Lógica computacional

Tema: Lógica de primer orden: Introducción y sintaxis.

Pilar Selene Linares Arévalo

Facultad de Ciencias
Universidad Nacional Autónoma de México

marzo 2018

Material desarrollado bajo el proyecto UNAM-PAPIME PE102117.

Sintaxis

$x_1, x_2, \ldots, x_n$

$c_1, c_2, \ldots, c_n$

$f(t_1, t_2, \ldots, t_n)$

TÉRMINOS

$$\boxed{\begin{array}{c} \top, \bot \\ P(t_1, \ldots, t_n) \end{array}}$$

$\neg\varphi\,,\ \varphi \wedge \psi\,,\ \varphi \vee \psi$

$\varphi \rightarrow \psi\,,\ \varphi \leftrightarrow \psi$

$\exists x \varphi\,,\ \forall y \varphi$

FÓRMULAS

Sintaxis

Términos

Los **términos** del lenguaje son aquellas expresiones que representan objetos, elementos o individuos en el universo del discurso y se generan con la siguiente gramática:

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

Al conjunto de términos en lenguaje de la Lógica de Primer Oren lo

denotaremos con TERM.

Fórmulas

El conjunto de expresiones atómicas se denotará con ATOM y está formado por:

Sintaxis

Fórmulas

El conjunto de expresiones atómicas se denotará con ATOM y está formado por:

- Las constantes lógicas $\bot, \top$.

Sintaxis

Fórmulas

El conjunto de expresiones atómicas se denotará con ATOM y está formado 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.

Fórmulas

El conjunto de expresiones atómicas se denotará con ATOM y está formado 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.

Fórmulas

El conjunto de expresiones atómicas se denotará con ATOM y está formado 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.

Fórmulas

El conjunto de expresiones atómicas se denotará con ATOM y está formado 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.

Los términos se construyen bajo la siguiente gramática:

$$ATOM \quad ::= \quad \bot \mid \top \mid P(t_1, \ldots, t_m) \mid t_1 = t_2$$

Sintaxis

Fórmulas

El conjunto FORM de fórmulas compuestas, llamadas usualmente *fórmulas*, se define recursivamente como sigue:

Sintaxis

Fórmulas

El conjunto FORM de fórmulas compuestas, llamadas usualmente *fórmulas*, se define recursivamente como sigue:

- Si $\varphi \in$ ATOM entonces $\varphi \in$ FORM. Es decir, toda fórmula atómica es una fórmula.

Fórmulas

El conjunto FORM de fórmulas compuestas, llamadas usualmente *fórmulas*, se define recursivamente como sigue:

- Si $\varphi \in$ ATOM entonces $\varphi \in$ FORM. Es decir, toda fórmula atómica es una fórmula.
- Si $\varphi \in$ FORM entonces $(\neg\varphi) \in$ FORM.

Sintaxis

Fórmulas

El conjunto FORM de fórmulas compuestas, llamadas usualmente *fórmulas*, se define recursivamente como sigue:

- Si $\varphi \in$ ATOM entonces $\varphi \in$ FORM. Es decir, toda fórmula atómica es una fórmula.
- Si $\varphi \in$ FORM entonces $(\neg\varphi) \in$ FORM.
- Si $\varphi, \psi \in$ FORM entonces $(\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \to \psi), (\varphi \leftrightarrow \psi) \in$ FORM.

Sintaxis

Fórmulas

El conjunto FORM de fórmulas compuestas, llamadas usualmente *fórmulas*, se define recursivamente como sigue:

- Si $\varphi \in$ ATOM entonces $\varphi \in$ FORM. Es decir, toda fórmula atómica es una fórmula.
- Si $\varphi \in$ FORM entonces $(\neg\varphi) \in$ FORM.
- Si $\varphi, \psi \in$ FORM entonces $(\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \to \psi), (\varphi \leftrightarrow \psi) \in$ FORM.
- Si $\varphi \in$ FORM y $x \in$ Var entonces $(\forall x\varphi), (\exists x\varphi) \in$ FORM.

Fórmulas

El conjunto FORM de fórmulas compuestas, llamadas usualmente *fórmulas*, se define recursivamente como sigue:

- Si $\varphi \in$ ATOM entonces $\varphi \in$ FORM. Es decir, toda fórmula atómica es una fórmula.
- Si $\varphi \in$ FORM entonces $(\neg\varphi) \in$ FORM.
- Si $\varphi, \psi \in$ FORM entonces $(\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \to \psi), (\varphi \leftrightarrow \psi) \in$ FORM.
- Si $\varphi \in$ FORM y $x \in$ Var entonces $(\forall x\varphi), (\exists x\varphi) \in$ FORM.

Fórmulas

El conjunto FORM de fórmulas compuestas, llamadas usualmente *fórmulas*, se define recursivamente como sigue:

- Si $\varphi \in$ ATOM entonces $\varphi \in$ FORM. Es decir, toda fórmula atómica es una fórmula.
- Si $\varphi \in$ FORM entonces $(\neg\varphi) \in$ FORM.
- Si $\varphi, \psi \in$ FORM entonces $(\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \to \psi), (\varphi \leftrightarrow \psi) \in$ FORM.
- Si $\varphi \in$ FORM y $x \in$ Var entonces $(\forall x\varphi), (\exists x\varphi) \in$ FORM.

La gramática correspondiente es:

$$
\begin{array}{rcl}
F & ::= & ATOM \mid (\neg F) \mid (F \star F) \mid (\forall x\, F) \mid (\exists x\, F) \\
\star & ::= & \wedge \mid \vee \mid \to \mid \leftrightarrow
\end{array}
$$

Sintaxis

Convención: Los cuantificadores se aplican a la *mínima* expresión sintácticamente posible delante del cuantificador.

Sintaxis

Convención: Los cuantificadores se aplican a la *mínima* expresión sintácticamente posible delante del cuantificador.

De manera que

$$\forall x\varphi \to \psi \quad \text{es} \quad (\forall x\varphi) \to \psi$$

Convención: Los cuantificadores se aplican a la *mínima* expresión sintácticamente posible delante del cuantificador.

De manera que

$$
\begin{array}{rcl}
\forall x\varphi \to \psi & \text{es} & (\forall x\varphi) \to \psi \\
\exists y\varphi \land \forall w\psi \to \chi & \text{es} & (\exists y\varphi) \land (\forall w\psi) \to \chi
\end{array}
$$

Definición recursiva de funciones sobre términos

Para definir una función $h : \text{TERM} \to A$, basta definir h como sigue:

Definición recursiva de funciones sobre términos

Para definir una función $h : \mathrm{TERM} \to A$, basta definir h como sigue:

- Definir $h(x)$ para $x \in \mathrm{Var}$.

Definición recursiva de funciones sobre términos

Para definir una función $h : \mathrm{TERM} \to A$, basta definir h como sigue:

- Definir $h(x)$ para $x \in \mathrm{Var}$.
- Definir $h(c)$ para cada constante $c \in \mathcal{C}$.

Definición recursiva de funciones sobre términos

Para definir una función $h : \text{TERM} \to A$, basta definir h como sigue:

- Definir $h(x)$ para $x \in \text{Var}$.
- 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.

Definición recursiva de funciones sobre fórmulas

Para definir una función $h : \text{FORM} \to A$, basta definir h como sigue:

Definición recursiva de funciones sobre fórmulas

Para definir una función $h : \text{FORM} \to A$, 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.

Definición recursiva de funciones sobre fórmulas

Para definir una función $h : \mathrm{FORM} \to A$, 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 $h(\exists x\varphi)$.

Principio de inducción estructural para términos

Sea $\mathcal{P}$ una propiedad acerca de términos. Para demostrar que $\mathcal{P}$ es válida para todos los términos, basta seguir los siguientes pasos:

Principio de inducción estructural para términos

Sea $\mathcal{P}$ una propiedad acerca de términos. Para demostrar que $\mathcal{P}$ es válida para todos los términos, basta seguir los siguientes pasos:

- Caso base: mostrar que

Principio de inducción estructural para términos

Sea $\mathcal{P}$ una propiedad acerca de términos. Para demostrar que $\mathcal{P}$ es válida para todos los términos, basta seguir los siguientes pasos:

- Caso base: mostrar que
  - $\mathcal{P}$ es válida para x, con $x \in \mathrm{Var}$.

Principio de inducción estructural para términos

Sea $\mathcal{P}$ una propiedad acerca de términos. Para demostrar que $\mathcal{P}$ es válida para todos los términos, basta seguir los siguientes pasos:

- Caso base: mostrar que
  - $\mathcal{P}$ es válida para x, con $x \in \mathrm{Var}$.
  - $\mathcal{P}$ es válida para c, con $c \in \mathcal{C}$.

Principio de inducción estructural para términos

Sea $\mathcal{P}$ una propiedad acerca de términos. Para demostrar que $\mathcal{P}$ es válida para todos los términos, basta seguir los siguientes pasos:

- Caso base: mostrar que
  - $\mathcal{P}$ es válida para x, con $x \in \text{Var}$.
  - $\mathcal{P}$ es válida para c, con $c \in \mathcal{C}$.

- Hipótesis de inducción: suponer $\mathcal{P}$ para cualesquiera $t_1, \ldots, t_n \in \text{TERM}$.

Principio de inducción estructural para términos

Sea $\mathcal{P}$ una propiedad acerca de términos. Para demostrar que $\mathcal{P}$ es válida para todos los términos, basta seguir los siguientes pasos:

- Caso base: mostrar que
  - $\mathcal{P}$ es válida para x, con $x \in \text{Var}$.
  - $\mathcal{P}$ es válida para c, con $c \in \mathcal{C}$.
- Hipótesis de inducción: suponer $\mathcal{P}$ para cualesquiera $t_1, \ldots, t_n \in \text{TERM}$.
- Paso inductivo: usando la Hipótesis de inducción mostrar que

Principio de inducción estructural para términos

Sea $\mathcal{P}$ una propiedad acerca de términos. Para demostrar que $\mathcal{P}$ es válida para todos los términos, basta seguir los siguientes pasos:

- Caso base: mostrar que
  - $\mathcal{P}$ es válida para x, con $x \in \text{Var}$.
  - $\mathcal{P}$ es válida para c, con $c \in \mathcal{C}$.
- Hipótesis de inducción: suponer $\mathcal{P}$ para cualesquiera $t_1, \ldots, t_n \in \text{TERM}$.
- Paso inductivo: usando la Hipótesis de inducción mostrar que
  - $f(t_1, \ldots, t_n)$ cumple $\mathcal{P}$, donde $f \in \mathcal{F}$ es un símbolo de función de índice n.

Principio de inducción estructural para fórmulas

Sea $\mathcal{P}$ una propiedad acerca de fórmulas. Para probar que toda fórmula $\varphi \in \mathrm{FORM}$ tiene la propiedad $\mathcal{P}$ basta seguir los siguientes pasos:

Principio de inducción estructural para fórmulas

Sea $\mathcal{P}$ una propiedad acerca de fórmulas. Para probar que toda fórmula $\varphi \in \mathrm{FORM}$ tiene la propiedad $\mathcal{P}$ basta seguir los siguientes pasos:

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

Principio de inducción estructural para fórmulas

Sea $\mathcal{P}$ una propiedad acerca de fórmulas. Para probar que toda fórmula $\varphi \in \text{FORM}$ 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 $\mathcal{P}$.

Principio de inducción estructural para fórmulas

Sea $\mathcal{P}$ una propiedad acerca de fórmulas. Para probar que toda fórmula $\varphi \in$ FORM 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

Principio de inducción estructural para fórmulas

Sea $\mathcal{P}$ una propiedad acerca de fórmulas. Para probar que toda fórmula $\varphi \in$ FORM 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 $\mathcal{P}$.

Inducción y recursión

Principio de inducción estructural para fórmulas

Sea $\mathcal{P}$ una propiedad acerca de fórmulas. Para probar que toda fórmula $\varphi \in$ FORM 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 $\star \in \{\to, \wedge, \vee, \leftrightarrow\}$

Principio de inducción estructural para fórmulas

Sea $\mathcal{P}$ una propiedad acerca de fórmulas. Para probar que toda fórmula $\varphi \in$ FORM 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 $\star \in \{\to, \land, \lor, \leftrightarrow\}$
  3. $\forall x\varphi$ y $\exists x\varphi$ cumplen $\mathcal{P}$.

Ligado y alcance

Ligado y alcance

Dada una cuantificación $\forall x \varphi$ o $\exists x \varphi$, la presencia de x en $\forall x$ o $\exists x$ es la variable que *liga* el cuantificador correspondiente; mientras que la fórmula φ es el **alcance**, ámbito o radio del cuantificador.

Ligado y alcance

Dada una cuantificación $\forall x \varphi$ o $\exists x \varphi$, la presencia de x en $\forall x$ o $\exists x$ es la variable que *liga* el cuantificador correspondiente; mientras que la fórmula φ es el **alcance**, ámbito o radio del cuantificador.

Ligado y alcance

Dada una cuantificación $\forall x \varphi$ o $\exists x \varphi$, la presencia de x en $\forall x$ o $\exists x$ es la variable que *liga* el cuantificador correspondiente; mientras que la fórmula φ es el **alcance**, ámbito o radio del cuantificador.

Una presencia de la variable x en la fórmula φ está **ligada** si figura en el alcance de un cuantificador y éste es el más cercano a x.

Si una presencia de la variable x no es ligada, decimos que es **libre**.

Sustitución

Sustitución sobre términos

La aplicación de una sustitución $[\vec{x} := \vec{t}]$ a un término r, denotada $r[\vec{x} := t]$, se define como el término obtenido al reemplazar **simultáneamente** todas las presencias de x_i en r por t_i. Este proceso de define recursivamente como sigue:

Sustitución sobre términos

La aplicación de una sustitución $[\vec{x} := \vec{t}]$ a un término r, denotada $r[\vec{x} := t]$, se define como el término obtenido al reemplazar **simultáneamente** todas las presencias de x_i en r por t_i. Este proceso de define recursivamente como sigue:

Sustitución sobre términos

La aplicación de una sustitución $[\vec{x} := \vec{t}]$ a un término r, denotada $r[\vec{x} := t]$, se define como el término obtenido al reemplazar **simultáneamente** todas las presencias de x_i en r por t_i. Este proceso de define recursivamente como sigue:

$$x_i[\vec{x} := \vec{t}] \quad = \quad t_i \qquad 1 \leqslant i \leqslant n$$

Sustitución

Sustitución sobre términos

La aplicación de una sustitución $[\vec{x} := \vec{t}]$ a un término r, denotada $r[\vec{x} := t]$, se define como el término obtenido al reemplazar **simultáneamente** todas las presencias de x_i en r por t_i. Este proceso de define recursivamente como sigue:

$$
\begin{array}{rcll}
x_i[\vec{x} := \vec{t}\,] & = & t_i & 1 \leqslant i \leqslant n \\
z[\vec{x} := \vec{t}\,] & = & z & \text{si } z \neq x_i \; 1 \leqslant i \leqslant n
\end{array}
$$

Sustitución

Sustitución sobre términos

La aplicación de una sustitución $[\vec{x} := \vec{t}]$ a un término r, denotada $r[\vec{x} := t]$, se define como el término obtenido al reemplazar **simultáneamente** todas las presencias de x_i en r por t_i. Este proceso de define recursivamente como sigue:

$$
\begin{array}{rcll}
x_i[\vec{x} := \vec{t}\,] & = & t_i & 1 \leqslant i \leqslant n \\
z[\vec{x} := \vec{t}\,] & = & z & \text{si } z \neq x_i \; 1 \leqslant i \leqslant n \\
c[\vec{x} := \vec{t}\,] & = & c & \text{si } c \in \mathcal{C}, \text{ es decir, } c \text{ constante}
\end{array}
$$

Sustitución

Sustitución sobre términos

La aplicación de una sustitución $[\vec{x} := \vec{t}]$ a un término r, denotada $r[\vec{x} := t]$, se define como el término obtenido al reemplazar **simultáneamente** todas las presencias de x_i en r por t_i. Este proceso de define recursivamente como sigue:

$$
\begin{array}{rcll}
x_i[\vec{x} := \vec{t}\,] & = & t_i & 1 \leqslant i \leqslant n \\
z[\vec{x} := \vec{t}\,] & = & z & \text{si } z \neq x_i \; 1 \leqslant i \leqslant n \\
c[\vec{x} := \vec{t}\,] & = & c & \text{si } c \in \mathcal{C}\text{, es decir, } c \text{ constante} \\
f(t_1, \ldots, t_m)[\vec{x} := \vec{t}\,] & = & f(t_1[\vec{x} := \vec{t}\,], \ldots, t_m[\vec{x} := \vec{t}\,]) & \text{con } f^{(m)} \in \mathcal{F}.
\end{array}
$$

Sustitución

Debido a la presencia de variables libres y ligadas, la aplicación de una *sustitución textual* a una fórmula puede llevar a situaciones problemáticas, por ejemplo:

Sustitución

Debido a la presencia de variables libres y ligadas, la aplicación de una *sustitución textual* a una fórmula puede llevar a situaciones problemáticas, por ejemplo:

- Generar expresiones que no son fórmulas:

$$\big(\forall x P(y, fx)\big)[x, y := gy, z] = \forall gy P\big(z, fgy)\big)$$

  La expresión de la derecha no es una fórmula.

Sustitución

- Captura de variables: Consideremos la siguiente aplicación de sustitución

$$\forall x\big(\exists y(\,x \neq y)\big)[x := y] \quad ->$$

Sustitución

- Captura de variables: Consideremos la siguiente aplicación de sustitución

$$\forall x\big(\exists y(\,x \neq y)\big)[x := y] \quad ->$$

Sustitución

- Captura de variables: Consideremos la siguiente aplicación de sustitución

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

Sustitución

- Captura de variables: Consideremos la siguiente aplicación de sustitución

$$
\begin{aligned}
\forall x \big(\exists y(x \neq y)\big)[x := y] \quad & -> \\
& -> \quad \exists y\big((x \neq y)\big)[x := y] \\
& -> \quad (x \neq y)[x := y]
\end{aligned}
$$

Sustitución

- Captura de variables: Consideremos la siguiente aplicación de sustitución

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

Sustitución

- Captura de variables: Consideremos la siguiente aplicación de sustitución

$$
\begin{array}{lll}
\forall x\big(\exists y(\, x \neq y)\big)[x := y] & -> & \\
 & -> & \exists y\big(\,(x \neq y)\big)[x := y] \\
 & -> & (x \neq y)[x := y] \\
 & -> & y \neq y
\end{array}
$$

$$\forall x\big(\exists y(\, x \neq y)\big)[x := y] \quad = \quad \forall x\big(\exists y(\, y \neq y)\big)$$

Sustitución

- Captura de variables: Consideremos la siguiente aplicación de sustitución

$$
\begin{array}{lll}
\forall x\big(\exists y(\,x \neq y)\big)[x := y] & -> & \\
 & -> & \exists y\big(\,(x \neq y)\big)[x := y] \\
 & -> & (x \neq y)[x := y] \\
 & -> & y \neq y
\end{array}
$$

$$\forall x\big(\exists y(\,x \neq y)\big)[x := y] \quad = \quad \forall x\big(\exists y(\,y \neq y)\big)$$

Para solucionar lo anterior, vamos a preferir un método utilizado en 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.*

Sustitución sobre fórmulas

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

Sustitución sobre fórmulas

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

Sustitución sobre fórmulas

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

$$\begin{array}{rcl} \bot[\vec{x} := \vec{t}\,] & = & \bot \\ \top[\vec{x} := \vec{t}\,] & = & \top \end{array}$$

Sustitución sobre fórmulas

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

$$
\begin{array}{rcl}
\bot[\vec{x} := \vec{t}\,] & = & \bot \\
\top[\vec{x} := \vec{t}\,] & = & \top \\
P(t_1, \ldots, t_m)[\vec{x} := \vec{t}\,] & = & P\big(t_1[\vec{x} := \vec{t}\,], \ldots, t_m[\vec{x} := \vec{t}\,]\big)
\end{array}
$$

Sustitución

Sustitución sobre fórmulas

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

$$\begin{array}{rcl}
\bot[\vec{x} := \vec{t}\,] & = & \bot \\
\top[\vec{x} := \vec{t}\,] & = & \top \\
P(t_1, \ldots, t_m)[\vec{x} := \vec{t}\,] & = & P\big(t_1[\vec{x} := \vec{t}\,], \ldots, t_m[\vec{x} := \vec{t}\,]\big) \\
(t_1 = t_2)[\vec{x} := \vec{t}\,] & = & t_1[\vec{x} := \vec{t}\,] = t_2[\vec{x} := \vec{t}\,]
\end{array}$$

Sustitución

Sustitución sobre fórmulas

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

$$
\begin{array}{rcl}
\bot[\vec{x} := \vec{t}\,] & = & \bot \\
\top[\vec{x} := \vec{t}\,] & = & \top \\
P(t_1, \ldots, t_m)[\vec{x} := \vec{t}\,] & = & P\big(t_1[\vec{x} := \vec{t}\,], \ldots, t_m[\vec{x} := \vec{t}\,]\big) \\
(t_1 = t_2)[\vec{x} := \vec{t}\,] & = & t_1[\vec{x} := \vec{t}\,] = t_2[\vec{x} := \vec{t}\,] \\
\\
(\neg\varphi)[\vec{x} := \vec{t}\,] & = & \neg\big(\varphi[\vec{x} := \vec{t}\,]\big)
\end{array}
$$

Sustitución sobre fórmulas

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

$$
\begin{array}{rcl}
\bot[\vec{x} := \vec{t}\,] & = & \bot \\
\top[\vec{x} := \vec{t}\,] & = & \top \\
P(t_1, \ldots, t_m)[\vec{x} := \vec{t}\,] & = & P\big(t_1[\vec{x} := \vec{t}\,], \ldots, t_m[\vec{x} := \vec{t}\,]\big) \\
(t_1 = t_2)[\vec{x} := \vec{t}\,] & = & t_1[\vec{x} := \vec{t}\,] = t_2[\vec{x} := \vec{t}\,] \\
 & & \\
(\neg\varphi)[\vec{x} := \vec{t}\,] & = & \neg\big(\varphi[\vec{x} := \vec{t}\,]\big) \\
(\varphi \star \psi)[\vec{x} := \vec{t}\,] & = & \big(\varphi[\vec{x} := \vec{t}\,] \star \psi[\vec{x} := \vec{t}\,]\big)
\end{array}
$$

Sustitución sobre fórmulas

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

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

Sustitución

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

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

Sustitución

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.

Sustitución

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.

Esta 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.

Sustitución

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:

Sustitución

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:

Alfa Equivalencia

Decimos que dos fórmulas φ_1, φ_2 son α-equivalentes lo cual escribimos $\varphi_1 \sim_\alpha \varphi_2$ si y sólo si φ_1 y φ_2 difieren unicamente en los nombres de sus variables ligadas.

Sustitución

Las siguientes expresiones son α-equivalentes.

Sustitución

Las siguientes expresiones son α-equivalentes.

$\forall x P(x, y) \rightarrow \exists y R(x, y, z) \quad \sim_\alpha$

Sustitución

Las siguientes expresiones son α-equivalentes.

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

Sustitución

Las siguientes expresiones son α-equivalentes.

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

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