La reevaluación y el "corte"

UNIVERSIDAD DE VALLADOLID

Índice:

Soluciones múltiples (I)

- Generación de soluciones finitas.
  Ejemplo: cuaderno de baile

```
chico(juan).
chico(pedro).
chico(roberto).

chica(maria).
chica(ana).
chica(rosa).
chica(marta).

posible_pareja(X, Y):- chico(X), chica(Y).
```

```
?- posible_pareja(X, Y).
X = juan        Y = maria ;
X = juan        Y = ana ;
X = juan        Y = rosa ;
X = juan        Y = marta ;

X = pedro       Y = maria ;
X = pedro       Y = ana ;
X = pedro       Y = rosa ;
X = pedro       Y = marta ;

X = roberto     Y = maria ;
X = roberto     Y = ana ;
X = roberto     Y = rosa ;
X = roberto     Y = marta ;
```

Soluciones múltiples (II)

- Generación de soluciones infinitas.
  Ejemplo: números naturales (recurrencia)

```
es_entero(0).
es_entero(X) :- es_entero(Y), X is Y+1.

    ?- es_entero(X)

    X=0;
    X=1;
    X=2; etc.
```

Corte (I)

- Ejemplo: Una biblioteca.
  - Libros existentes.
  - Libros prestados y a quién.
  - Fecha de devolución del préstamo.
  - Servicios básicos (accesibles a cualquiera):
    - Biblioteca de referencias o mostrador de consulta
  - Servicios adicionales (regla):
    - Préstamo normal o interbiblioteca.
  - Regla: no permitir servicios adicionales a personas con libros pendientes de devolución fuera de plazo.

Corte (II)

```
libros_por_devolver(c_perez, libro10089).
libros_por_devolver(a_ramos, libro29907).
.
.
.
cliente(a_ramos).
cliente(c_perez).
cliente(p_gonzalez).
.
.
```

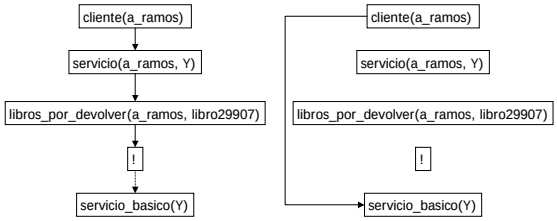
```
servicio(Pers, Serv) : -
        cliente(Pers),
        libros_por_devolver(Pers, Libro),
        !,
        servicio_basico(Serv).
servicio(Pers, Serv) :-
        servico_general(Serv).
servicio_basico(referencia).
servicio_basico(consulta).

servicio_adicional(prestamo).
servicio_adicional(pres_inter_biblio).

servicio_general(X) :- servicio_basico(X).
servicio_general(X) :- servicio_adicional(X).
```

Corte (III)

?- cliente(X), servicio(X, Y).



Corte (IV)

- Formalmente, el corte es un objetivo "!", que siempre se satisface.
- Fuerza a no evaluar más objetivos dentro de la cláusula.
- Las variables se instancian al valor antes del "corte".

Corte (V)

- Ahorro de tiempo no satisfaciendo objetivos que no contribuyen a solución alguna.
- Menor consumo de recursos.
- Herramienta para un correcto funcionamiento.
- Puede restar eficiencia ante preguntas no planteadas en el diseño del programa:

  ?-servicio(X, referencia).
  ?-cliente(X), servicio(X, referencia)

Aplicaciones del corte

- Decir a PROLOG: "si has llegado aquí, es que has escogido la regla adecuada para este objetivo".
- Forzar el fracaso de un objetivo: justo lo opuesto del punto anterior.
- Finalizar la generación de soluciones alternativas mediante reevaluaciones: "si has llegado aquí, es que has encontrado la solución única y no hay razón para continuar"

Confirmación de regla

- Ejemplo: sumar los N primeros naturales

```
sumara(1, 1) :- !.
sumara(N, X) :- N1 is N-1, sumara(N1, Res), X is Res+N.

?- sumara(7, X).

X=28 (7+6+5+4+3+2+1)

sumara(N, 1) :- N=<1, !.
sumara(N, X) :- N1 is N-1, sumara(N1, Res), X is Res+N.

sumara(1, 1).
sumara(N, X) :- not(N=1), N1 is N-1, sumara(N1, Res), X is Res+N.
```

Se puede demostrar que la utilización del corte para confirmar regla se puede Simular con el empleo del "not" (predefinido).

Combinación corte-"fail" (I)

- "fail" es un predicado predefinido en PROLOG.
- Siempre produce un fracaso en la satisfacción del objetivo.
- Desencadena proceso de reevaluación.

```
carnet_uva(X):- matriculado(X), fail.
matriculado(juan).
matriculado(pedro).
matriculado(maria).
matriculado(ana).
```

```
?- carnet_uva(X).

No
```

Combinación corte-"fail" (II)

- Ejemplo: filtro de más nivel.

```
contribuyente_medio(X):- extranjero(X), fail.
contribuyente_medio(X):- .....
?-contribuyente_medio(peter_smith).
Yes

contribuyente_medio(X):- extranjero(X), !, fail.
contribuyente_medio(X):- .....
?-contribuyente_medio(peter_smith).
No
```

- Es posible simular con operaciones lógicas.

Generación y comprobación (I)

- Ejemplo: plantear una relación dividir

```
divide(N1, N2, Resultado):-
        es_entero(Resultado),
        Producto1 is Resultado*N2,
        Producto2 is (Resultado+1)*N2,
        Producto1 =< N1, Producto2>N1, !.

es_entero(0).
es_entero(N) :- es_entero(Y), N is Y+1.
```

```
?- divide(100, 25, X).
X=4;
No

?- divide(100,3, X).
X=33;
No

?- divide(100, 5, 50)
<bloqueo>
```

- Generador -> relación *es_entero*
- Comprobador -> relación *divide*

Generación y comprobación (II)

- Ejemplo: juego de las tres en raya
  - Tablero: estructura de 9 elementos

X		O
X	O	X

Tablero=[x, "", o, "", "", "", x, o, x]

  - Jugadas en línea:
    - Horizontales: [1,2,3],[4,5,6],[7,8,9]
    - Verticales: [1,4,7],[2,5,8],[3,6,9]
    - Diagonales: [1,5,9],[3,5,7]

Generación y comprobación (III)

- Amenaza de línea:
  - vacío, cruz, cruz.
  - cruz, vacío, cruz.
  - cruz, cruz, vacío.
- OBJETIVO:
  - Movimiento forzoso (para las caras).
    - Generar líneas
    - Comprobar amenaza

Generación y comprobación (IV)

```
enlinea([1, 2, 3]).
enlinea([4, 5, 6]).
enlinea([7, 8, 9]).
enlinea([1, 4, 7]).
enlinea([2, 5, 8]).
enlinea([3, 6, 9]).
enlinea([1, 5, 9]).
enlinea([3, 5, 7]).

amenaza([X, Y, Z], B, X):- vacio(X, B), cruz(Y, B), cruz(Z, B).
amenaza([X, Y, Z], B, Y):- cruz(X, B), vacio(Y, B), cruz(Z, B).
amenaza([X, Y, Z], B, Z):- cruz(X, B), cruz(Y, B), vacio(Z, B).

vacio(Casilla, Tablero):- argumento(Casilla, Tablero, Valor), Valor=[].
cruz(Casilla, Tablero):- argumento(Casilla, Tablero, Valor), Valor=x.
cara(Casilla, Tablero):- argumento(Casilla, Tablero, Valor), Valor=o.

argumento(Posicion, Lista, X):- Posicion=1, arg(1, Lista, X).
argumento(Posicion, Lista, X):- Posicion>1, arg(2, Lista, Y), Pos is Posicion-1,
argumento(Pos, Y, X).
```

Generación y comprobación (V)

```
movimiento_forzoso(Tablero, Casilla) :- enlinea(Linea), amenaza(Linea, Tablero, Casilla), !.
```

Nota: Si se quita el corte de esta última regla, se generarían todas las casillas para evitar amenaza de línea. Esto supone que el jugador en cuestión (caras) ha perdido, porque solamente tiene derecho a un movimiento cada vez. Asimismo, es posible que una misma casilla sea solución de dos amenazas de línea; con este fin, también puede ser utilizado el corte.