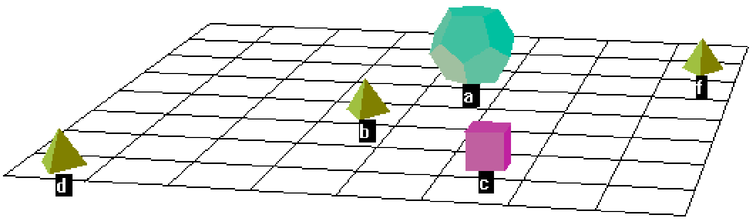
UNLu Teoría de la Computación II 2014 Parcial 2 6 - 05 - 2014 Duración 2 horas.

Legajo : ___ ___ ___ ___ _ N. y A. : ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___

EXAMEN A LIBRO CERRADO



EJERCICIO 1.

$\alpha = \exists y\, \exists z\, (y \neq z \wedge \forall w\, (\mathrm{BackOf}(b, w) \rightarrow (w = y \vee w = z)))$

Se requiere:

i) Definir el vocabulario del FOL correspondiente al TW de la figura.

ii) Definir la interpretación que contemple un modelo de cómputo del TW y donde el símbolo de constante *b* está interpretado por el objeto "b".

iii) Escribir el algoritmo que evalúe la formula α.

EJERCICIO 2.

i) Dada una estructura G = (P, E) escribir una fórmula de primer orden que exprese: En G existen maximales.

$\exists y\, (\forall z\, (E(y, z) \rightarrow y = z))$ Este enunciado debe se verdadero en estructuras con maximales.

ii) Dada una estructura G = (P, E) escribir una fórmula algebraica que exprese el conjunto de maximales de G, utilizando como función característica una fórmula de primer orden.

$\phi(\mathbf{y}) = \forall z\, (E(\mathbf{y}, z) \rightarrow \mathbf{y} = z)$

$\mathrm{Max}_G = \{y \in P : \phi(\mathbf{y})\}$

EJERCICIO 3.

Sea $\Gamma = \{\forall x\, R(x, x),\ \forall x \forall y\, ((R(x, y) \wedge R(y, x)) \rightarrow x = y),\ \forall x \forall y \forall z\, ((R(x, y) \wedge R(y, z)) \rightarrow R(x, z))\}$

Se requiere: Definir el vocabulario y una interpretación con universo finito de cardinalidad 8 o superior, que sea modelo de Γ. ¿Qué expresa Γ interpretado?

$\mathcal{L} = \langle R^2 \rangle$ Una Interpretación: $\mathcal{I} = (\mathcal{P}(\{1,2,3\}, \subseteq)$ Todo orden parcial es modelo de Γ

EJERCICIO 4.

i) Definir la semántica de primer orden. Dada en teoría

ii) Definir con precisión el operador *Libres*(ϕ) donde ϕ es una fórmula de primer orden y aplicar la definición al siguiente caso:

$\gamma = ((\exists x_1\, (\forall x_2\, (\forall x_3\, (P(x_1, x_2) \wedge P(f(x_2, c), x_4))))) \vee (\forall x_4\, P(x_3, x_4)))$

Podemos determinar las variables libres de una fórmula φ calculando la función $Libres(\varphi)$, la que se define recursivamente de la siguiente manera:

- Si φ es una fórmula atómica, entonces $Libres(\varphi)$ es el conjunto de variables que ocurren en φ.
- En cualquier otro caso:
  - Si φ es $\neg\psi$ entonces $Libres(\varphi) \stackrel{def}{=} Libres(\psi)$.
  - Si φ es $(\psi \vee \eta)$, o $(\psi \wedge \eta)$ ó $(\psi \rightarrow \eta)$ entonces $Libres(\varphi) \stackrel{def}{=} Libres(\psi) \cup Libres(\eta)$.
  - Si φ es $(\forall x\ \psi)$ o $(\exists x\ \psi)$ entonces $Libres(\varphi) \stackrel{def}{=} Libres(\psi) - \{x\}$.

Una variable x se dice *ligada* en $(\forall x\ \psi)$ o $(\exists x\ \psi)$.

$\gamma = ((\exists x_1\ (\forall x_2\ (\forall x_3\ (P(x_1, x_2) \wedge P(f(x_2, c), x_4))))) \vee (\forall x_4\ P(x_3, x_4)))$

$\gamma = (\gamma_1 \vee \gamma_2)$ luego Libres $(\gamma) =$ Libres $(\gamma_1) \bigcup$ Libres (γ_2)

Libres (γ_2) = Libres $(\forall x_4\ P(x_3, x_4)) \stackrel{def}{=}$ Libres$(P(x_3, x_4))$ -$\{x_4\}$ = $\{x_3,\ x_4\}$ - $\{x_4\}$ =$\{x_3\}$

Libres (γ_1) = Libres $((\exists x_1\ (\forall x_2\ (\forall x_3\ \gamma_3))))$ $\stackrel{def}{=}$ Libres$(\forall x_2\ (\forall x_3\ (\gamma_3)))$ - $\{x_1\}$ =

Libres $(\forall x_3\ (\gamma_3)))$ - $\{x_1\}$ - $\{x_2\}$ = Libres $(P(x_1,\ x_2) \wedge P(f(x_2, c), x_4))$ - $\{x_1\}$ - $\{x_2\}$ - $\{x_3\}$ =

Libres $(P(x_1,\ x_2)) \bigcup$ Libres$(P(f(x_2, c), x_4))$ - $\{x_1\}$ - $\{x_2\}$ - $\{x_3\}$ =

= $\{x_1,\ x_2\} \bigcup \{x_2, x_4\}$ - $\{x_1\}$ - $\{x_2\}$ - $\{x_3\}$ =

= $\{x_1,\ x_2, x_4\}$- $\{x_1\}$ - $\{x_2\}$ - $\{x_3\}$ = $\{x_4\}$

Luego Libres(γ) = $\{x_3, x_4\}$

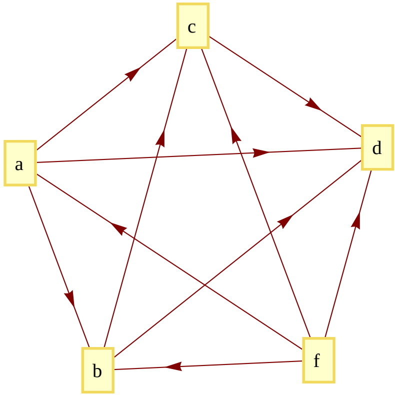
EJERCICIO 1. **(Correcciones en rojo)**

Nuestro objetivo es definir una representación y un cómputo para comprender (reproducir) la respuesta que da TW en el enunciado dado.

Un problema un poco más general es computar la fórmula con una var. libre,
$\theta(\underline{x})$ = $\exists$y $\exists$z ((y ≠ z) ∧ ∀w(BackOf($\underline{x}$, w) ⇒ (w = y ∨ w = z))) [1] **Corregido**
que permite obtener el enunciado al computar θ(b)

Como primer paso se define el lenguaje de P. O. con igualdad $\mathcal{L} = \langle B \rangle$ donde B es un símbolo de predicado binario.
La interpretación es: $\mathcal{I}$ = {P={"a", "b", "c", "d"}; BackOf}
BackOf = {{"f","a"},{"f","b"},{"f","c"},{"f","d"},{"a","b"},{"a","c"},{"a","d"},{"b","c"},{"b","d"},{"c","d"}};

BackOf



Para definir el cómputo de [1] sobre el modelo anterior primero se tratará la subfómula:
$\phi(x, y, z) = \forall w(\text{BackOf}(x, w) \Rightarrow (w = y \ \vee w = z))$ [2]

El cuantificador universal se tranforma en una conjunción sobre todos los valores de la variable ligada w en el universo de interpretación establecido (conjunto de nodos P)
Luego si P = $\{p_1, p_2, ..., p_n\}$ entonces,

$$\phi(x, y, z) = \left((\text{BackOf}(x, p_1) \Rightarrow (p_1 = y \vee p_1 = z)) \wedge ... \wedge (\text{BackOf}(x, p_n) \Rightarrow (p_n = y \vee p_n = z)) \right) =$$
$$= \bigwedge_{w \in P} (\text{BackOf}(x, w) \Rightarrow (w = y \vee w = z)) \quad [3]$$

Luego la fórmla [1] queda:
$\theta(x) = \exists y \ \exists z \ ((y \neq z) \wedge \phi(x, y, z))$ [4] **Corregido**

El cuantificador existencial se tranforma en una disyunción sobre los posibles valores de las variables ligadas: y, z. Luego se tiene,

$$\theta(x) = \bigvee_{y \in P} \bigvee_{z \in P} ((y \neq z) \wedge \phi(x, y, z)) \quad [5]$$

Aplicando la identidad [3] en [5] queda:

$$\theta(x) = \bigvee_{y \in P} \bigvee_{z \in P} ((y \neq z) \wedge \bigwedge_{w \in P} (\text{BackOf}(x, w) \Rightarrow (w = y \vee w = z)))$$

De esta manera queda definido el cómputo de la fórmula $\theta(x)$ tal que $\theta(b)$ es el enunciado en estudio.

```
Clear[P, BackOf, ϕ, θ];

P = {"a", "b", "c", "d", "f"};

BackOf = {{"f", "a"}, {"f", "b"}, {"f", "c"}, {"f", "d"}, {"a", "b"},
   {"a", "c"}, {"a", "d"}, {"b", "c"}, {"b", "d"}, {"c", "d"}};

ϕ[x_, y_, z_] := Module[{TF = True}, For[i = 1, i <= Length[P], i++, {TF =
      TF && Implies[MemberQ[BackOf, {x, P[[i]]}], (P[[i]] == y || P[[i]] == z)]}];
   TF];

θ[x_] :=
  Module[{TF2 = False}, For[y = 1, y <= Length[P], y++, {For[z = 1, z <= Length[P],
      z++, {TF2 = TF2 || (y ≠ z  ⋀ ϕ[x, P[[y]], P[[z]]]) }]}]; TF2];

(* Algunos enunciados y su valor de verdad ...*)

θ["b"]

True

θ["f"]

False

θ["a"]

False
```

Profesor.: A. H. Diolaiti. 2014