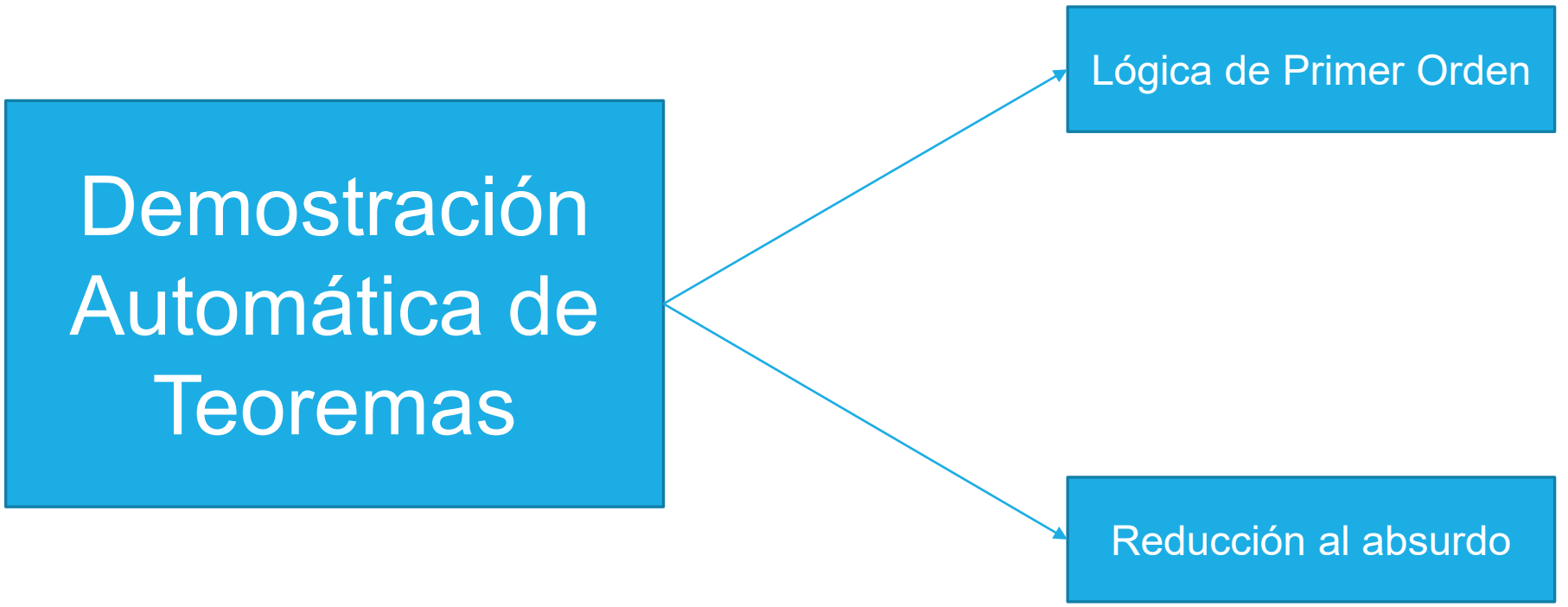
Programación Declarativa

Cláusulas Definidas

FACULTAD DE MATEMÁTICA Y COMPUTACIÓN

4TO AÑO DE COMPUTACIÓN

Surgimiento de la Programación Lógica



Lógica de predicados

ALFABETO

- Constantes
- Variables
- Conectores lógicos: $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$
- Signos de agrupación: [,], { , }, (,)
- Relaciones y funciones

Lógica de predicados

FÓRMULAS: secuencias de símbolos del lenguaje que se consideran sintácticamente correctas denominadas *fórmulas bien formadas (fbf).*

- Las constantes y las variables son *fbf*.
- Si $t_1, t_2, \dots, t_n$ son *fbf* entonces si f es una función o relación n-aria entonces $f(t_1, t_2, \dots, t_n)$ son fórmulas bien formadas.
- Si A y B son fbf entonces: $\neg A, \neg B, A \ / \ B, A \ / \ B, A <=> B, A => B$ son *fbf*.
- Si A es una *fbf* donde x ocurre libre entonces $\forall(X)A \ \ y \ \exists(X)A$ son *fbf*.

Una fórmula básica en una *fbf* donde no contiene variables

Lógica de Cláusulas Definidas

Programa Lógico: es un conjunto de cláusulas definidas.

Cláusulas Definidas: son un subconjunto de las *fbf* de la lógica de predicados

Fórmula elemental o átomos: las funciones o relaciones n-arias son átomos.

Literal: Si A es una fórmula elemental o átomo entonces A y ¬A son literales

Cláusula: Si A es una *fbf* de la forma:

$$\forall(X1)\forall(X2) \ldots \forall(Xn)\ (L1 \backslash/ L2 \ \backslash/ \cdots \ \backslash/ Ln)$$

son *fbf* donde todas las variables que ocurren en ella están cuantificadas universalmente.

Lógica de Cláusulas Definidas

De acuerdo con las leyes de la lógica proposicional cualquier cláusula puede ser representada como:

$$\forall(x_1) \ldots \forall(x_n)[A_1 \vee \ldots \vee A_k \vee \neg B_1 \vee \ldots \vee \neg B_m]$$

Con A_i literales positivos y B_j literales negativos

Expresándola como implicación

$$\forall(x_1) \ldots \forall (x_n)[A_1 \vee \ldots \vee A_k \Leftarrow B_1 \wedge \ldots \wedge B_m]$$

Dado que todas las variables que ocurren en una cláusula están cuantificadas universalmente, podemos hacer una representación implícita de esta cuantificación y eliminar la representación de los cuantificadores.

Lógica de Cláusulas Definidas

Cláusula Definida: Una cláusula se denomina cláusula definida si la misma contiene a lo sumo un literal positivo. Es decir, una cláusula definida es de la forma

$$A \Leftarrow B_1 \wedge \ldots \wedge B_m.$$

Regla: es una cláusula definida y denominaremos a A como cabeza de la regla y a $B_1 \wedge \ldots \wedge B_m$ cuerpo de la regla.

Objetivo: es una cláusula definida que no tiene cabeza.

Hecho: es una cláusula definida que no tiene cuerpo.

Programa Prolog

Es un conjunto de cláusulas definidas. Los hechos y las reglas de un programa que comienzan con el mismo nombre de predicado o relación constituyen la definición de dicho predicado en el programa.

Programa Prolog

1. $\boldsymbol{abuelo(X,Y) :- padre(X,Z), padre(z,y).}$

$$\forall(x)\,\forall(y)\,\forall(z)\,[abuelo(x,y) \Leftarrow padre(x,z) \wedge padre(z,y)].$$

$$\forall(x)\forall(y)\,\forall(z)\,(abuelo(x,y) \vee \neg padre(x,z) \vee \neg padre(z,y)).$$

2. concatena([x|v], y,[x|z]):- concatena(v, y, z).

$$\forall(x)\forall(v)\forall(y)\forall(z)[concatena([x|v],y,[x|z]) \Leftarrow concatena(v,y,z)]$$

$$\forall(x)\,\forall(v)\forall(y)\,\forall(z)\,(concatena([x|v],y,[x|z] \vee \neg concatena(v,y,z)).$$

Programa Prolog

- *Un programa* → una teoría de primer orden en la lógica de cláusulas definidas
- *Los hechos y las reglas del programa* → axiomas de la teoría.
- *Un objetivo* → teorema

Intérprete de Prolog será desde un punto de vista lógico un demostrador por refutación de teoremas

Programa Prolog

Se verifica que un objetivo es una cláusula definida de la forma:

$$\forall(x_1) \ldots \forall(x_n)[\neg B_1 \vee \ldots \vee \neg B_m] \cong \neg\exists(x_1) \ldots \exists(x_n)[B_1 \wedge \ldots \wedge B_m]$$

$:- concatena([1,2,3],[4,5],X).$

No Existe X tal que X sea el resultado de concatenar [1,2,3] y [4,5]

$:- concatena(X,[1,2,3],[1,2,3]).$

No Existe X, tal que al concatenarla con [1,2,3] se obtiene la lista [1,2,3]

Ejemplo Prolog

En los casos del Inspector Craig, siempre se plantean proposiciones que se saben que son verdaderas y a partir de ellas hay que descubrir al o a los culpables. En cierto caso, cuatro defendidos A, B, C y D estaban involucrados y se establecieron las cuatro afirmaciones siguientes:

- Si A y B son culpables entonces C es culpable.
- Si A es culpable entonces al menos uno entre B y C es culpable.
- Si C es culpable entonces D también es culpable.
- Si A es inocente entonces D es culpable.

Ejemplo Prolog

a) Formule declarativamente las proposiciones anteriores en Prolog.

%Si A y B son culpables entonces C es culpable.

```
p1(c,c,c,_).
p1(c,i,_, _).
p1(i,c,_, _).
p1(i,i,_, _).
```

%Si A es culpable entonces al menos uno entre B y C es culpable

```
p2(c, c, _, _).
p2(c,_,c, _).
p2(i, _, _, _).
```

%Si C es culpable entonces D también es culpable.

```
p3(_,_,c, c).
p3(_,_,i, _).
```

%Si A es inocente entonces D es culpable.

```
p4(i,_,_, c).
p4(c,_,_,_).
```

Ejemplo Prolog

a) Declare el predicado solve(A, B, C, D) que triunfa si las proposiciones anteriores son verdaderas atendiendo a la inocencia o culpabilidad de los sospechosos.

Ejemplo:

:-solve(culpable, B, C, culpable)

B = C, C = culpable

solve(A, B, C, D):- p1(A,B,C,D), p2(A,B,C,D), p3(A,B,C,D), p4(A,B,C,D).

Ejemplo Prolog

a) Suponga que el Inspector Craig cree que D es culpable, formule un objetivo que permita corroborar o refutar la suposición del inspector Craig. Justifique su respuesta.

solve(_,_,_,c)