Universidad Tecnológica Nacional
FACULTAD REGIONAL CORDOBA

PARADIGMAS DE PROGRAMACION

Unidad V
Paradigma Lógico
Parte III

CONTENIDOS ABORDADOS

- Repaso de conceptos
- Búsquedas de soluciones.
- Unificación
- Backtracking.
- El corte y el fallo.

Repaso de Términos

- Un término o predicado, se compone de un functor seguido de cero a N argumentos entre paréntesis y separados por comas.
- En LPO, Enunciado atómico
  - Genérico: estudia(Persona, Materia)
  - Espécifico: estudia(ana, matematica)
- Representaciones:
  - enseña(Docente, Alumno1, Alumno2)
  - enseña(ana, juan, maria)

  } EA simple

  ó

  - enseña(Docente, Alumno)
  - enseña(ana, juan) and enseña(ana, maria)

  } EA combinado

Prolog: Repaso de Estructura

Hechos *alumno(ana)*

(afirmación incondicional)

B<-

Reglas *alumno(ana) :- cursaMateria(ana)*

(afirmación condicional)

B<-A1,A2,…,An

archivo.pl

Consultas

(objetivo)

<-A1,A2,…,An

exactas: ? alumno(ana).

true o false

con variables: ? alumno(X).

X=ana

interprete de comandos

Ejemplificación

Definición de hechos y reglas:

```
alumno (jose).
alumno (ana).
alumno (juan).
aprobo_primer_parcial(jose).
aprobo_primer_parcial(juan).
aprobo_primer_parcial(ana).
aprobo_segundo_parcial(jose).
aprobo_segundo_parcial(ana).
alumno_regular (X) :- alumno (X),
                      aprobo_primer_parcial(X),
                      aprobo_segundo_parcial(X).
```

hechos

regla

Prolog: Búsqueda de soluciones

- Una llamada concreta a un predicado, con unos argumentos concretos, se denomina **objetivo** (goal).
- Todos los objetivos tiene un resultado de **éxito o fallo** tras su ejecución indicando si el predicado es cierto (verdadero) para los argumentos dados, o por el contrario falso.
- Cuando un objetivo tiene éxito las variables libres que aparecen en los argumentos pueden quedar **ligadas**. Estos son los valores que hacen cierto el predicado. Si el predicado falla, no ocurren ligaduras en las variables **libres**.
- El caso más básico es aquél que no contiene variables:

```
?- alumno_regular(jose).
true
?- alumno_regular(juan).
false
```

Prolog: Búsqueda de soluciones

- Una consulta puede contener variables. En este caso PROLOG buscara por toda la base de hechos aquellos objetos que pueden ser representados por la variable. ejemplo:

```
?- alumno_regular(X).
```

es posible que existan varios valores para dicha variable que hacen cierto el objetivo.

```
X = jose
X = ana
```

En este caso obtenemos todas las combinaciones de ligaduras para las variables que hacen cierto el objetivo.

La búsqueda de soluciones

Secuencias de objetivos o consultas

- Los objetivos se ejecutan secuencialmente por orden de escritura (es decir, de izquierda a derecha).
- Si un objetivo falla, los siguientes objetivos ya no se ejecutan. Además la conjunción, en total, falla.
- Si un objetivo tiene éxito, algunas o todas sus variables quedan ligadas, y por tanto, dejan de ser variables libres para el resto de objetivos en la secuencia.
- Si todos los objetivos tienen éxito, la conjunción tiene éxito y mantiene las ligaduras de los objetivos que la componen.

La búsqueda de soluciones

- Para seguir buscando respuestas posibles se pondrá el punto y coma (;) hasta que el sistema nos informe que no hay más soluciones.
- Por ejemplo:

Hechos		?- enseña(ana, Alguien).
		El resultado de la consulta es:
enseña(ana, juan).	<----------	Alguien = juan;
enseña(ana, maria).	<----------	Alguien = maria;
enseña(ana, pablo).	<----------	Alguien = pablo.
		?-

Principio de resolución o regla de inferencia

- Es un algoritmo que fue propuesto por Robinson.
- Propone una **regla de inferencia** a la que llama **resolución**, mediante la cual la demostración de un teorema puede ser llevada a cabo de manera automática.
- Prolog utiliza este principio de resolución y trabaja con cláusulas de Horn.
- Esta regla se aplica sobre cláusulas y la demostración de teoremas se lleva a cabo por reducción al absurdo.

Principio de resolución o regla de inferencia

- La implementación de la Regla de Inferencia en Prolog se basa en :
  - **La Unificación.**
  - **El Backtracking.**

Unificación

- Es el mecanismo mediante el cual las variables lógicas toman valor en Prolog.
- Cuando una variable no tiene valor se dice que está libre. Pero una vez que se le asigna valor, éste ya no cambia, por eso se dice que la variable está ligada. Ejemplo:

  ```
  ?- docente(X).
  ```

  ```
  docente('ana')
  ```
  el valor ana se liga a la variable X
- Dos términos unifican cuando existe una posible ligadura (asignación de valor) de las variables, tal que ambos términos son idénticos sustituyendo las variables por dichos valores. Ejemplo: Para `k(A,B) y k(4,5).`

  El primer argumento, A se liga al valor 4.

  El segundo argumento B, se liga al valor 5.

Unificación

- Otro ejemplo:

  objetivo: <-p1(A,Z), p2(Z,B).

  hechos:

  p1(1,3).===> unifica 1

  p1(2,1).

  p1(3,2). ===> unifica 2

  p2(7,5).

  p2(3,2).===> unifica 1

  p2(9,8).

  p2(2,4).===> unifica 2

  p2(2,2).===> unifica 2

  unificación 1: <-p1(1,3), p2(3,2). Resultado: A = 1, Z = 3, B = 2

  unificación 2: <-p1(3,2), p2(2,4). Resultado: A = 3, Z = 2, B = 4

  unificación 2: <-p1(3,2), p2(2,2). Resultado: A = 3, Z = 2, B = 2

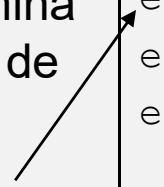
Unificación

Normas de unificación:

- Una variable siempre unifica con un término, quedando ésta ligada a dicho término.
- Dos variables siempre unifican entre sí, además, cuando una de ellas se liga a un término, todas las que unifican se ligan a dicho término.
- Para que dos términos unifiquen, deben tener el mismo functor y la misma aridad. Después se comprueba que los argumentos unifican uno a uno manteniendo las ligaduras que se produzcan en cada uno.
- Si algún término no unifica, ninguna variable queda ligada.

Unificación y búsqueda de soluciones

- Con la **unificación**, cada *objetivo* determina un subconjunto de cláusulas susceptibles de ser ejecutadas (*punto de elección)*.
- Prolog selecciona el primer punto de elección y sigue ejecutando el programa hasta determinar si el objetivo es verdadero o falso.
- Si en un punto caben varios caminos, se recorren en el orden que aparecen en el programa, esto es, de arriba a abajo y de izquierda a derecha.



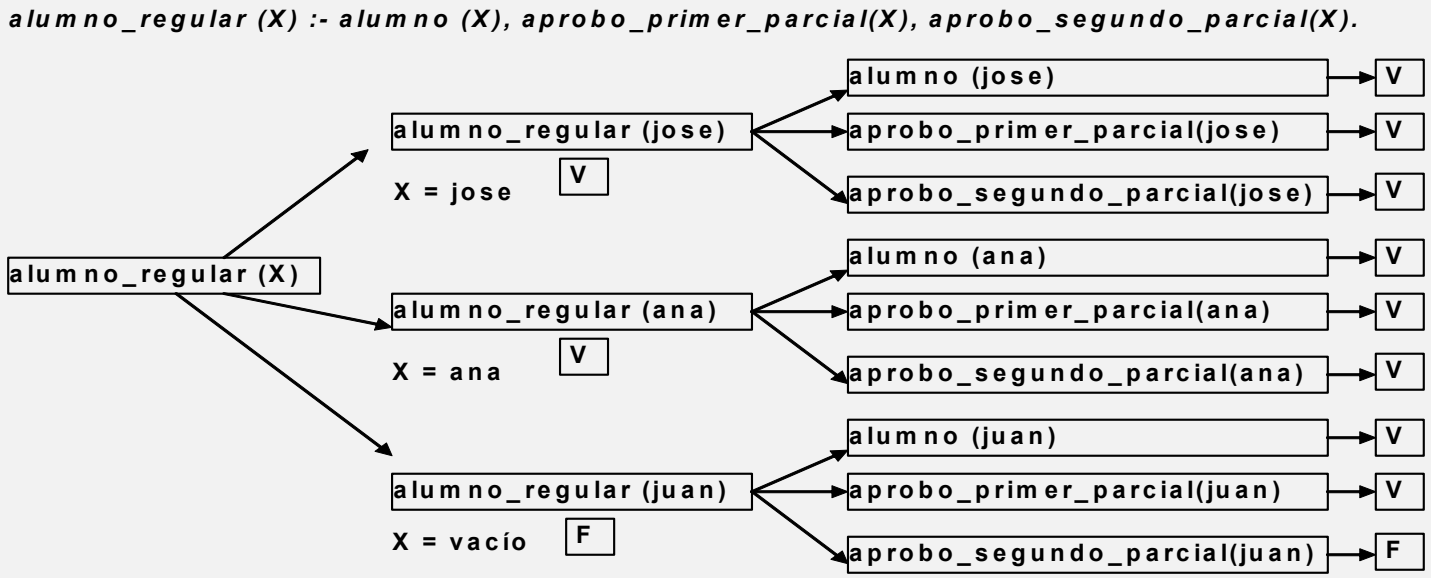
```
enseña(ana,juan).
enseña(ana,maria).
enseña(ana,pablo).

?- enseña(ana,Alguien).

La variable Alguien
unifica con :
        Alguien = juan;
        Alguien = maria;
        Alguien = pablo.
```

BackTracking y búsqueda de soluciones

- Es un mecanismo que consiste en realizar una búsqueda en profundidad y retroceso tratando de unificar la cláusula objetivo con las contenidas en la base de hechos, hasta lograr alcanzar la cláusula vacía.



BackTracking

- Si el objetivo de una búsqueda es falso entra en juego el ***backtracking***, que consiste en deshacer todo lo ejecutado situando el programa en el mismo estado en el que estaba justo antes de llegar al punto de elección. Entonces se toma el siguiente punto de elección que estaba pendiente y se repite de nuevo el proceso. Todos los objetivos terminan su ejecución bien en *éxito* ("verdadero"), bien en *fracaso* ("falso").

BackTracking

Características generales

- Cada solución es el resultado de una secuencia de decisiones.
- Las decisiones pueden deshacerse ya sea porque no lleven a una solución o porque se quieran explorar todas las soluciones (para obtener la solución óptima)
- Existe una función objetivo que debe ser satisfecha u optimizada por cada selección
- Las etapas por las que pasa el algoritmo se pueden representar mediante un árbol de expansión.
- El árbol de expansión no se construye realmente, sino que está implícito en la ejecución del algoritmo.
- Cada nivel del árbol representa una etapa de la secuencia de decisiones.

El control del backtracking

- El control del bactracking se puede hacer con:
  - **El predicado corte (!).**
  - **El predicado fail.**

Prolog: El control del backtracking

El predicado corte (!)

- Es un predicado que siempre se cumple, que genera un resultado verdadero en la primera ejecución, y falla en el proceso de *backtracking*, impidiendo dicho retroceso.
- El corte tiene la propiedad de eliminar los puntos de elección del predicado que lo contiene.
- Su aplicación principal es generar código más eficiente por el efecto que causa en la reducción o poda del árbol de búsqueda generado durante el procedimiento de resolución.
- Su utilización puede ser muy importante para que el programa funcione mas rápido y no malgaste memoria y tiempo en intentar satisfacer objetivos que podemos decir de antemano que nunca contribuirán a una solución.

Prolog: El control del backtracking

El predicado fail

- Es un predicado predefinido, sin argumentos que siempre falla, por tanto, implica la realización del proceso de retroceso (backtracking) para que se generen nuevas soluciones.
- Produce la generación de todas las posibles soluciones para un problema.
- Recordemos que cuando la *máquina Prolog* encuentra una solución para y devuelve el resultado de la ejecución. Con *fail* podemos forzar a que no pare y siga construyendo el árbol de búsqueda hasta que no queden más soluciones que mostrar.

Prolog: El control del backtracking

Ejemplo:

/*Hechos y reglas*/ alumno(ana). alumno(juan). alumno(pedro). alumnos(X) :- alumno(X). unAlumno(X) :- alumno(X),!. losAlumnos(X) :- alumno(X), write(X), nl, fail.	?- alumnos(X). X = ana ; X = juan ; X = pedro. ?- unAlumno(X). X = ana. ?- losAlumnos(X). ana juan pedro false.