Programación declarativa (2008–09)

Tema 9: Retroceso, corte y negación

José A. Alonso Jiménez

Grupo de Lógica Computacional
Departamento de Ciencias de la Computación e I.A.
Universidad de Sevilla

Tema 9: Retroceso, corte y negación

Tema 9: Retroceso, corte y negación

Ejemplo de nota sin corte

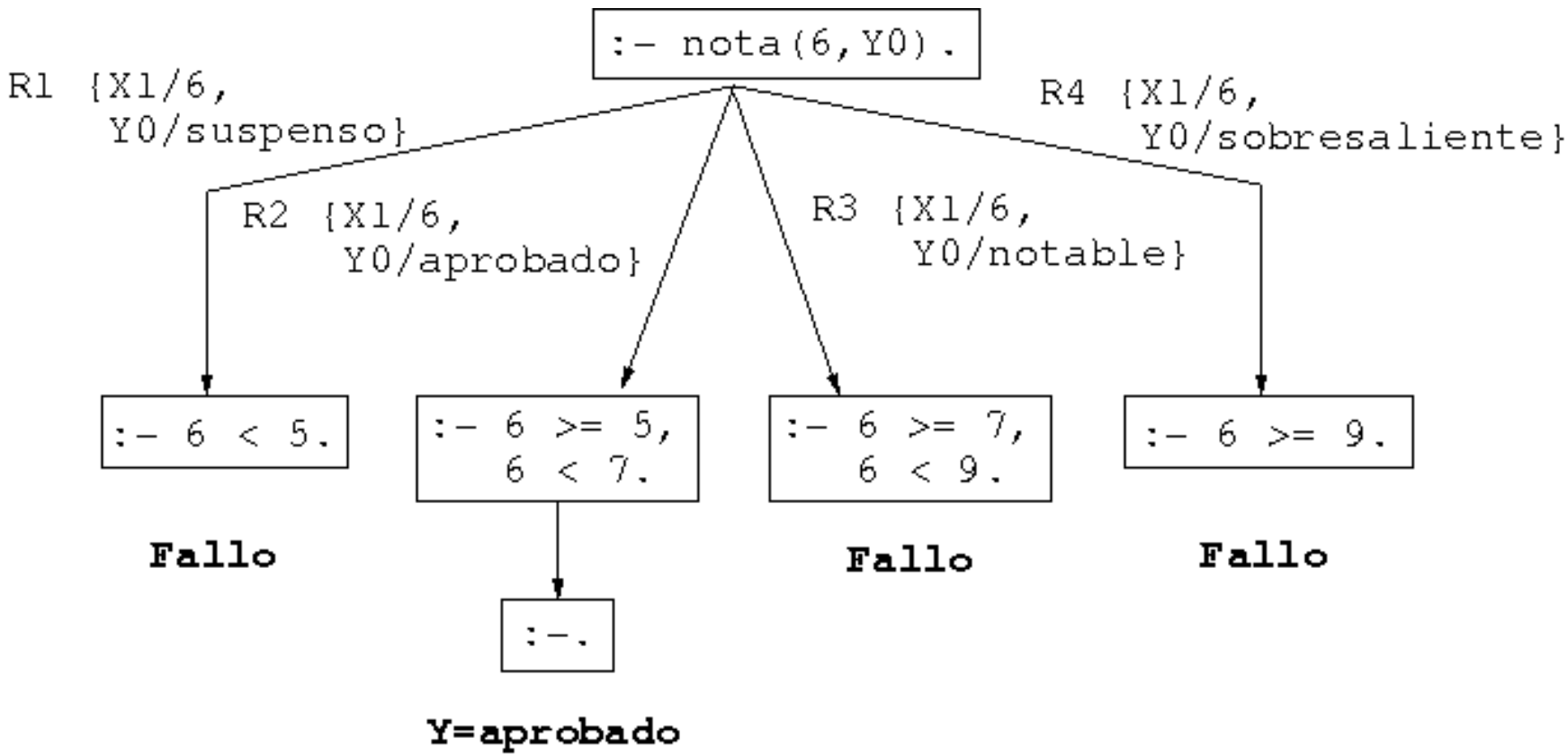
- `nota(X,Y)` se verifica si `Y` es la calificación correspondiente a la nota `X`; es decir, `Y` es `suspenso` si `X` es menor que 5, `Y` es `aprobado` si `X` es mayor o igual que 5 pero menor que 7, `Y` es `notable` si `X` es mayor que 7 pero menor que 9 e `Y` es sobresaliente si `X` es mayor que 9. Por ejemplo,

```
?- nota(6,Y).
Y = aprobado;
No
```

```
nota(X,suspenso)       :- X < 5.
nota(X,aprobado)       :- X >= 5, X < 7.
nota(X,notable)        :- X >= 7, X < 9.
nota(X,sobresaliente)  :- X >= 9.
```

Deducción en el ejemplo sin corte

- Árbol de deducción de `?- nota(6,Y).`

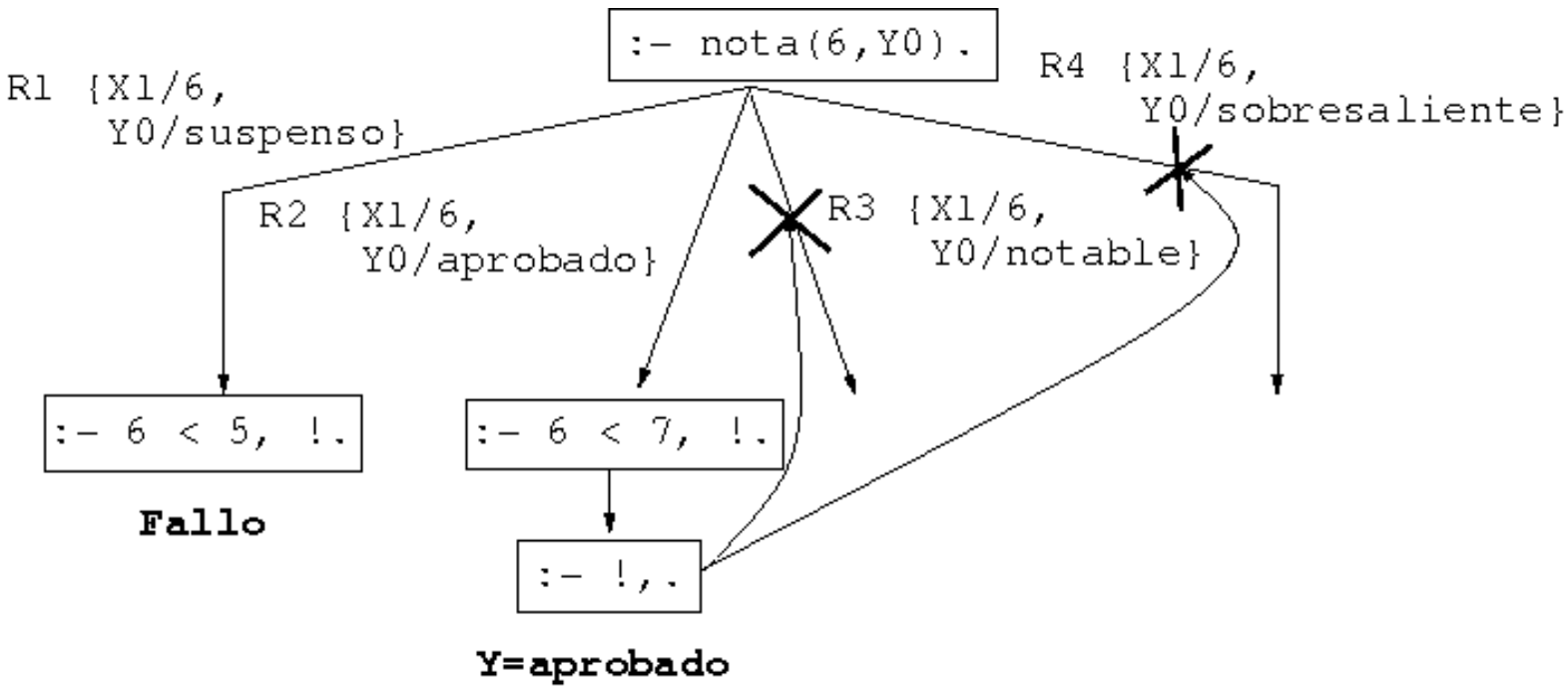


Ejemplo de `nota` con cortes

- Definición de `nota` con cortes

```
nota(X,suspenso)        :- X < 5, !.
nota(X,aprobado)        :- X < 7, !.
nota(X,notable)         :- X < 9, !.
nota(X,sobresaliente).
```

Deducción en el ejemplo con cortes



- ¿Un 6 es un sobresaliente?

```
?- nota(6,sobresaliente).
Yes
```

Tema 9: Retroceso, corte y negación

Uso de corte para respuesta única

- Diferencia entre `member` y `memberchk`

```
?- member(X,[a,b,a,c]), X=a.
X = a ;
X = a ;
No
?- memberchk(X,[a,b,a,c]), X=a.
X = a ;
No
```

- Definición de `member` y `memberchk`:

```
member(X,[X|_]).
member(X,[_|L]) :- member(X,L).

memberchk(X,[X|_]) :- !.
memberchk(X,[_|L]) :- memberchk(X,L).
```

Tema 9: Retroceso, corte y negación

Definición de la negación como fallo

- Definición de la negación como fallo `not`:

```
no(P) :- P, !, fail.          % No 1
no(P).                        % No 2
```

Tema 9: Retroceso, corte y negación

Programa con negación

- Programa:

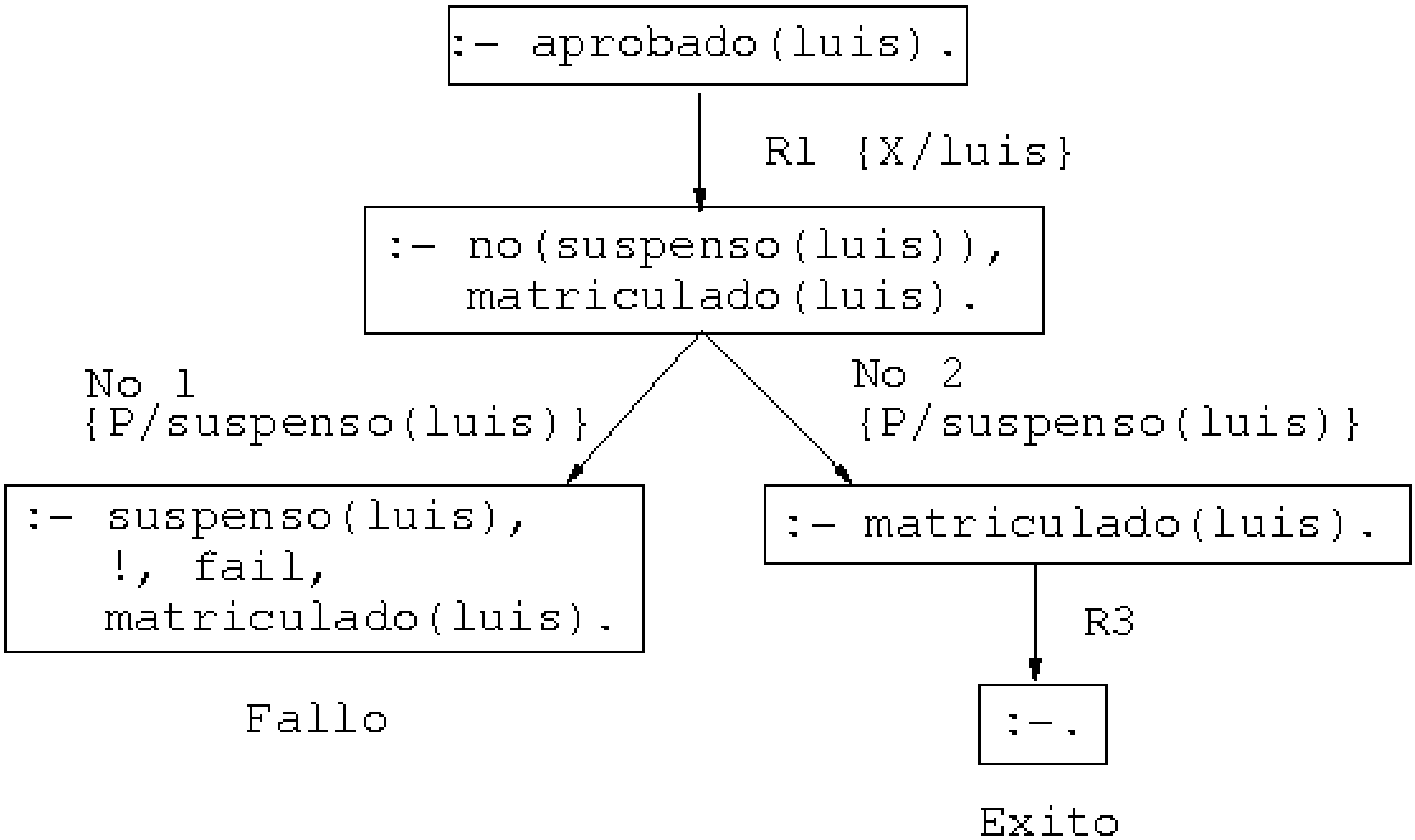
```
aprobado(X) :-                    % R1
   no(suspenso(X)),
   matriculado(X).
matriculado(juan).                % R2
matriculado(luis).                % R3
suspenso(juan).                   % R4
```

- Consultas:

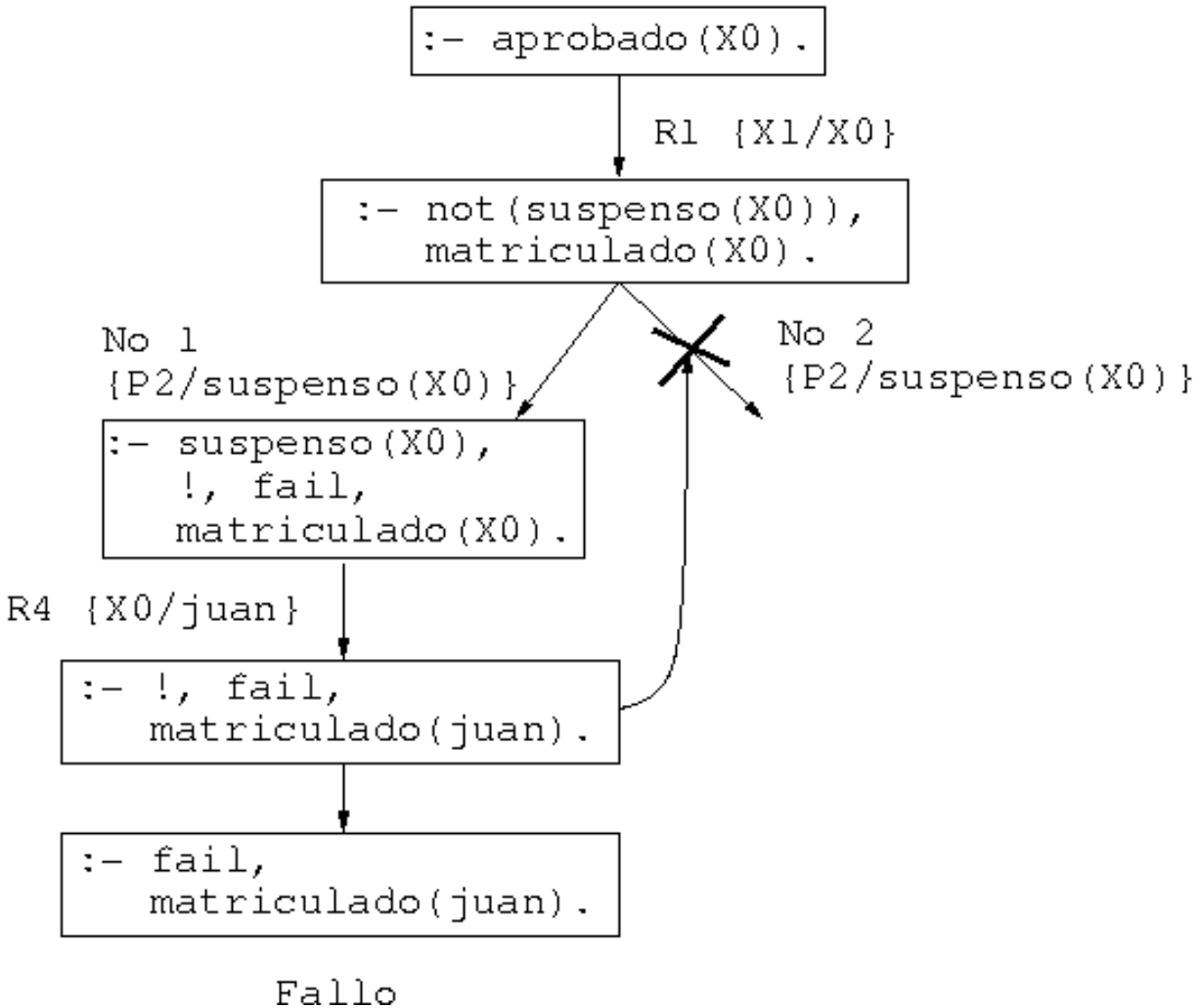
```
?- aprobado(luis).
Yes
```

```
?- aprobado(X).
No
```

Árbol de deducción de ?- aprobado(luis).



Árbol de deducción de ?- aprobado(X).



Modificación del orden de los literales

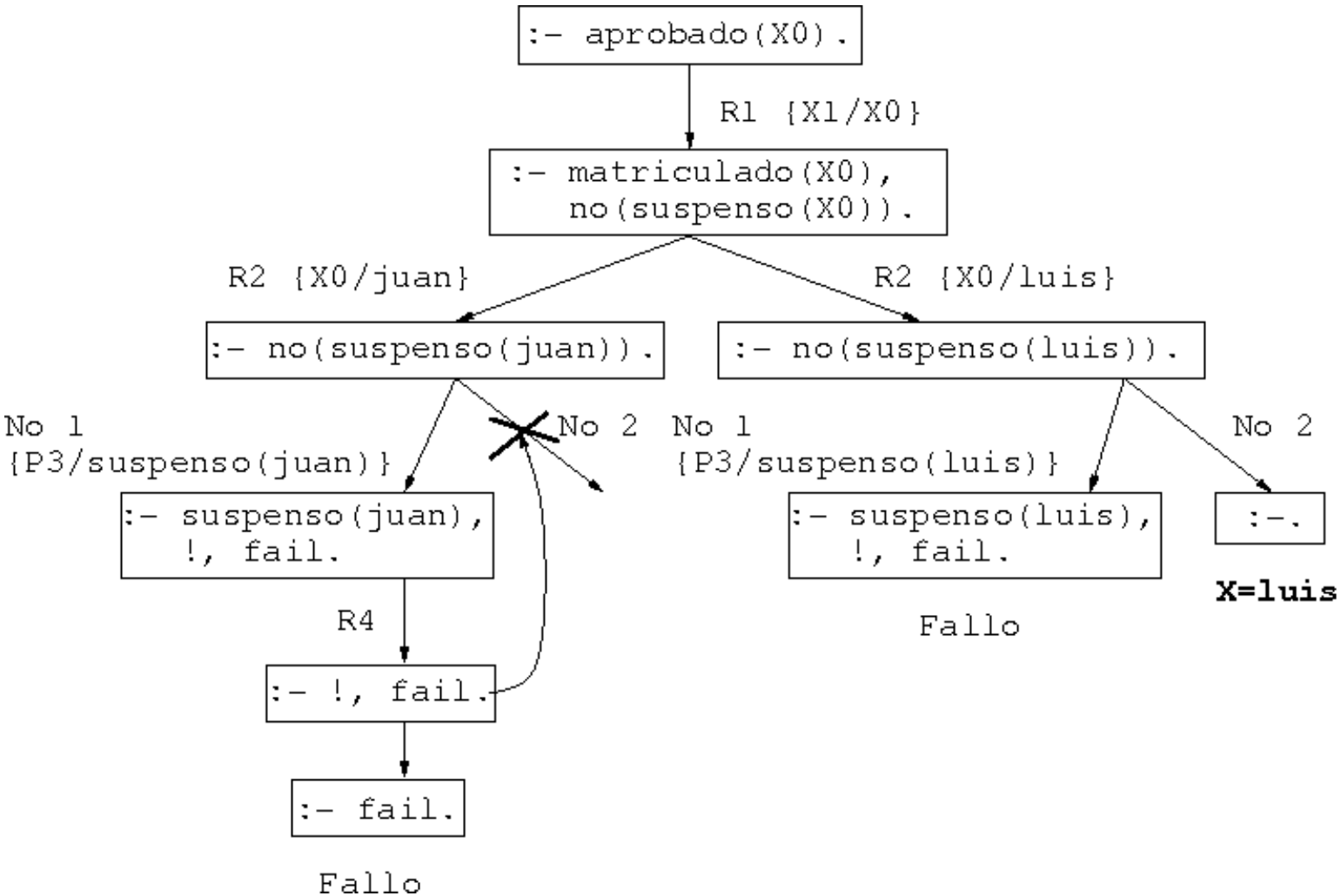
- Programa:

```
aprobado(X) :-              % R1
   matriculado(X),
   no(suspenso(X)).
matriculado(juan).          % R2
matriculado(luis).          % R3
suspenso(juan).             % R4
```

- Consulta:

```
?- aprobado(X).
X = luis
Yes
```

Árbol de deducción de ?- aprobado(X).



Ejemplo de definición con `not` y con corte

- `borra(L1,X,L2)` se verifica si `L2` es la lista obtenida eliminando los elementos de `L1` unificables simultáneamente con `X`. Por ejemplo,

```
?- borra([a,b,a,c],a,L).
L = [b, c] ;
No
?- borra([a,Y,a,c],a,L).
Y = a
L = [c] ;
No
?- borra([a,Y,a,c],X,L).
Y = a
X = a
L = [c] ;
No
```

Ejemplo de definición con `not` y con corte

- Definición con `not`:

```
borra_1([],_,[]).
borra_1([X|L1],Y,L2) :-
    X=Y,
    borra_1(L1,Y,L2).
borra_1([X|L1],Y,[X|L2]) :-
    not(X=Y),
    borra_1(L1,Y,L2).
```

Ejemplo de definición con `not` y con corte

- Definición con corte:

```
borra_2([],_,[]).
borra_2([X|L1],Y,L2) :-
    X=Y, !,
    borra_2(L1,Y,L2).
borra_2([X|L1],Y,[X|L2]) :-
    % not(X=Y),
    borra_2(L1,Y,L2).
```

Ejemplo de definición con `not` y con corte

- Definición con corte y simplificada

```
borra_3([],_,[]).
borra_3([X|L1],X,L2) :-
    !,
    borra_3(L1,Y,L2).
borra_3([X|L1],Y,[X|L2]) :-
    % not(X=Y),
    borra_3(L1,Y,L2).
```

Definición de `nota` con el condicional

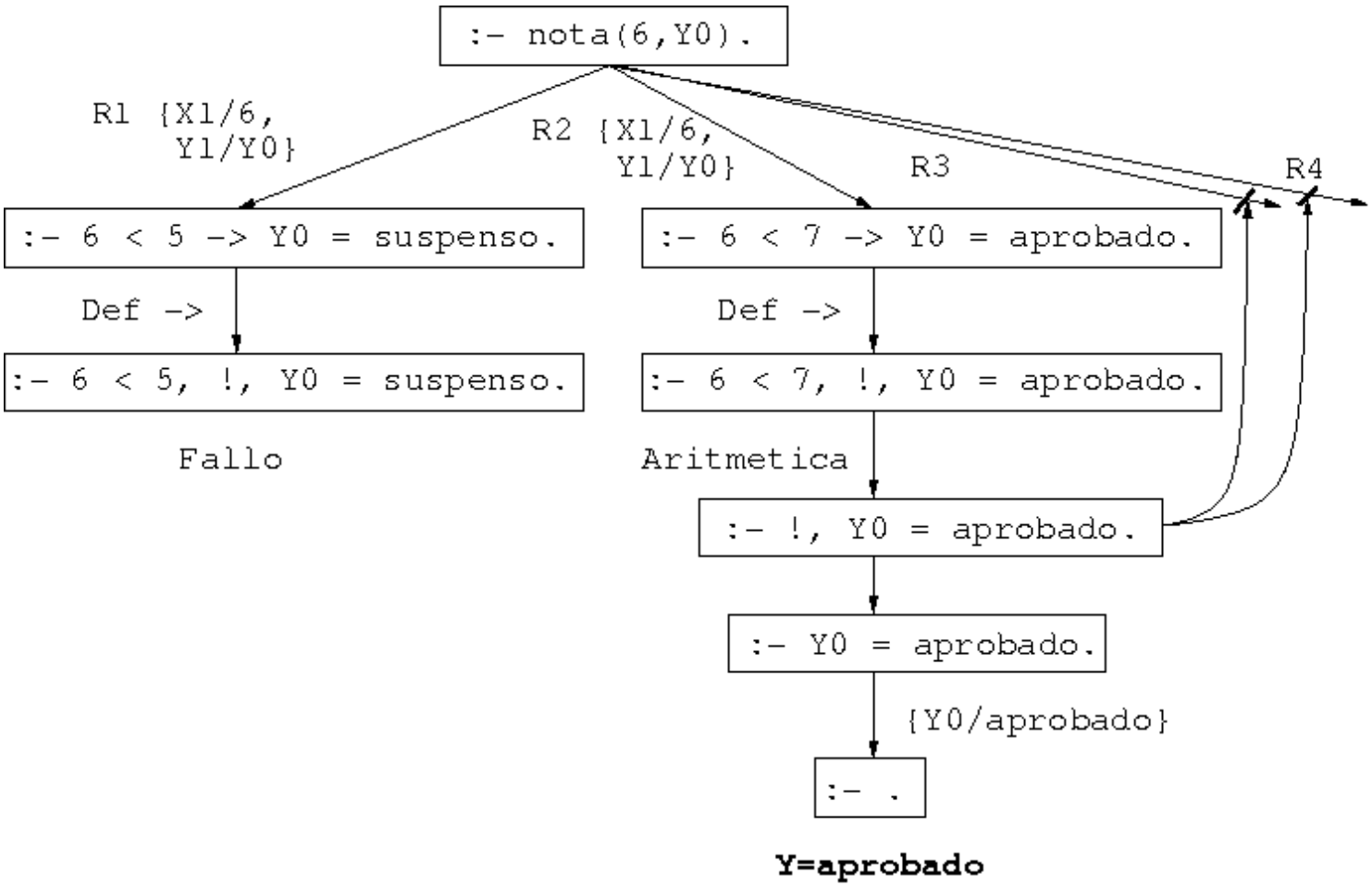
- Definición de `nota` con el condicional:

```
nota(X,Y) :-
    X < 5 -> Y = suspenso ;          % R1
    X < 7 -> Y = aprobado ;          % R2
    X < 9 -> Y = notable ;           % R3
    true  -> Y = sobresaliente.      % R4
```

- Definición del condicional y verdad:

```
P -> Q :- P, !, Q.                   % Def. ->
P -> Q.
true.                                % Def. true
```

Árbol de deducción de ?- nota(6,Y).



Bibliografía

1. J.A. Alonso *Introducción a la programación lógica con Prolog.*
   - Cap. 7: "Control mediante corte"
   - Cap. 8: "Negación
2. I. Bratko *Prolog Programming for Artificial Intelligence (3 ed.)* (Addison–Wesley, 2001)
   - Cap. 5: "Controlling backtracking"
3. W.F. Clocksin y C.S. Mellish *Programming in Prolog (Fourth Edition)* (Springer Verlag, 1994)
   - Cap. 4: "Backtracking and the cut"
4. L. Sterling y E. Shapiro *The Art of Prolog (2nd Edition)* (The MIT Press, 1994)
   - Cap. 11: "Cuts and negation"