Lógica de Primer Orden: Introducción y Sintaxis
Lógica Computacional 2023-I, Nota de clase 7

Favio Ezequiel Miranda Perea Araceli Liliana Reyes Cabello
Lourdes Del Carmen González Huesca Pilar Selene Linares Arévalo

11 de noviembre de 2022
Facultad de Ciencias UNAM

1. Introducción

La lógica ha sido estudiada desde los tiempos de Aristóteles. En sus inicios se dedicó a fundamentar de manera sólida el análisis del razonamiento humano, formalizando las "leyes del pensamiento", es decir, las nociones de argumento válido o inválido. Aristóteles aisló principios lógicos llamados *silogismos* los cuales fueron utilizados para analizar y verificar argumentos hasta la edad media por lógicos como Peter Abelard o Guillermo de Ockham.

En el siglo 19 George Boole y otros desarrollaron versiones algebráicas de la lógica debido a las limitaciones de la teoría del silogismo para manejar relaciones binarias, por ejemplo "x es hijo de y". Esto dio nacimiento a la lógica algebraica, formalismo que aun existe aunque ha perdido popularidad.

Al final del siglo 19 Gottlob Frege desarrollo la lógica cuantificada sentando así los cimientos de gran parte de la lógica moderna, aún cuando Bertrand Russell le mostró que su sistema era inconsistente. Al mismo tiempo Charles S. Pierce de manera independiente, propuso una lógica similar.

La *paradoja de Russell* generó una crisis en los fundamentos de la matemática la cual se resolvío al inicio del siglo XX usando la lógica de primer orden, la cual funge hasta la actualidad como un fundamento de las matemáticas modernas. Debido a su gran poder, la lógica de primer orden también tiene grandes limitaciones que fueron descubiertas en los años 1930 por Kurt Gödel, Alan Turing y Alonzo Church, entre otros. Gödel dio un sistema deductivo completo y correcto para la lógica de primer orden, Alfred Tarski le dio la semántica formal que estudiaremos y que es la base para el estudio de las semánticas denotacionales de los actuales lenguajes de programación y Gerhard Gentzen y otros desarrollarón la teoría de la demostración. A partir de este punto la lógica de primer orden toma su forma actual.

Durante la siguiente parte del curso nos dedicaremos a estudiar los siguientes aspectos de la lógica de primer orden:

- Su sintaxis y semántica formal.
- El proceso de especificación formal del español a la lógica.
- Los métodos para establecer válidez de argumentos:

1. Argumentación semántica (método útil pero prácticamente imposible de implementar de manera eficaz).
2. Tableaux semánticos (método fácilmente implementable)
3. Resolución binaria (método fundamental de la programación lógica)
4. Deducción natural (método completo y correcto, fundamento de los sistemas de tipos para lenguajes de programación).

1.1. Lógica proposicional: expresividad vs. decidibilidad

Por un lado, observemos que la lógica proposicional no es lo suficientemente expresiva. Considérese por ejemplo el siguiente razonamiento:

Algunas personas van al teatro. Todos los que van al teatro se divierten.
De manera que algunas personas se divierten.

La intuición nos dice que el argumento es correcto. Sin embargo la representación correspondiente en lógica proposicional es:

$$p,\ q\ /\therefore\ r \qquad \text{¡Incorrecto!}$$

Otros ejemplos de expresiones que no pueden ser formalizadas adecuadamente en la lógica proposicional son:

La lista está ordenada.

Cualquier empleado tiene un jefe.

Si los caballos son animales entonces las cabezas de caballos son cabezas de animales.

Este problema de expresividad se soluciona al introducir la lógica de predicados.

Por otro lado, la lógica proposicional es decidible, es decir, existen diversos algoritmos para responder a las preguntas

$$\text{¿ } \models \varphi \text{ ?, ¿ Existe } \mathcal{I} \text{ tal que } \mathcal{I} \models \varphi \text{ ?,}$$
$$\text{¿ Es } \Gamma \text{ satisfacible?, ¿ } \Gamma \models \varphi \text{ ?}$$

Como vimos anteriormente, su estudio es de gran importancia en la teoría de complejidad computacional (problema SAT). Esta propiedad, de gran importancia desde el punto de vista computacional, se pierde en la lógica de predicados.

Así, la lógica proposicional nos ofrece decibilidad de algunas propiedades pero nos limita respecto a la expresividad. La lógica de predicados o de primer orden que estudiaremos, nos ofrecerá mayor expresividad para formalizar expresiones más complejas.

Antes de empezar el estudio formal de la lógica de primer orden discutiremos con detalle un ejemplo de especificación no trivial en lógica de proposiciones. De aquí podremos observar una aplicación directa del problema SAT fuera del ámbito de la lógica. Esta clase de representaciones son típicas de los métodos de reducción de problemas en teoría de la complejidad.

2. Lógica de predicados de manera informal

En la lógica proposicional debemos representar a ciertas frases del español como por ejemplo

Algunos programas usan ciclos anidados

mediante fórmulas proposicionales atómicas, es decir mediante una simple variable proposicional p. En la lógica de predicados descomponemos a este tipo de frases distinguiendo dos categorias: objetos y relaciones entre objetos. Ambas categorias construidas sobre un universo de discurso fijo el cual varía según el problema particular que se desee representar. Estudiemos con mas detalle estas categorias:

2.1. Objetos y universo de discurso

*Algunas **personas** van al teatro*
*Todos los **programas** en Java usan clases.*

- Universo o dominio de discurso: conjunto de objetos que se consideran en el razonamiento, pueden ser cosas, personas, datos, objetos abstractos, etc.
- En el caso de los enunciados anteriores en el universo podemos tener personas y programas en Java, y posiblemente el teatro y las clases.

2.2. Relaciones entre objetos

- Las propiedades o relaciones atribuibles a los objetos del dominio de discurso se representan con predicados.
- En el caso de los enunciados anteriores tenemos *ir al teatro, usar clases.*

2.3. Cuantificadores

Los cuantificadores especifican el número de individuos que cumplen con el predicado. Por ejemplo:

- *Todos* los estudiantes trabajan duro.
- *Algunos* estudiantes se duermen en clase.
- *La mayoría* de los maestros están locos
- *Ocho de cada diez* gatos prefieren croquetas Michi Miau.
- *Nadie* es más tonto que yo.
- *Al menos seis* estudiantes están despiertos.
- *Hay una infinidad* de números primos.
- *Hay más* PCs que Macs.

En lógica de primer orden sólo consideraremos dos cuantificadores: *todos* y *algunos.*

2.4. Proposiciones vs. Predicados

El uso de predicados en lugar de proposiciones podría parecer simplemente otra manera de escribir las cosas, por ejemplo, la proposición:

Chubaka recita poesía nordica

se representa con predicados como

Recita(Chubaka, poesía nordica)

La gran ventaja es que el predicado puede cambiar de argumentos, por ejemplo

Recita(Licantro, odas en sánscrito)

o en el caso general podemos usar variables para denotar individuos:

Recita(x,y)

Obsérvese que está última expresión no es una proposición. En este sentido un mismo predicado está representando un número potencialmente infinito de proposiciones.

Reconozcamos ahora las componentes anteriores en un argumento particular:

Todos los matemáticos estudian algún teorema. Los teoremas son elementales o trascendentes.
Giuseppe es matemático. Luego entonces, Giuseppe estudia algo elemental o trascendente.

- Universo: *matemáticos y teoremas*
- Predicados: *estudiar, elemental, trascendente.*
- Cuantificadores: *todos, los, algo.*
- Los individuos se representarán formalmente mediante dos categorías:
  - Constantes: las cuales denotan individuos particulares, como Giuseppe.
  - Variables: las cuales denotan individuos genéricos o indeterminados como los matemáticos o el teorema.
    Obsérvese que las variables no figuran en el lenguaje natural como el español, pero son necesarias para la formalización y representación de individuos no fijos.

Formalicemos ahora todos los conceptos discutidos anteriormente.

3. Sintaxis de la lógica de primer orden

En contraposición al lenguaje de la lógica proposicional que está determinado de manera única, no es posible hablar de un solo lenguaje para la lógica de predicados. Dependiendo de la estructura semántica que tengamos en mente será necesario agregar símbolos particulares para denotar objetos y relaciones entre objetos. De esta manera el alfabeto consta de dos partes ajenas entre sí, la parte común a todos los lenguajes está determinada por los símbolos lógicos y auxiliares y la parte particular, llamada tipo de semejanza o signatura del lenguaje. La parte común a todos los lenguajes consta de:

- Un conjunto infinito de variables $\mathsf{Var} = \{x_1, \ldots, x_n, \ldots\}$

- Constantes lógicas: $\bot, \top$
- Conectivos u operadores lógicos: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- Cuantificadores: $\forall, \exists$.
- Símbolos auxiliares: (,) y , (coma).
- Si se agrega el símbolo de igualdad $=$, decimos que el lenguaje tiene igualdad.

La signatura $\Sigma = \mathcal{P} \cup \mathcal{F} \cup \mathcal{C}$ de un lenguaje en particular está dada por:

- Un conjunto $\mathcal{P}$, posiblemente vacío, de símbolos o letras de predicado: $P_1, \ldots, P_n, \ldots$
  A cada símbolo se le asigna un índice [1] o número de argumentos m, el cual se hace explícito escribiendo $P_n^{(m)}$ que significa que el símbolo P_n necesita de m argumentos.
- Un conjunto $\mathcal{F}$, posiblemente vacío, de símbolos o letras de función: $f_1, \ldots, f_n, \ldots$
  Análogamente a los símbolos de predicado cada símbolo de función tiene un índice asignado, $f_n^{(m)}$ significará que el símbolo f_n necesita de m argumentos.
- Un conjunto $\mathcal{C}$, posiblemente vacío, de símbolos de constante: $c_1, \ldots, c_n, \ldots$
  En algunos libros los símbolos de constante se consideran como parte del conjunto de símbolos de función, puesto que pueden verse como funciones de índice cero, es decir, funciones sin argumentos.

Es importante enfatizar que los tres conjuntos de símbolos que conforman una signatura son siempre ajenos dos a dos.

Dado que un lenguaje de primer orden queda determinado de manera única por su signatura, abusaremos de la notación y escribiremos $\mathcal{L} = \mathcal{P} \cup \mathcal{F} \cup \mathcal{C}$ para denotar al lenguaje dado por la signatura correspondiente Σ.

3.1. Términos

Los términos del lenguaje son aquellas expresiones que representarán individuos del universo en la semántica y ya fueron estudiados a detalle. Recordamos aquí su definición.

- Los símbolos de constante $c_1, \ldots, c_n, \ldots$ son términos.
- Las variables $x_1, \ldots, x_n, \ldots$ son términos.
- Si $f^{(m)}$ es un símbolo de función y $t_1, \ldots, t_m$ son términos entonces $f(t_1, \ldots, t_m)$ es un término.
- Son todos.

Es decir, los términos están dados por la siguiente gramática:

$$t ::= x \mid c \mid f(t_1, \ldots, t_m)$$

Y el conjunto de términos de un lenguaje dado se denota con $\mathsf{TERM}_{\mathcal{L}}$, o simplemente TERM si es claro cuál es el lenguaje.

[1] Este índice suele llamarse también "aridad", pero dado que tal palabra no existe en el diccionario de la lengua española, trataremos de evitar su uso.

3.2. Fórmulas

El siguiente paso es determinar las expresiones o fórmulas *atómicas*, dadas por:

- Las constantes lógicas $\bot, \top$.
- Las expresiones de la forma: $P_1(t_1, \ldots, t_n)$ donde $t_1, \ldots, t_n$ son términos
- Las expresiones de la forma $t_1 = t_2$, si el lenguaje cuenta con igualdad

El conjunto de expresiones atómicas se denotará con $\mathsf{ATOM}_{\mathcal{L}}$.

Así el conjunto $\mathsf{FORM}_{\mathcal{L}}$ de expresiones compuestas aceptadas en un lenguaje $\mathcal{L}$, llamadas usualmente fórmulas, se define recursivamente como sigue:

- Si $A \in \mathsf{ATOM}_{\mathcal{L}}$ entonces $A \in \mathsf{FORM}_{\mathcal{L}}$. Es decir, toda fórmula atómica es una fórmula.
- Si $A \in \mathsf{FORM}_{\mathcal{L}}$ entonces $(\neg A) \in \mathsf{FORM}_{\mathcal{L}}$.
- Si $A, B \in \mathsf{FORM}_{\mathcal{L}}$ entonces $(A \wedge B), (A \vee B), (A \to B), (A \leftrightarrow B) \in \mathsf{FORM}_{\mathcal{L}}$.
- Si $A \in \mathsf{FORM}_{\mathcal{L}}$ y $x \in \mathsf{Var}$ entonces $(\forall x A), (\exists x A) \in \mathsf{FORM}_{\mathcal{L}}$.
- Son todas.

En una fórmula de la forma $\forall x A$ o $\exists x A$ la variable x se conoce como la variable del cuantificador $\forall$ o $\exists$ respectivamente. Es decir, la variable de un cuantificador es aquella que está escrita junto a dicho cuantificador.

La gramática para las fórmulas en forma de Backus-Naur es:

$$\begin{array}{rcl} A, B & ::= & at \mid (\neg A) \mid (A \wedge B) \mid (A \vee B) \mid (A \to B) \mid (A \leftrightarrow B) \mid (\forall x A) \mid (\exists x A) \\ at & ::= & \bot \mid \top \mid P(t_1, \ldots, t_m) \mid t_1 = t_2 \end{array}$$

Cada lenguaje definido a partir de una signatura y conforme a la gramática recién descrita será un *lenguaje de primer orden*.
Las siguientes observaciones son de suma importancia:

- Los términos y las fórmulas son dos categorías sintácticas ajenas, es decir **ningún** término es fórmula y **ninguna** fórmula es término.
- Los términos denotan exclusivamente individuos u objetos.
- Las fórmulas atómicas denotan proposiciones o propiedades acerca de los términos.
- **Sólo** los individuos u objetos son cuantificables y **únicamente** sobre ellos se puede "predicar". Esta característica justifica la denominación "primer orden".

3.3. Precedencia

Si bien los cuantificadores no son propiamente operadores entre fórmulas y por lo tanto no se puede hablar de una precedencia en el sentido usual, usamos la siguiente convención:

Los cuantificadores se aplican a la *mínima* expresión sintácticamente posible.

de manera que

$$\begin{array}{rcl} \forall x A \to B & \text{es} & (\forall x A) \to B \\ \exists y A \wedge \forall w B \to C & \text{es} & (\exists y A) \wedge (\forall w B) \to C \end{array}$$

4. Inducción y Recursión

Las definiciones recursivas para los elementos del lenguaje generan principios de inducción. Dado que ahora tenemos dos categorías sintácticas, términos y fórmulas, tenemos también principios de inducción correspondientes a cada categoría. Si bien ya hemos discutido y empleado inducción y recursión para términos, repetimos aquí los esquemas correspondientes observando que en muchas ocasiones para dar una definición recursiva sobre las fórmulas (para probar una propiedad de fórmulas por inducción) es necesario dar previamente una definición recursiva similar para términos (dar una prueba por inducción para términos). Por ejemplo para definir la lista de subtérminos de una fórmula se requiere de la lista de subtérminos de un término y para probar propiedades de sustitución en fórmulas se necesita probar propiedades análogas de sustitución en términos.

Esquema de Definición Recursiva para Términos Para definir una función $h : \mathsf{TERM}_{\mathcal{L}} \to A$, basta definir h como sigue:

- Definir $h(x)$ para $x \in \mathsf{Var}$.
- Definir $h(c)$ para cada constante $c \in \mathcal{C}$.
- Suponiendo que $h(t_1), \ldots, h(t_n)$ están definidas, definir $h(f(t_1, \ldots, t_n))$ para cada símbolo de función $f \in \mathcal{F}$ de índice n, utilizando $h(t_1), \ldots, h(t_n)$.

Esquema de Definición Recursiva para Fórmulas . Para definir una función $h : \mathsf{FORM}_{\mathcal{L}} \to A$, basta definir h como sigue:

- Definir h para cada fórmula atómica, es decir, definir $h(\bot), h(\top), h(P(t_1, \ldots, t_n))$ y $h(t_1 = t_2)$ si el lenguaje tiene igualdad.
- Suponiendo definidas $h(\varphi)$ y $h(\psi)$, definir a partir de ellas a $h(\neg\varphi),\ h(\varphi \vee \psi),\ h(\varphi \wedge \psi),\ h(\varphi \to \psi),\ h(\varphi \leftrightarrow \psi),\ h(\forall x\varphi)$ y $h(\exists x\varphi)$.

Enunciamos ahora los principios de inducción.

Definición 1 (Principio de Inducción Estructural para $\mathsf{TERM}_{\mathcal{L}}$) *Sea $\mathcal{P}$ una propiedad acerca de términos. Para demostrar que $\mathcal{P}$ es válida para todos los términos, basta seguir los siguientes pasos:*

- *Base de la inducción: mostrar que*
  1. *Si $x \in \mathsf{Var}$ entonces $\mathcal{P}$ es válida para x.*
  2. *Si $c \in \mathcal{C}$ entonces $\mathcal{P}$ es válida para c.*
- *Hipótesis de inducción: suponer $\mathcal{P}$ para cualesquiera $t_1, \ldots, t_n \in \mathsf{TERM}_{\mathcal{L}}$.*
- *Paso inductivo: usando la Hipótesis de inducción mostrar que*
  1. *Si $f \in \mathcal{F}$ es un símbolo de función de índice n entonces $f(t_1, \ldots, t_n)$ cumple $\mathcal{P}$.*

Definición 2 (Principio de Inducción Estructural para $\mathsf{FORM}_{\mathcal{L}}$) *Sea $\mathcal{P}$ una propiedad acerca de fórmulas. Para probar que toda fórmula $\varphi \in \mathsf{FORM}_{\mathcal{L}}$ tiene la propiedad $\mathcal{P}$ basta seguir los siguientes pasos:*

- *Caso base: mostrar que toda fórmula atómica tiene la propiedad $\mathcal{P}$.*

- *Hipótesis de inducción: suponer que φ y ψ cumplen $\mathcal{P}$.*
- *Paso inductivo: mostrar usando la Hipótesis de inducción que*
  1. *$(\neg\varphi)$ también cumple $\mathcal{P}$.*
  2. *$(\varphi \star \psi)$ tiene la propiedad $\mathcal{P}$, donde $\star \in \{\rightarrow, \wedge, \vee, \leftrightarrow\}$*
  3. *$\forall x\varphi$ y $\exists x\varphi$ cumplen $\mathcal{P}$.*

5. Conceptos sintácticos importantes

Al tener cuantificadores en la lógica de primer orden se deben revisar con detenimiento algunos conceptos sintácticos como la importancia de un cuantificador en una fórmula así como los diferentes casos a estudiar al tratar de realizar una sustitución.

5.1. Ligado y alcance

En lógica de predicados, los cuantificadores $\forall x$ y $\exists x$ son ejemplos del fenómeno de *ligado* que también surge en los lenguajes de programación. La idea general es que ciertas presencias de variables son presencias *ligadas* o simplemente *ligas*, cada una de las cuales se asocia con una expresión llamada *alcance*.

El alcance de un cuantificador es la expresión sintáctica sobre la que dicho cuantificador ejerce control. En la fórmula $\forall xA$ el alcance del cuantificador $\forall x$ es la expresión que involucra a todas las presencias de la variable x en A y que no han sido afectadas previamente por otro cuantificador de la misma variable x. Es importante observar que el alcance de $\forall x$ en $\forall xA$ no necesariamente es la fórmula A, aunque es común encontrar dicha definición en la literatura. Sin embargo en el caso general el alcance ni siquiera es una fórmula, aunque siempre es fácil transformar la fórmula $\forall xA$ en una fórmula de la misma forma sintáctica, digamos $\forall xA'$ de modo que la fórmula A' sí sea el alcance de $\forall x$, lo cual se suele obviar en lógica matemática aunque es algo muy relevante de detectar y hacer explícito en lógica computacional, sobre todo con respecto a implementaciones. Las mismas observaciones aplican para el cuantificador $\exists x$.

Definición 3 *El alcance del cuantificador $\forall x$ (o $\exists xA$) en la fórmula $\forall xA$ (o $\exists xA$) se define como la expresión que consiste en eliminar de A cualquier subfórmula de la forma $\forall xB$ o $\exists xB$. Es decir, eliminamos de A cualquier otra cuantificación de la misma variable.*

Veamos unos ejemplos para aclarar la noción de alcance.

Ejemplo 5.1 *Alcance*

- *En la fórmula $\forall x(Px \rightarrow Rxy)$ el alcance de $\forall x$ es la fórmula $Px \rightarrow Rxy$.*
- *En la fórmula $\forall x(Qxy \wedge \neg Rzx) \vee Txax$ el alcance de $\forall x$ es la fórmula $Qxy \wedge \neg Rzx$*
- *en la fórmula $\forall y(Ruy \wedge \neg\exists zPz) \vee Ryz$ el alcance de $\forall y$ es $Ruy \wedge \neg\exists zPz$ y el alcance de $\exists z$ es Pz.*
- *En la fórmula $\forall x(Rzx \rightarrow \forall x(Px \wedge Qxx) \vee Tx)$ tenemos dos cuantificadores $\forall x$. El alcance del segundo (la fórmula por supuesto se lee de izquierda derecha) es la fórmula $Px \wedge Qxx$. Sin embargo el alcance del primero es la expresión $Rzx \rightarrow \bullet \vee Tx$ que NO es una fórmula. Hemos colocado un símbolo $\bullet$ indicando que esa parte de la fórmula original, a saber $\forall x(Px \wedge Qxx)$ no forma parte del alcance del primer $\forall x$.*

Una manera práctica de obtener el alcance de un cuantificador es mediante el árbol de sintaxis abstracta de una fórmula: El alcance de un cuantificador $\forall x$ o $\exists x$ es la expresión que se obtiene al eliminar del árbol cuya raíz es el cuantificador en cuestión todos los subárboles cuya raiz es otro cuantificador de la misma variable, es decir, otro $\forall x$ o $\exists x$. Nótese que si no hay otro cuantificador de la misma variable entonces el alcance de $\forall x A$ o $\exists x A$ sí es la fórmula A.

Definición 4 *Una presencia de la variable x en la fórmula A está ligada o acotada si es la variable de un cuantificador de A o si figura en el alcance de un cuantificador $\forall x$ o $\exists x$ de A.*
Si una presencia de la variable x en la fórmula φ no está ligada, decimos que está libre en A.

Ejemplo 5.2 Sea $A =_{def} \forall x \exists z (Q(y,z) \vee R(z,x,y)) \wedge P(z,x)$.

- El alcance del cuantificador $\forall x$ es la fórmula $\exists z (Q(y,z) \vee R(z,x,y))$.
- El alcance del cuantificador $\exists z$ es la fórmula $Q(y,z) \vee R(z,x,y)$.
- En A hay tres presencias de x, las dos primeras ligadas y la última libre.
- Las presencias de z son cuatro, ligadas las tres primeras y libre la última.
- Finalmente las dos presencias de y son libres.

Una manera práctica de decidir si una presencia de una variable dada, digamos x, es ligada es recorriendo el árbol de sintaxis abstracta desde dicha x hacia la raiz y parando al encontrar un cuantificador de la misma variable x en cuyo caso la presencia de x es ligada y el cuantificador encontrado es quien la liga. Si se llega a la raiz sin encontrar un cuantificador que ligue a x entonces x es libre.

Definición 5 *Sea A una fórmula. El conjunto de variables libres de A, se denota $FV(A)$.*

$$FV(A) = \{x \in \mathsf{Var} \mid x \text{ figura libre en } A\}$$

La notación $A(x_1, \ldots, x_n)$ quiere decir que $\{x_1, \ldots, x_n\} \subseteq FV(A)$.

Obsérvese que una misma variable x puede figurar tanto libre como ligada en una fórmula.

Definición 6 *Una fórmula A es* cerrada *si no tiene variables libres, es decir, si $FV(A) = \varnothing$. Una fórmula cerrada también se conoce como enunciado o sentencia.*

Definición 7 *Sea $A(x_1, \ldots, x_n)$ una fórmula con $FV(A) = \{x_1, \ldots, x_n\}$. La cerradura universal de A, denotada $\forall A$, es la fórmula $\forall x_1 \ldots \forall x_n\, A$. La cerradura existencial de A, denotada $\exists A$ es la fórmula $\exists x_1 \ldots \exists x_n\, A$.*

Obsérvese que una cerradura se obtiene cuantificando, universal o existencialmente, todas las variables libres de una fórmula.

6. Sustitución en fórmulas

La noción de sustitución en la lógica de predicados es más complicada que en la lógica proposicional debido a la existencia de términos y de variables ligadas. A diferencia de lo que sucede en lógica proposicional, donde una sustitución es sólo una operación textual sobre las fórmulas, en la lógica de predicados las sustituciones son operaciones sobre los términos y sobre las fórmulas y en este último caso deben respetar el ligado de variables, por lo tanto **no** son operaciones textuales.

La aplicación de sustituciones a fórmulas, necesita de ciertos cuidados debido a la presencia de variables ligadas mediante cuantificadores. La aplicación de una sustitución textual a una fórmula puede llevar a situaciones problemáticas con respecto a la sintaxis y a la semántica de la lógica. En particular, las sustituciones en fórmulas deben respetar el significado de la fórmula original y deben evitarse los siguientes problemas:

Sustitución de variables ligadas La definición de sustitución textual

$$(\forall y A)[\vec{x} := \vec{t}] =_{def} \forall (y[\vec{x} := \vec{t}])(A[\vec{x} := \vec{t}])$$

obliga a sustituir todas las variables de una fórmula, pero tal definición genera expresiones que ni siquiera son fórmulas, por ejemplo tendríamos que

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

La expresión de la derecha no es una fórmula bien formada puesto que la cuantificación sobre términos que no sean variables no está permitida. Lo mismo sucederá al definir la sustitución de esta manera para una fórmula existencial. En conclusión, la sustitución en fórmulas **NO** debe ser una sustitución textual puesto que las presencias ligadas de variables no pueden sustituirse.

Una solución inmediata al problema anterior consiste en definir la sustitución sólamente sobre variables libres, como sigue

$$(\forall x A)[\vec{x} := \vec{t}] =_{def} \forall x (A[\vec{x} := \vec{t}\,]_x)$$

donde $[\vec{x} := \vec{t}\,]_x$ denota a la sustitución que elimina al par $x := t$ de $[\vec{x} := \vec{t}\,]$.

Por ejemplo $[z, y, x := y, f(x), c]_x = [z, y := y, f(x)]$.

Así la aplicación de sustitución para la fórmula del ejemplo anterior se ejecuta como sigue:

$$\begin{array}{rcl} (\forall x P(y, f(x)))[x, y := g(y), z] & = & \forall x (P(y, f(x)[x, y := g(y), z]_x) \\ & = & \forall x (P(y, f(x))[y := z]) \\ & = & \forall x P(z, f(x)) \end{array}$$

De esta manera se garantiza que el resultado de aplicar una sustitución a una cuantificación siempre será una fórmula. Análogamente podemos definir la sustitución para una fórmula existencial.
Pasemos ahora a ver el segundo problema a evitar.

Captura de variables libres Si bien la solución dada al problema anterior es suficiente desde el punto de vista sintáctico, no lo es al entrar en juego la semántica o el significado de las fórmulas.

Considérese la fórmula $\forall x A$, la semántica intuitiva nos dice que esta fórmula será cierta siempre y cuando $A(x)$ sea cierta para cualquier valor posible de x, de manera que nos gustaría poder obtener, a partir de la de verdad $\forall x A$, la verdad de cualquier aplicación de sustitución $A[x := t]$.
Veamos ahora qué sucede si consideramos la fórmula

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

Está fórmula expresa el hecho de que para cualquier individuo existe otro distinto de él y es cierta si el universo tiene al menos dos elementos.
Ahora bien, al momento de calcular el caso particular $(\exists y (x \neq y))[x := y]$ con la definición anterior de aplicación de sustituciones obtenemos:

$$(\exists y\, (x \neq y))[x := y] \;=\; \exists y (\, (x \neq y)[x := y]_y) \;=\; \exists y (\, (x \neq y)[x := y]) \;=\; \exists y\, (y \neq y)$$

Pero en esta última fórmula se ha cambiado el sentido: se está diciendo que hay un objeto y que es distinto de si mismo lo cual es absurdo.

De manera que con la definición dada no podemos garantizar la verdad de un caso particular de A aún suponiendo la verdad de la cuantificación universal $\forall x A$.

¿ Qué es lo que anda mal? obsérvese que la presencia libre de x en la fórmula $\exists y(x \neq y)$ se ligó al aplicar la sustitución $[x := y]$. Así que una variable libre que representa a un objeto particular se sustituyó por una variable que representa al objeto cuya existencia se está asegurando. Esta clase de sustituciones son inadmisibles ya que cambian el significado de la fórmula: las posiciones que corresponden a una presencia libre de una variable representan a objetos particulares y el permitir que se liguen después de una sustitución modificará el significado intensional de la fórmula.

Existen dos maneras de solucionar el problema, la primera es aceptando la definición dada arriba mediante el uso de sustituciones de la forma $[\vec{x} := \vec{t}]_x$ pero prohibiendo la aplicación de sustituciones que liguen posiciones libres de variables. Para esto debe darse una una definición sustitución admisible para una fórmula, este método es el más usado en textos de lógica matemática.

Nosotros preferimos el segundo método, utilizado generalmente en la teoría de lenguajes de programación: *la aplicación de una sustitución a una fórmula se define renombrando variables ligadas de manera que siempre podremos obtener una sustitución admisible.*

Después de analizar los problemas anteriores, podemos dar una definición correcta para la sustitución:

Definición 8 *La aplicación de una sustitución $[\vec{x} := \vec{t}]$ a una fórmula A, denotada $A[\vec{x} := \vec{t}]$ se define como la fórmula obtenida al reemplazar simultáneamente todas las presencias libres de x_i en A por t_i, verificando que este proceso no capture posiciones de variables libres.*

La aplicación de una sustitución a una fórmula $A[\vec{x} := \vec{t}]$ se define **recursivamente** como sigue

$$\begin{array}{rcl}
\bot[\vec{x} := \vec{t}] & = & \bot \\
\top[\vec{x} := \vec{t}] & = & \top \\
P(t_1, \ldots, t_m)[\vec{x} := \vec{t}] & = & P(t_1[\vec{x} := \vec{t}], \ldots, t_m[\vec{x} := \vec{t}]) \\
(t_1 = t_2)[\vec{x} := \vec{t}] & = & t_1[\vec{x} := \vec{t}] = t_2[\vec{x} := \vec{t}] \\
 & & \\
(\neg A)[\vec{x} := \vec{t}] & = & \neg(A[\vec{x} := \vec{t}]) \\
(A \wedge \psi)[\vec{x} := \vec{t}] & = & (A[\vec{x} := \vec{t}] \wedge \psi[\vec{x} := \vec{t}]) \\
(A \vee \psi)[\vec{x} := \vec{t}] & = & (A[\vec{x} := \vec{t}] \vee \psi[\vec{x} := \vec{t}]) \\
(A \to \psi)[\vec{x} := \vec{t}] & = & (A[\vec{x} := \vec{t}] \to \psi[\vec{x} := \vec{t}]) \\
(A \leftrightarrow \psi)[\vec{x} := \vec{t}] & = & (A[\vec{x} := \vec{t}] \leftrightarrow \psi[\vec{x} := \vec{t}]) \\
 & & \\
(\forall y A)[\vec{x} := \vec{t}] & = & \forall y(A[\vec{x} := \vec{t}]) \ \ \textbf{si } y \notin \vec{x} \cup Var(\vec{t}) \\
(\exists y A)[\vec{x} := \vec{t}] & = & \exists y(A[\vec{x} := \vec{t}]) \ \ \textbf{si } y \notin \vec{x} \cup Var(\vec{t})
\end{array}$$

Mencionamos ahora algunas propiedades de la sustitución que pueden verificarse mediante inducción sobre las fórmulas.

Proposición 1 *La operación de sustitución tiene las siguientes propiedades, considerando $x \notin \vec{x}$.*

- *Si $x \notin FV(A)$ entonces $A[\vec{x}, x := \vec{t}, r\,] = A[\vec{x} := \vec{t}]$.*
- *Si $\vec{x} \cap FV(A) = \varnothing$ entonces $A[\vec{x} := \vec{t}] = A$.*
- *$FV(A[\vec{x} := \vec{t}]) \subseteq (FV(A) - \vec{x}) \cup Var(\vec{t})$.*

- *Si* $x \notin \vec{x} \cup Var(\vec{t})$ *entonces* $x \in FV(A)$ *si y sólo si* $x \in FV(A[\vec{x} := \vec{t}])$.
- *Si* $x \notin \vec{x} \cup Var(\vec{t})$ *entonces* $A[x := t][\vec{x} := \vec{t}] = A[\vec{x} := \vec{t}]\,[x := t[\vec{x} := \vec{t}]]$.
- *Si* $x \notin \vec{x} \cup Var(\vec{t})$ *entonces* $A[\vec{x}, x := \vec{t}, t] = A[\vec{x} := \vec{t}][x := t]$.

6.1. La relación de α-equivalencia

La definición de sustitución en fórmulas cuenta con una restricción aparente en el caso de los cuantificadores, por ejemplo, la sustitución

$$\forall x(Q(x) \to R(z, x))[z := f(x)]$$

no está definida, puesto que x figura en $f(x)$ es decir $x \in Var(f(x))$, con lo que no se cumple la condición necesaria para aplicar la sustitución, a saber que las variables en la sustitución son ajenas a las de la fórmula. Por lo tanto, la aplicación de cualquier sustitución a una fórmula hasta ahora es una *función parcial*.

Esta aparente restricción desaparece al notar que los nombres de las variables ligadas no importan: por ejemplo, las fórmulas $\forall x P(x)$ y $\forall y P(y)$ significan exactamente lo mismo, a saber que todos los elementos del universo dado cumplen la propiedad P.

Por lo tanto convenimos en identificar fórmulas que sólo difieren en sus variables ligadas, esto se hace formalmente mediante la llamada relación de α-equivalencia definida como sigue:

Definición 9 *Decimos que dos fórmulas* A_1, A_2 *son* α*-equivalentes lo cual escribimos* $A_1 \sim_\alpha A_2$ *si y sólo si* A_1 *y* A_2 *difieren a lo más en los nombres de sus variables ligadas.*

Ejemplo 6.1 Las siguientes expresiones son α-equivalentes.

$$\forall x P(x, y) \to \exists y R(x, y, z) \quad \sim_\alpha \quad \forall w P(w, y) \to \exists v R(x, v, z) \quad \sim_\alpha \quad \forall z P(z, y) \to \exists u R(x, u, z)$$

Más tarde demostraremos que las fórmulas α-equivalentes también son lógicamente equivalentes y por lo tanto son intercambiables en cualquier contexto o bajo cualquier operación.

Usando la α-equivalencia, la operación de sustitución en fórmulas se vuelve una función *total* por lo que siempre está definida.

Ejemplo 6.2 Por ejemplo se tiene que

$$\forall x(Q(x) \to R(z, x))[z := f(x)] \;\sim_\alpha\; \forall y(Q(y) \to R(z, y))[z := f(x)] \;=\; \forall y(Q(y) \to R(f(x), y))$$

Veamos otros ejemplos de sustituciones:

Ejemplo 6.3 Las siguientes sustituciones se sirven, en caso necesario, de la α-equivalencia

$$\begin{aligned}
\forall x R(x,y,f(z))[y,z := d, g(w)] &= \forall x R(x,d,f(g(w))) \\
\forall x P(x,y)[y,z := f(c), w] &= \forall x P(x,f(c)) \\
Q(y,z,y)[y,z := g(d), f(y)] &= Q(g(d),f(y),g(d)) \\
\exists w Q(y,z,y)[y,z := g(d), f(y)] &= \exists w Q(g(d),f(y),g(d)) \\
\exists y Q(y,z,y)[y,z := g(d), f(y)] &= \exists w Q(w,f(y),w) \\
\exists z Q(y,z,y)[y,z := g(d), f(y)] &= \exists w Q(g(d),w,g(d)) \\
\forall x\big(Q(z,y,x) \wedge \exists z T(f(z),w,y)\big)[x,y,z := a,z,g(w)] &= \forall u\big(Q(g(w),z,u) \wedge \exists v T(f(v),w,z)\big) \\
\forall x\big(Q(z,y,x) \wedge \exists z T(f(z),w,y)\big)[z,w,y := g(x),h(z),w] &= \forall u\big(Q(g(x),w,u) \wedge \exists v T(f(v),h(z),w)\big)
\end{aligned}$$