



Estructuras de Control en PROLOG

- **Control de la búsqueda e instanciación**
- **Satisfacer/resatisfacer un objetivo**
- **El predicado corte (!)**
- **Aplicaciones del predicado corte**
 - **Confirmación de una regla**
 - **Combinación corte-fail**
 - **Detener la Generación y Comprobación**



Control de la búsqueda e instanciación

Generación de soluciones múltiples (I)

- Generación de soluciones finitas.
 - Ejemplo: cuaderno de baile

chico(antonio).
chico(juan).
chico(luis).
chico(pedro).

chica(ana).
chica(eva).
chica(isabel).
chica(maria).

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

?- posible_pareja(X, Y).

?- trace, posible_pareja(X, Y), notrace.



Generación de 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.`



El predicado corte, !

- 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.



El predicado Corte(II)

libros_por_devolver(c_perez, libro10089).
libros_por_devolver(a_amos, libro29907).

cliente(a_amos).
cliente(c_perez).
cliente(p_gonzalez).

servicio_basico(referencia).
servicio_basico(consulta).
servicio_adicional(prestamo).
servicio_adicional(pres_inter_biblio).

servicio(Pers, Serv):-
 cliente(Pers),
 libros_por_devolver(Pers, Libro),
 !,
 servicio_basico(Serv).

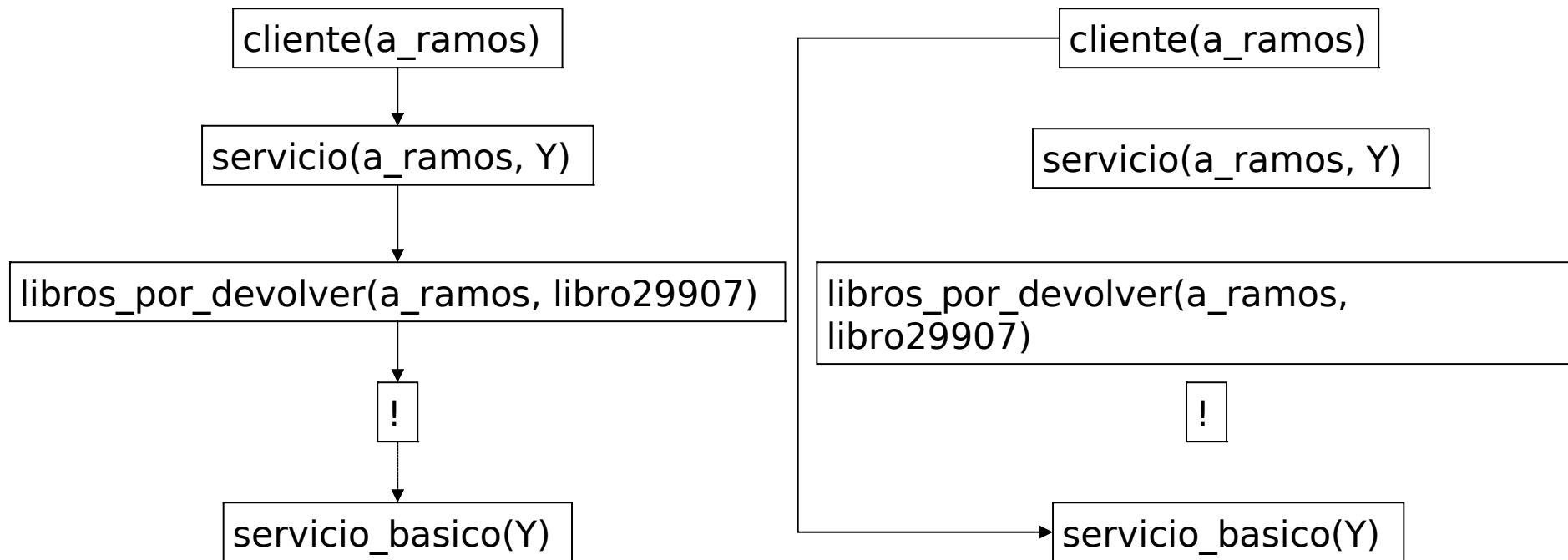
servicio(Pers, Serv):-
 servicio_basico(Serv).

servicio(Pers, Serv):-
 servicio_adicional(Serv).



El predicado Corte (III)

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





El predicado Corte (IV)

- Formalmente, el corte es un sub-objetivo, “!”, que siempre se satisface.
- Fuerza a no evaluar más sub-objetivos que estén antes que él dentro de la cláusula.
- Las variables se instancian al valor antes del “corte”, y no vuelven a intentar satisfacerse.



El predicado 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 predicado corte

- Indicar al sistema que ha llegado a un regla adecuada para satisfacer un objetivo (“si has llegado hasta aquí has escogido la opción adecuada”)
- Indicar al sistema que debe fallar y no buscar soluciones alternativas (“si has llegado hasta aquí, debes dejar de intentar satisfacer el objetivo”)
- Evitar la generación de soluciones múltiples mediante reevaluaciones (“si has llegado hasta aquí, has encontrado una solución, no sigas buscando”)



Confirmación de una regla [no hacer]

- Ejemplo: sumar los N primeros naturales

`sumar_n(1, 1) :- !.`

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

`?- sumar_n (7, X).`

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

`sumar_n(N, 1) :- N=<1, !.`

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

`sumar_n(1, 1).`

`sumar_n(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)

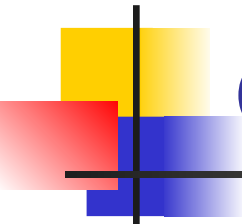
- “fail” es un predicado predefinido en PROLOG.
- Siempre produce un fracaso en la satisfacción del objetivo.
- Desencadena proceso de reevaluación.

carnet_uva(X):- matriculado(X),
fail.

matriculado(juan).
matriculado(pedro).
matriculado(maria).
matriculado(ana).

?- carnet_uva(X).

No



Combinación corte-“fail” (II)[no hacer]

- 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, !.

?- divide(100, 25, X).

X=4;

No

?- divide(100,3, X).

X=33;

No

es_entero(0).

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

?- divide(100, 5, 50)

<bloqueo>

- Generador -> relación *es_entero*
 Sirve como bucle for
- Comprobador -> relación *divide*



Ejercicios de Cálculo

- Calcular el término n-ésimo de la sucesión:

$$f(0)=0; f(1)=1; \dots f(n)=3 \cdot f(n-1)+2 \cdot f(n-2)$$

- Asimismo, obtener su correspondiente elemento dentro de la serie:

$$S(n)=\sum_{i=0}^n f(i)$$



Generación y comprobación (II)

- Práctica (entrega): 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