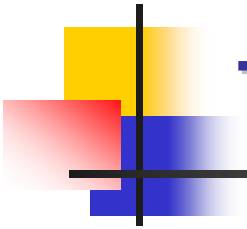
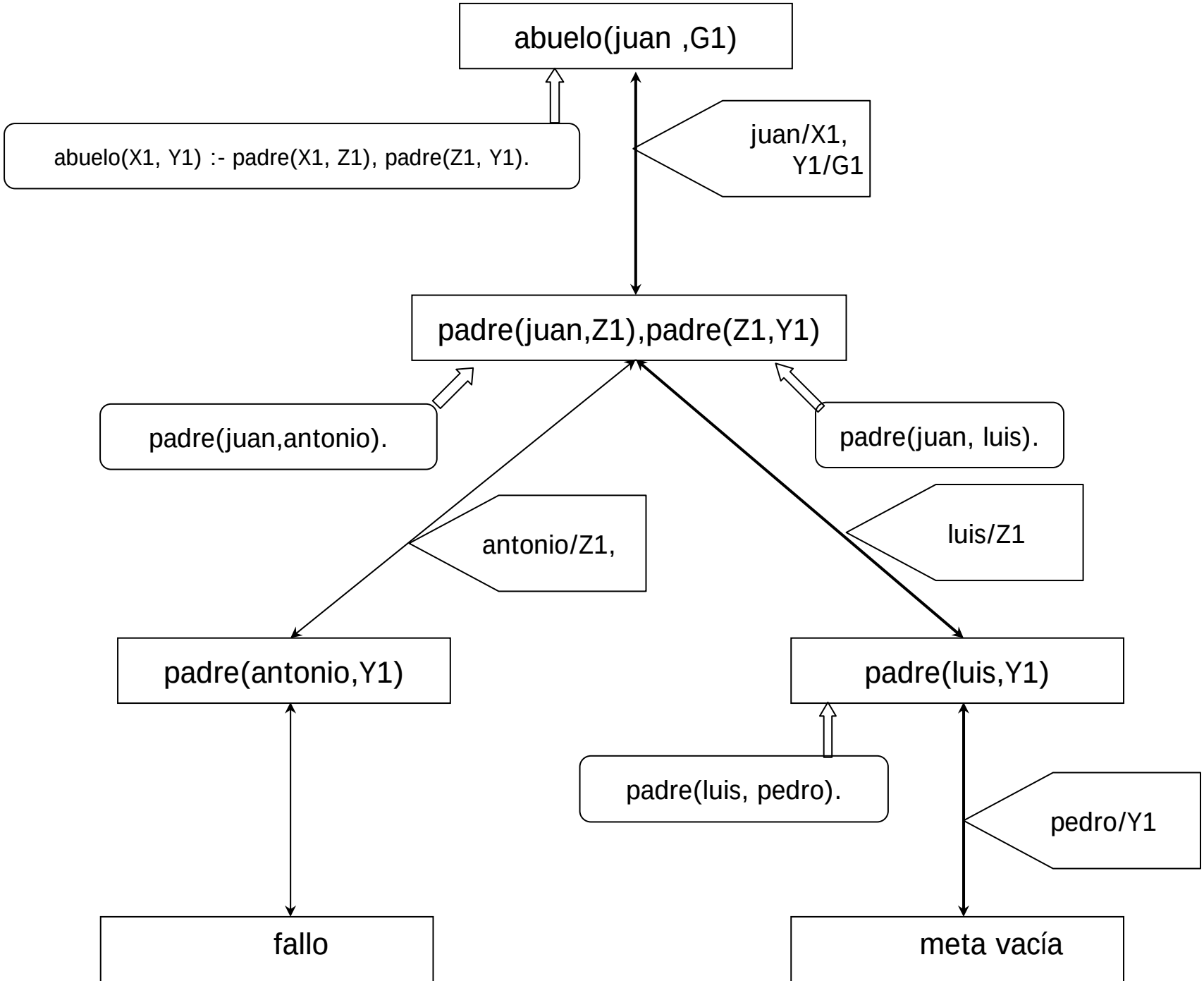


Tema 4. Control en PROLOG

1. ¿Cómo se satisface/resatisface un objetivo?
2. Procedimientos iterativos
3. El predicado corte (!)
4. Aplicaciones del predicado corte
    1. Confirmación de una regla
    2. Combinación corte-fail
    3. Detener la Generación y Prueba

4.1 Satisfacción de objetivos (I)

```
padre(juan, antonio).
padre(juan, luis).
padre(luis, pedro).
abuelo(X, Y) :- padre(X, Z), padre(Z, Y).

?- trace, abuelo(juan, X), notrace.
   Call: (8) abuelo(juan, _G382) ? creep
   Call: (9) padre(juan, _L162) ? creep
   Exit: (9) padre(juan, antonio) ? creep
   Call: (9) padre(antonio, _G382) ? creep
   Fail: (9) padre(antonio, _G382) ? creep
   Redo: (9) padre(juan, _L162) ? creep
   Exit: (9) padre(juan, luis) ? creep
   Call: (9) padre(luis, _G382) ? creep
   Exit: (9) padre(luis, pedro) ? creep
   Exit: (8) abuelo(juan, pedro) ? creep

X = pedro ; [trace]

No
```


abuelo(juan ,G1)
abuelo(X1, Y1) :- padre(X1, Z1), padre(Z1, Y1).
juan/X1,
Y1/G1
padre(juan,Z1),padre(Z1,Y1)
padre(juan,antonio).
padre(juan, luis).
antonio/Z1,
luis/Z1
padre(antonio,Y1)
padre(luis,Y1)
padre(luis, pedro).
pedro/Y1
fallo
meta vacía

Satisfacción de objetivos (II)

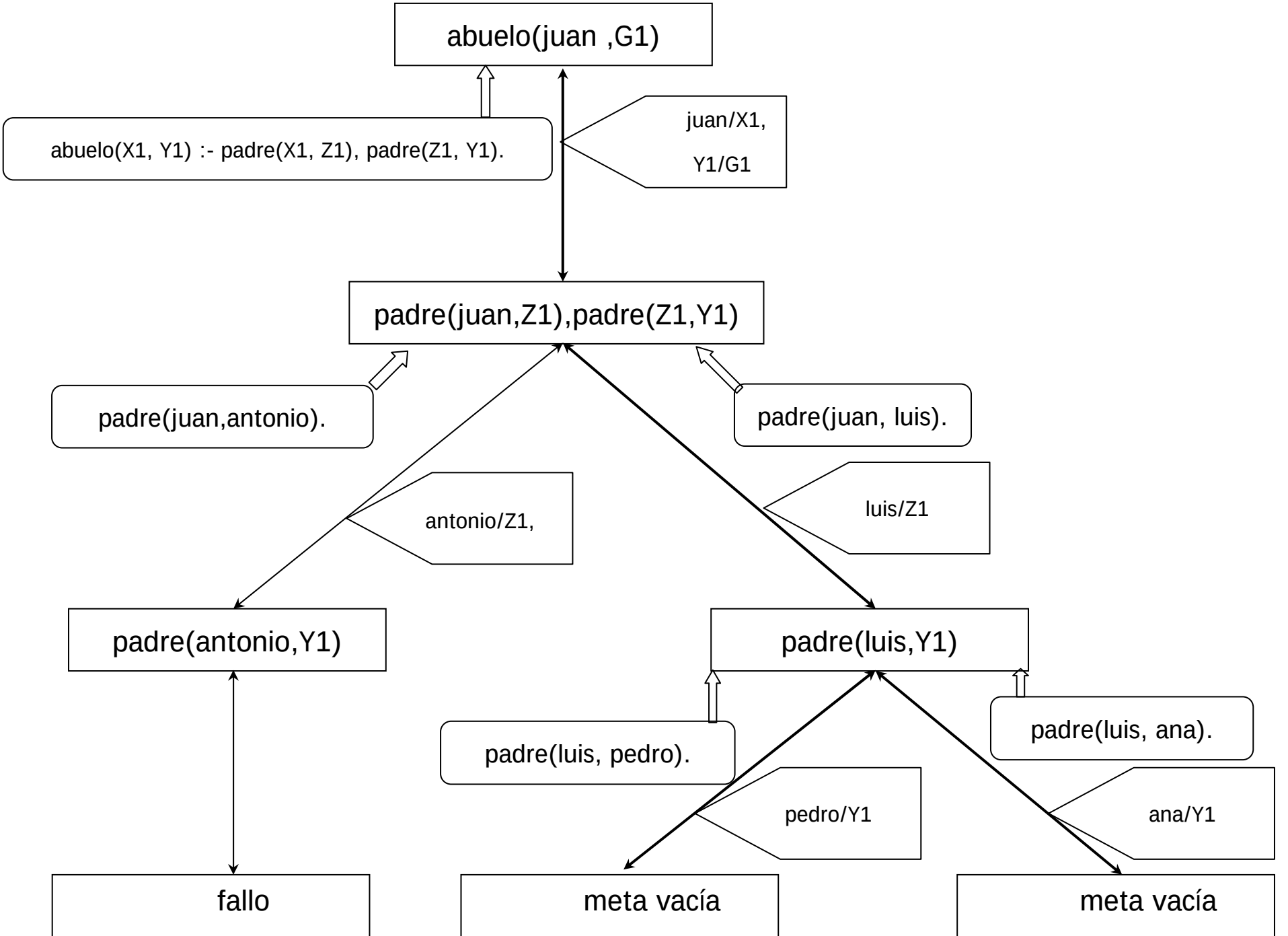
padre(juan, antonio). padre(juan, luis). padre(luis, pedro). padre(luis,ana).
abuelo(X, Y) :- padre(X, Z), padre(Z, Y).

?- trace,abuelo(juan,X),notrace.
   Call: (8) abuelo(juan, _G537) ? creep
   Call: (9) padre(juan, _L200) ? creep
   Exit: (9) padre(juan, antonio) ? creep
   Call: (9) padre(antonio, _G537) ? creep
   Fail: (9) padre(antonio, _G537) ? creep
   Redo: (9) padre(juan, _L200) ? creep
   Exit: (9) padre(juan, luis) ? creep
   Call: (9) padre(luis, _G537) ? creep
   Exit: (9) padre(luis, pedro) ? creep
   Exit: (8) abuelo(juan, pedro) ? creep

X = pedro ; [trace]
   Redo: (9) padre(luis, _G537) ? creep
   Exit: (9) padre(luis, ana) ? creep
   Exit: (8) abuelo(juan, ana) ? creep

X = ana ; [trace]

No

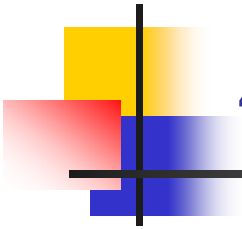

abuelo(juan ,G1)
abuelo(X1, Y1) :- padre(X1, Z1), padre(Z1, Y1).
juan/X1,
Y1/G1
padre(juan,Z1),padre(Z1,Y1)
padre(juan,antonio).
padre(juan, luis).
antonio/Z1,
luis/Z1
padre(antonio,Y1)
padre(luis,Y1)
padre(luis, pedro).
padre(luis, ana).
pedro/Y1
ana/Y1
fallo
meta vacía
meta vacía

Satisfacción de objetivos (III)

- Generación de soluciones infinitas: recursión por la izquierda

  Ejemplo: números naturales

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

?- es_entero(X).

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



4.2 Procedimientos iterativos

- Cláusula iterativa:
  - cláusula no recursiva
  - cláusula recursiva en la cual la llamada recursiva ocurre en el ultimo literal del cuerpo (recursividad final)

- Procedimiento iterativo: procedimiento que solo contiene cláusulas unitarias y cláusulas iterativas

Ejemplos procedimientos iterativos

progenitor(juan, antonio).

progenitor(juan, luis).

predecesor(X, Z) :- progenitor(X, Z).

predecesor(X, Z) :- progenitor(X, Y), predecesor(Y, Z).

- No iterativo

factorial(0,1).

factorial(N,F) :- N>0, N1 is N-1, factorial(N1,F1), F is N*F1.

- Versión iterativa

factorial(N,F):- factorial(0,N,1,F).

factorial(I,N,T,F):- I<N, I1 is I+1, T1 is T*I1, factorial(I1, N, T1, F).

factorial(N,N,F,F).

4.3 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.

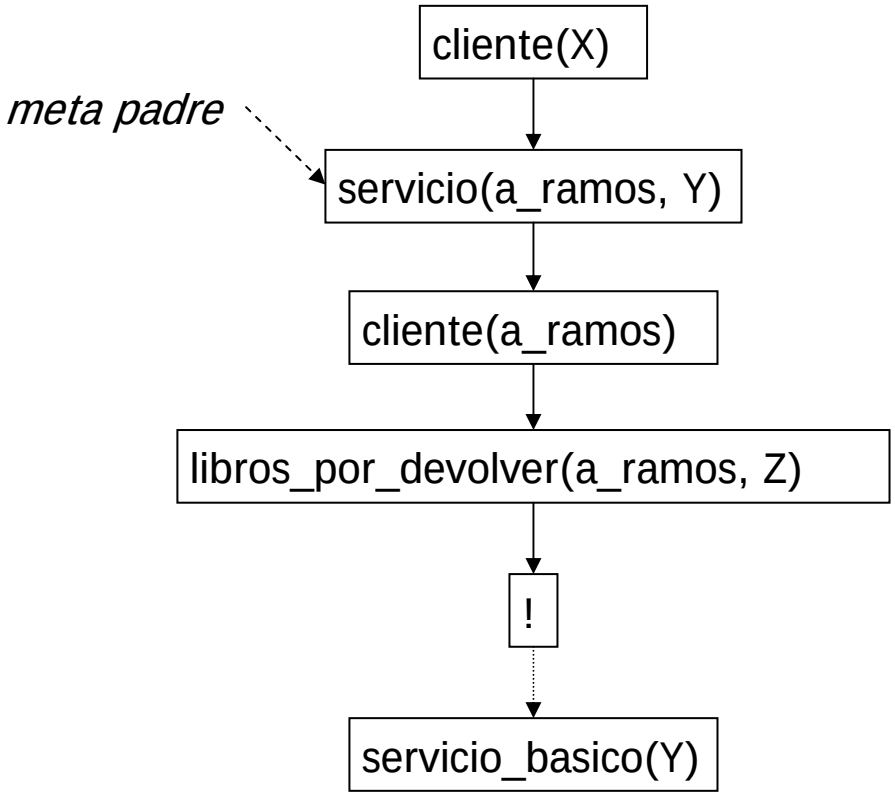
  - Regla: no permitir servicios adicionales a personas con libros pendientes de devolución fuera de plazo.

El predicado Corte(II)

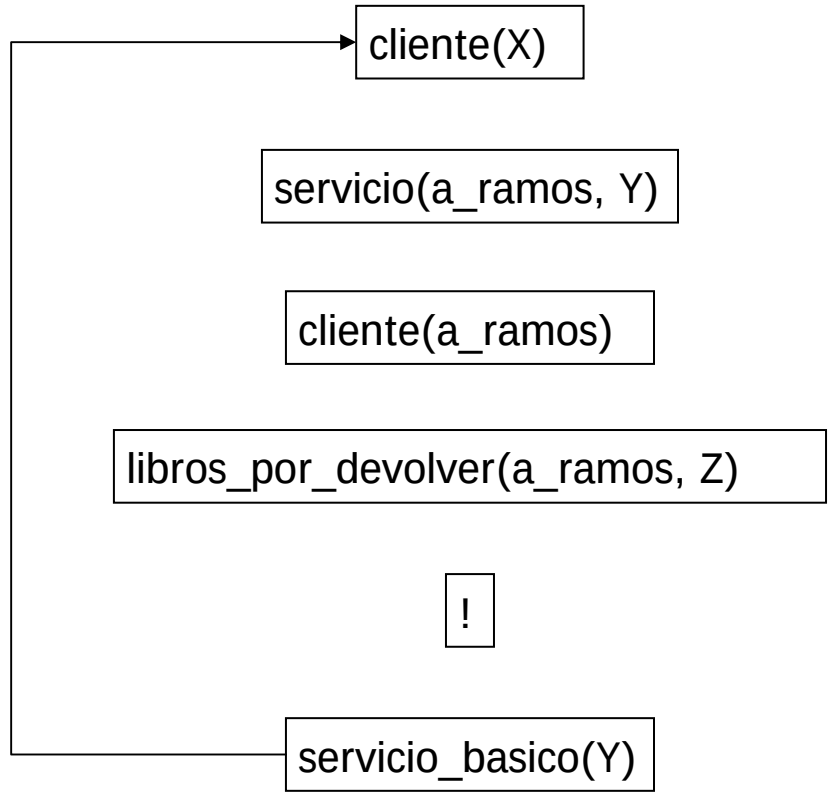
```
libros_por_devolver(c_perez, libro10089). libros_por_devolver(a_ramos, libro29907).
cliente(a_ramos). cliente(c_perez). cliente(p_gonzalez).
servicio_basico(referencia). servicio_basico(consulta).
servicio_adicional(prestamo). servicio_adicional(pres_inter_biblio).

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

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

servicio(Pers, Serv):-
    servicio_general(Serv).
```

El predicado Corte (III)

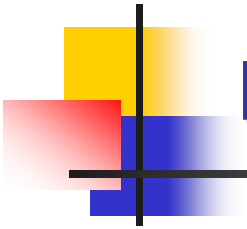
?- cliente(X), servicio(X, Y).



Si intentamos re-satisfacer:

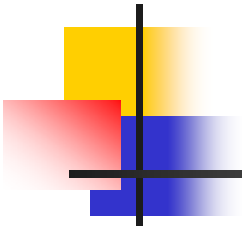


El predicado Corte (IV)

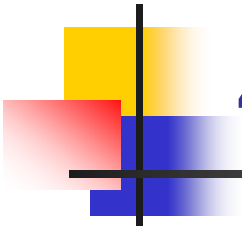


- El predicado corte siempre se satisface
- Definición operacional
  - Meta padre: meta que se redujo con la cláusula que contiene el predicado corte
  - Si se alcanza el predicado corte
    - Se satisface eliminándolo de la meta
  - Si se vuelve al predicado corte (backtracking)
    - Devuelve control a la meta anterior a la meta padre
- Obliga a mantener todas las elecciones realizadas entre la meta padre y el corte
  - Las posibles alternativas no se consideran

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.
- Modifica significado declarativo
  - El programa sin el corte no significa lo mismo



4.4 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”)

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(4, X).
        X=10 ;                  /* X=10 (4+3+2+1) */
        out of stack;           ¿por qué?

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


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

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)

- ‘A Elena le gustan los animales, salvo las serpientes’

```
gusta(elena, X):- serpiente(X), !, fail.
gusta(elena, X):-animal(X).

animal(snoopy).
animal(lamia).
serpiente(lamia).
gusta(elena, X):- serpiente(X), !, fail.
gusta(elena, X):-animal(X).

?- gusta(elena, lamia).
no
?- gusta(elena, snoopy).
Yes
?-gusta(elena,X).
no
```

Combinación corte-”fail” (III)

- Negación: ‘\+’

```
animal(snoopy).
animal(lamia).
serpiente(lamia).
gusta2(elena, X):-animal(X), \+ serpiente(X).

? gusta2(elena, lamia).
no
```

Generación y comprobación (I)

- Ejemplo: plantear una relación división entera

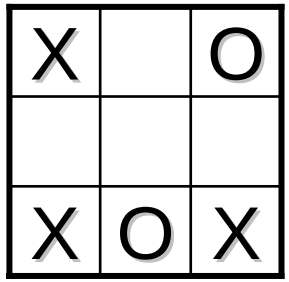
```
divide(N1, N2, Resultado):-
        es_entero(Resultado),
        Producto1 is Resultado*N2,
        Producto2 is (Resultado+1)*N2,
        Producto1 =< N1, Producto2>N1, !.

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



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]

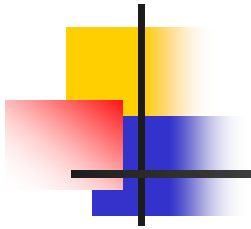
Generación y comprobación (III)

- 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]).

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).

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.

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).

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



caso1(X):- movimiento_forzoso([x, “”, o, “”, “”, “”, x, o, x], X).