x := t] &= \forall z\ \Big((\varphi[y := z])[x := t]\Big) \\
&\qquad \text{si } y \in t, \text{ con } z \text{ una variable nueva} \\
(\exists x\ \varphi)[x := t] &= \exists x\ \varphi \\
(\exists y\ \varphi)[x := t] &= \exists y\ (\varphi[x := t]) \quad \text{si } y \notin t \\
(\exists y\ \varphi)[x := t] &= \exists z\ \Big((\varphi[y := z])[x := t]\Big) \\
&\qquad \text{si } y \in t, \text{ con } z \text{ una variable nueva}
\end{aligned}
$$

Ejemplo 2.11 Lleve a cabo las sustituciones que aparecen a continuación:

- $\Big(\forall x\big(Q(z, y, x) \wedge \exists z T(f(z), w, y)\big)\Big)[x, y, z := a, z, g(w)]$

  $= \Big(\forall x\big(Q(z, y, x) \wedge \exists z T(f(z), w, y)\big)\Big)[y, z := z, g(w)]$

  $= \forall x\Big((Q(z, y, x))[y, z := z, g(w)] \wedge (\exists z T(f(z), w, y))[y, z := z, g(w)]\Big)$

  $= \forall x\Big(Q(g(w), z, x) \wedge (\exists u T(f(u), w, y))[y, z := z, g(w)]\Big)$

  $= \forall x\big(Q(g(w), z, x) \wedge \exists u T(f(u), w, z)\big)$

- $\Big(\big(\forall y\, P(x,y,z) \wedge \forall x\, Q(x,y,z)\big)[x := f(y,z)]\Big)[y := g(x,z)]$

  $= \big((\forall y\, P(x,y,z))[x := f(y,z)] \wedge (\forall x\, Q(x,y,z))[x := f(y,z)]\big)[y := g(x,z)]$

  $= \big(\forall u\, P(x,u,z)[x := f(y,z)] \wedge \forall x\, Q(x,y,z)\big)[y := g(x,z)]$

  $= \big(\forall u\, P(f(y,z),u,z) \wedge \forall x\, Q(x,y,z)\big)[y := g(x,z)]$

  $= \big(\forall u\, P(f(y,z),u,z)\big)[y := g(x,z)] \wedge \big(\forall x\, Q(x,y,z)\big)[y := g(x,z)]$

  $= \forall u\, P\big(f(g(x,z),z),u,z\big) \wedge \big(\forall v\, Q(v,y,z)\big)[y := g(x,z)]$

  $= \forall u\, P\big(f(g(x,z),z),u,z\big) \wedge \forall v\, Q(v,g(x,z),z)$

Ejercicios 2.12 Realice las siguientes sustituciones.

- $\big(\exists y\, Q(y,z,y)\big)[y,z := g(d), f(y)]$
- $\Big(\big(\forall x \exists y\, P(x,z) \vee Q(y)\big)[x := f(z)]\Big)[y := g(z,x)]$

3. Especificación funcional y relacional con tipos

En ciencias de la computación es importante conocer de donde provienen los objetos con los que trabajamos, por esa razón necesitamos tener predicados para tipos.

3.1. Predicados para tipos

Considere la siguiente oración declarativa *Existe un entero que es más pequeño que cualquier natural* cuya traducción puede pensarse como $\exists x \forall y\, (x < y)$; sin embargo, se pierde información por lo que se prefiere la traducción con tipos $\exists x\, \big(Int(x) \wedge \forall y\, (Nat(y) \to x < y)\big)$.

Introducimos abreviaturas para ahorrarnos algunos conectivos:

$$\forall x : \mathsf{A}.P(x) \text{ significa } \forall x\, (A(x) \to P(x)),$$

$$\exists x : \mathsf{A}.P(x) \text{ significa } \exists x\, (A(x) \wedge P(x)).$$

La traducción anterior se expresa como $\exists x : \mathsf{Int}\ \forall y : \mathsf{Nat}\, (x < y)$. Adoptamos la convención de que los tipos aparecen en otro tipo de letra.

Ejemplo 3.1 Mostramos el uso de tipos al realizar la traducción de las siguientes oraciones declarativas respecto a libros y autores. Considere los siguientes predicados y constantes:

$L(x,y)$: x es más extenso que y
$E(x,y)$: x está escrito por y
$M(x)$: x es matemático
$F(x)$: x es filósofo
$B(x)$: x es un libro
q : "El Quijote de la Mancha"
c : "La construcción lógica del mundo"
a : "Alicia en el país de las maravillas"

Las oraciones y sus traducciones aparecen a continuación:

- *Hay un libro más extenso que "El Quijote de la Mancha".*

$$\exists x : \mathsf{B}.L(x, q).$$

- *"La construcción lógica del mundo" está escrito por un filósofo y no es más extenso que "Alicia en el país de las maravillas".*

$$\big(\exists x : \mathsf{F}.E(c, x)\big) \wedge \neg L(c, a).$$

- *Si "Alicia en el país de las maravillas" está escrito por un matemático, entonces está escrito por un filósofo.*

$$\big(\exists x : \mathsf{M}.E(a, x)\big) \rightarrow \big(\exists y : \mathsf{F}.E(a, y)\big).$$

- *No es el caso que cualquier libro es escrito por un filósofo.*

$$\neg\forall x : \mathsf{B}.\exists y : \mathsf{F}.E(x, y).$$

- *Hay alguien que es filósofo y matemático y ha escrito un libro más extenso que "Alicia en el país de las maravillas".*

$$\exists x\big(F(x) \wedge M(x) \wedge \exists y : \mathsf{B}.(E(y, x) \wedge L(y, a))\big)$$

3.2. Especificación de listas

En lenguajes de programación dado un tipo de datos A se define el tipo de datos de listas con elementos de A, denotado L_A, como sigue:

- La lista vacía, *nil*, es un miembro de L_A.
- Si a es un elemento de A y ℓ es una lista de L_A, entonces $a : \ell$ es un miembro de L_A.
- Son todos.

Para especificar el tipo de datos de listas con elementos de A debemos considerar a *nil* como un símbolo de constante para la lista vacía; $:^{(2)}$ un símbolo de función (infijo) que construye una lista[†]; y los símbolos de predicado $A^{(1)}$ y $L_A^{(1)}$ que denotan el tipo A y listas L_A. La formalización es la siguiente:

▷ $L_A(nil)$,

▷ $\forall x : \mathsf{A}.\forall \ell : \mathsf{L_A}.\ L_A(x : \ell)$.

Ejemplo 3.2 Realice la especificación formal de las siguientes propiedades de las funciones concatenación y reversa de listas. En cada caso dar dos especificaciones, una funcional y otra relacional.

i. La operación de concatenación es asociativa.

[†]En PROLOG las listas son una estructura de datos incorporada en el lenguaje, donde *nil* y $h : t$ corresponden a los patrones `[]` y `[h|t]`, respectivamente.

ii. Si la concatenación de dos listas es vacía, entonces ambas listas son vacías.

iii. Si la concatenación de dos listas es una lista unitaria, entonces una de tales listas es vacía y la otra es la misma lista unitaria.

iv. La reversa de la reversa de una lista es la misma lista.

Especificación funcional

Usaremos las constantes, funciones y predicados empleados en la definición del tipo de datos de listas con elementos de A; además de los símbolos de funciones $concat^{(2)}$ y $rev^{(1)}$, para la concatenación y reversa de listas, respectivamente. Las especificaciones que buscamos son:

i. $\forall \ell : \mathsf{L_A}.\forall m : \mathsf{L_A}.\forall n : \mathsf{L_A}\ \big(concat(concat(\ell, m), n) = concat(\ell, concat(m, n))\big)$.

ii. $\forall \ell : \mathsf{L_A}.\forall m : \mathsf{L_A}.\ \big(concat(\ell, m) = nil \to \ell = nil \wedge m = nil\big)$.

iii. $\forall \ell : \mathsf{L_A}.\forall m : \mathsf{L_A}.\forall x : \mathsf{A}\ \big(concat(\ell, m) = x : nil \to (\ell = x : nil \wedge m = nil) \vee (\ell = nil \wedge m = x : nil)\big)$.

iv. $\forall \ell : \mathsf{L_A}.\ \big(rev(rev(\ell)) = \ell\big)$.

Especificación relacional

Usaremos las constantes, funciones y predicados empleados en la definición del tipo de datos de listas con elementos de A; además de los siguientes predicados,

$$\begin{aligned} C(x, y, z) &: \ z \text{ es la concatenación de } x \text{ y } y \\ R(x, y) &: \ y \text{ es la reversa de } x \end{aligned}$$

Las especificaciones que buscamos son:

i. $\forall \ell : \mathsf{L_A}.\forall m : \mathsf{L_A}.\forall n : \mathsf{L_A}\forall u : \mathsf{L_A}.\forall v : \mathsf{L_A}.\forall w : \mathsf{L_A}\ \big(C(\ell, m, u) \wedge C(u, n, \boldsymbol{v}) \wedge C(m, n, w) \to C(\ell, w, \boldsymbol{v})\big)$.

ii. $\forall \ell : \mathsf{L_A}.\forall m : \mathsf{L_A}.\ \big(C(\ell, m, nil) \to \ell = nil \wedge m = nil\big)$.

iii. $\forall \ell : \mathsf{L_A}.\forall m : \mathsf{L_A}.\forall x : \mathsf{A}\ \big(C(\ell, m, x : nil) \to (\ell = x : nil \wedge m = nil) \vee (\ell = nil \wedge m = x : nil)\big)$.

iv. $\forall \ell : \mathsf{L_A}.\forall \ell' : \mathsf{L_A}.\ \big(R(\ell, \ell') \to R(\ell', \ell)\big)$.

Ejercicio

Realizar la siguiente especificación formal de dos maneras, con un enfoque funcional y relacional.

1. La lista vacía está ordenada.
2. Si la lista tiene un único elemento, entonces está ordenada.
3. Si una lista ℓ con al menos un elemento está ordenada y si x es menor que la cabeza de ℓ, entonces la lista $(x : \ell)$ está ordenada.