

Programación declarativa (2005–06)

Tema 4: Retroceso, corte y negación

José A. Alonso Jiménez

Andrés Cordon Franco

Grupo de Lógica Computacional

Dpto. Ciencias de la Computación e Inteligencia Artificial

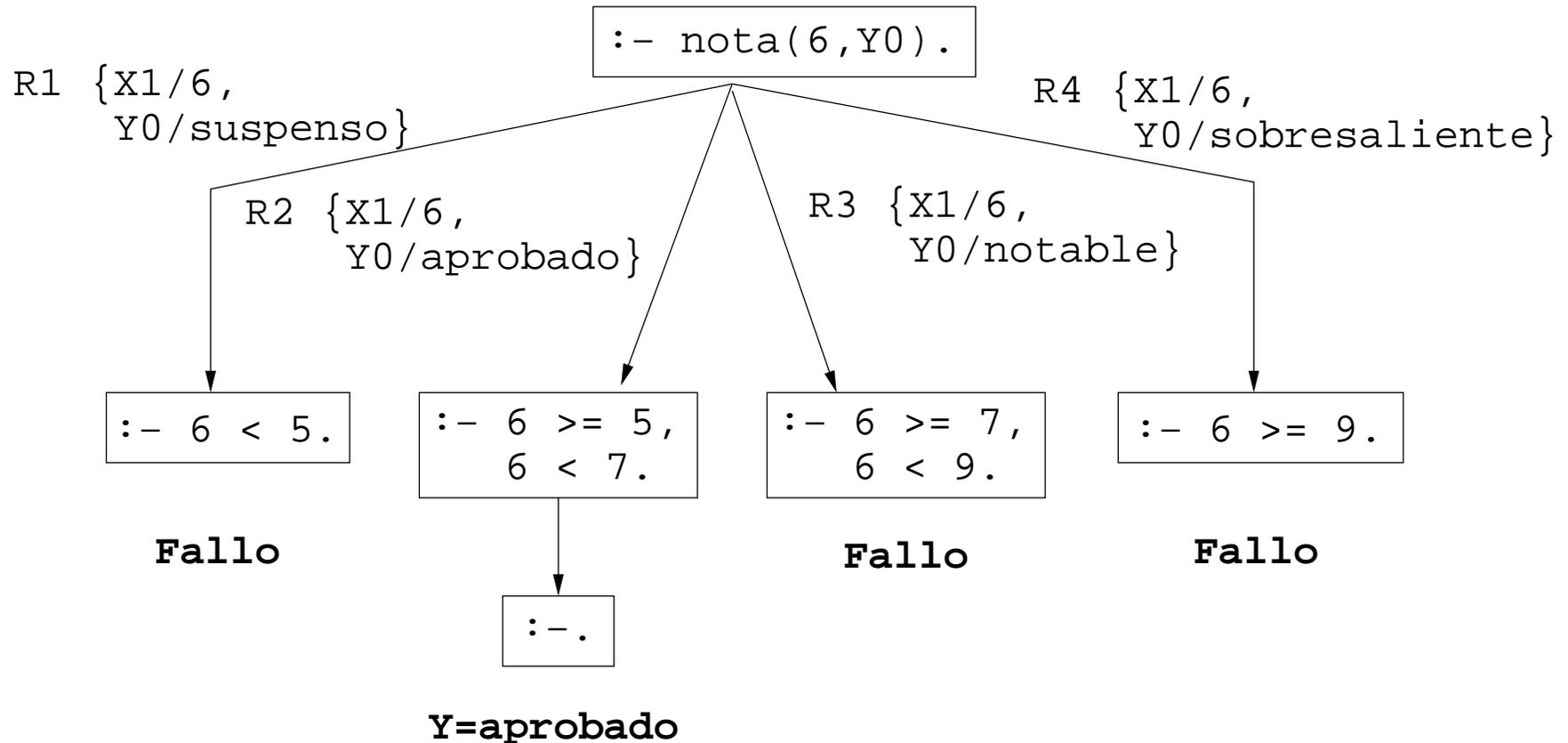
Universidad de Sevilla

Control mediante corte: Ejemplo sin corte

- Ejemplo de nota sin corte:
 - ▶ $\text{nota}(X, Y)$ 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.
 - ▶ Definición:
 $\text{nota}(X, \text{suspenso}) \quad :- \quad X < 5.$
 $\text{nota}(X, \text{aprobado}) \quad :- \quad X \geq 5, X < 7.$
 $\text{nota}(X, \text{notable}) \quad :- \quad X \geq 7, X < 9.$
 $\text{nota}(X, \text{sobresaliente}) \quad :- \quad X \geq 9.$
 - ▶ Ejemplo: ¿cuál es la calificación correspondiente a un 6?:
?- $\text{nota}(6, Y).$
 $Y = \text{aprobado};$
No

Control mediante corte: Deducción en el ejemplo sin corte

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



Control mediante corte: Ejemplo con corte

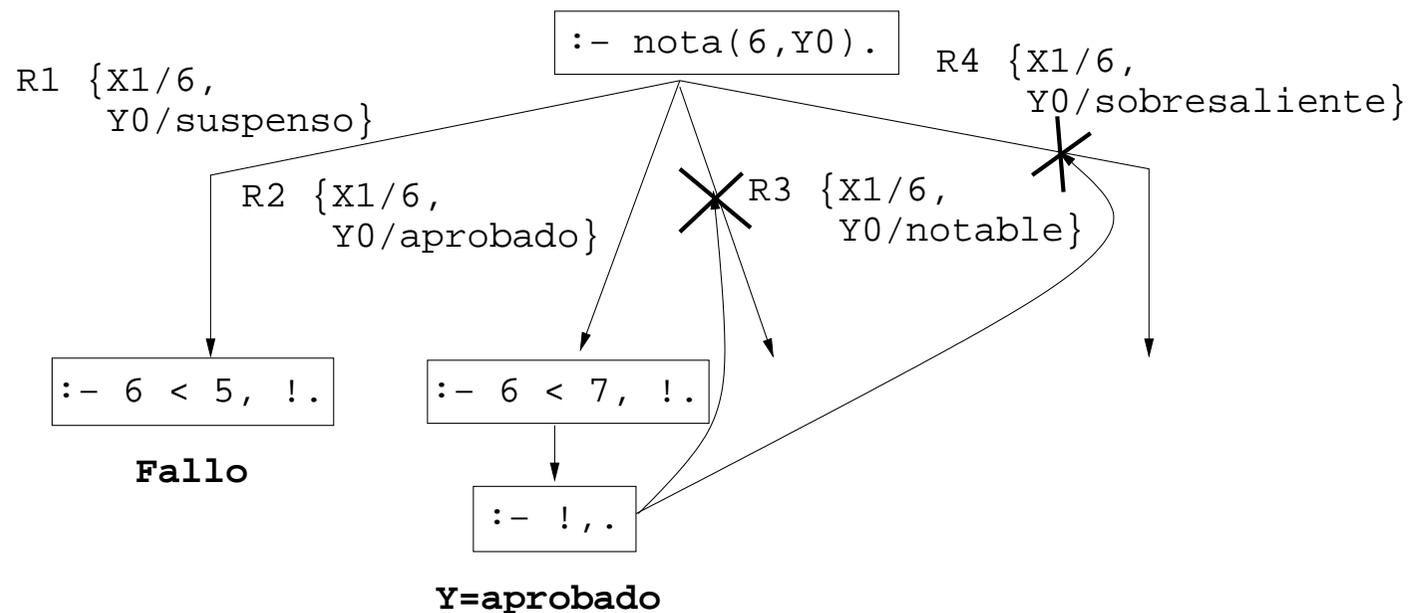
- Ejemplo de nota con cortes

nota (X, suspenso) $\text{:- } X < 5, !.$

nota (X, aprobado) $\text{:- } X < 7, !.$

nota (X, notable) $\text{:- } X < 9, !.$

nota (X, sobresaliente).



?- nota(6, sobresaliente).

Yes

Uso de corte para respuesta única

- 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)` .

Definición de la negación como fallo

- Negación como fallo:

- ▶ Definición de la negación como fallo (`not`):

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

- `no(P). % No 2`

- ▶ Programa con negación:

- `aprobado(X) :- no(suspenso(X)), matriculado(X). % R1`

- `matriculado(juan). % R2`

- `matriculado(luis). % R3`

- `suspenso(juan). % R4`

- ▶ Consultas:

- `?- aprobado(luis).`

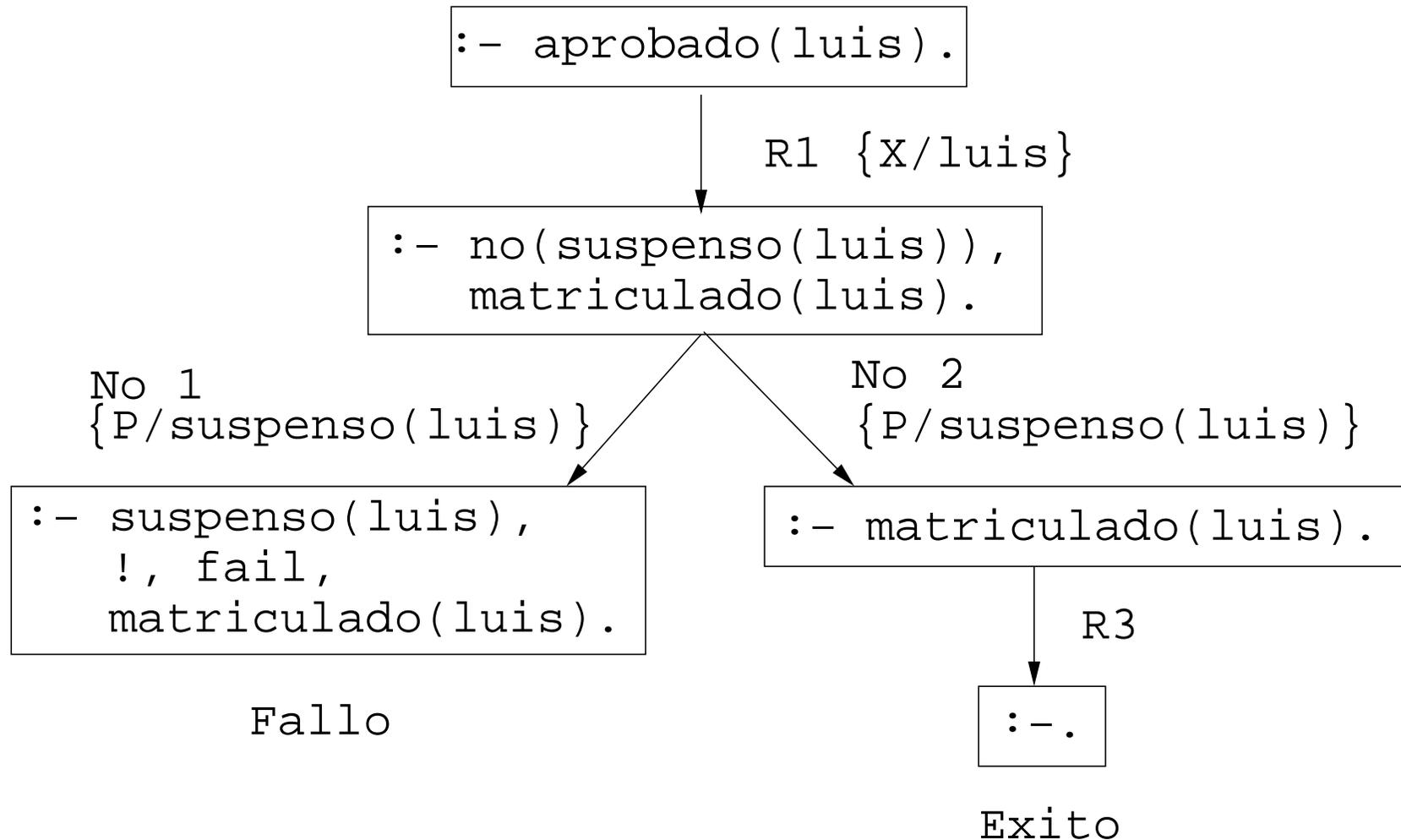
- Yes

- `?- aprobado(X).`

- No

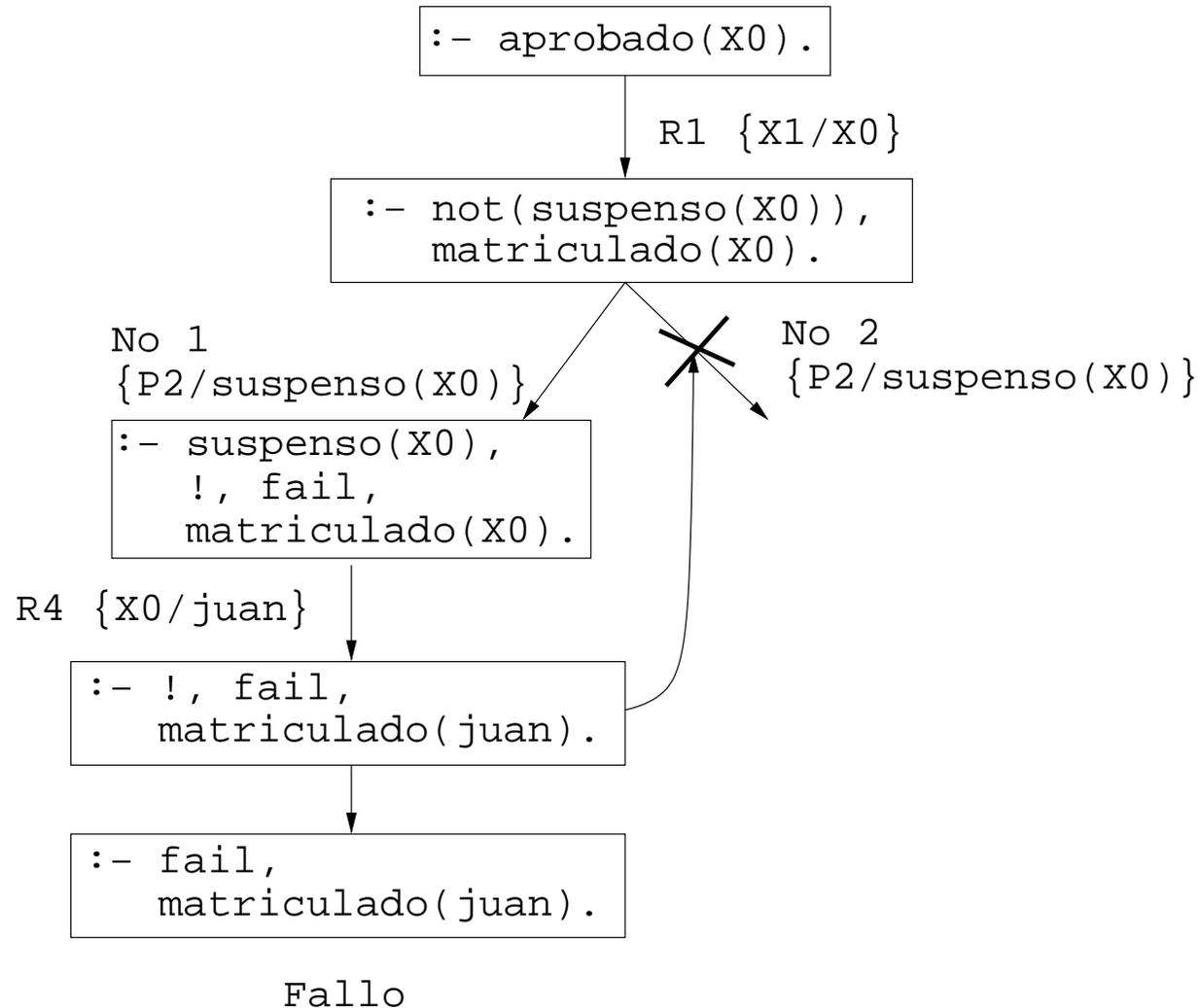
Definición de la negación como fallo

- Árbol de deducción de `?- aprobado(luis) .:`



Definición de la negación como fallo

- Árbol de deducción de `?- aprobado(X)`.



Definición de la negación como fallo

- Modificación del orden de los literales

- ▶ Programa:

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

- ▶ Consulta:

```
?- aprobado(X).
```

```
X = luis
```

```
Yes
```


Ejemplo de uso de la negación y el corte

- 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 uso de la negación y el corte

- Ejemplo de definición con `not` y con corte (cont.):

- ▶ 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) .
```

- ▶ 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 uso de la negación y el corte

- Ejemplo de definición con `not` y con corte (cont.):

- ▶ 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) .
```

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:

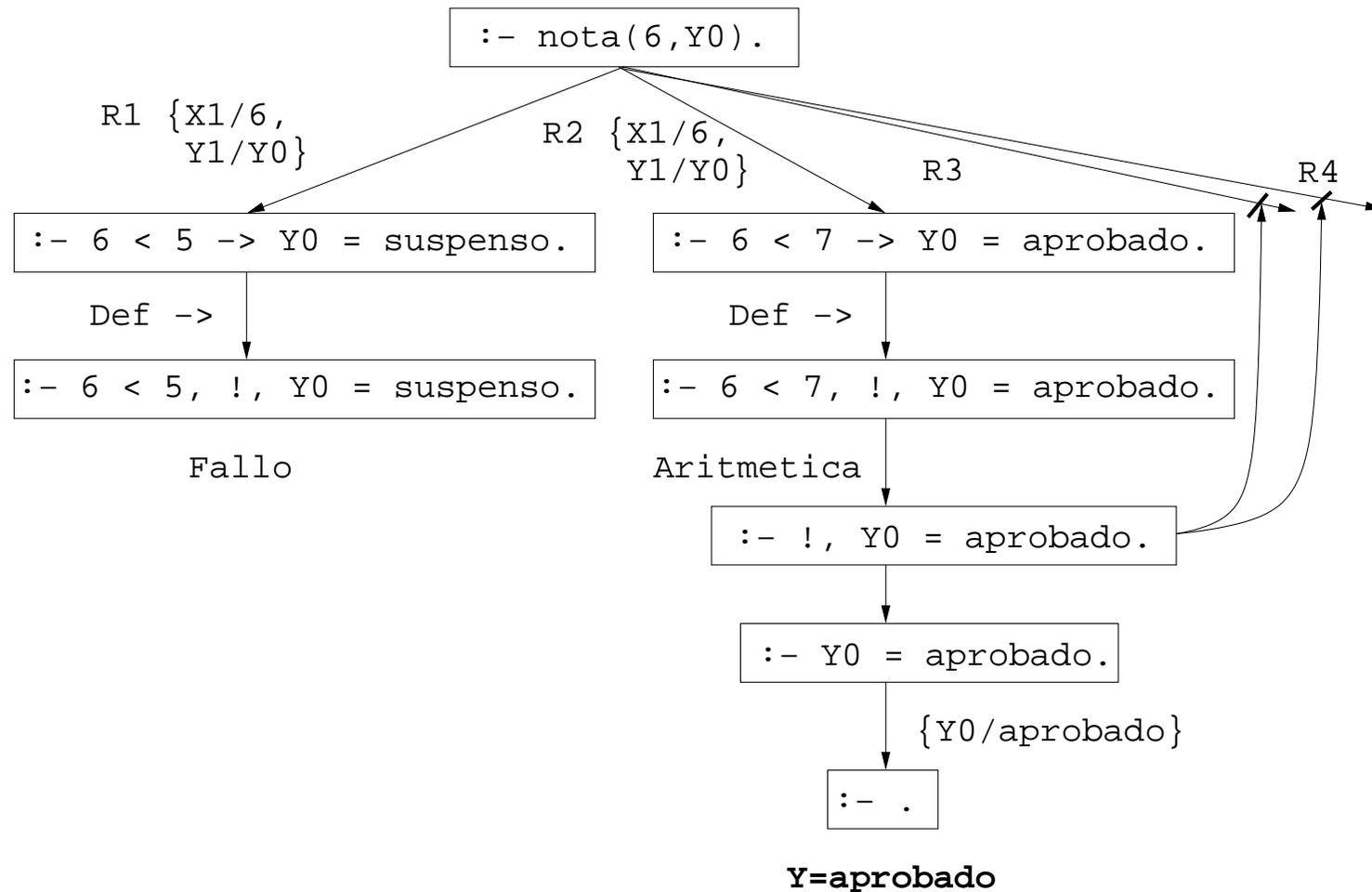
```
P -> Q :- P, !, Q.                   % Def. -> 1
```

```
P -> Q :- Q.                          % Def. -> 2
```

```
true .                                % Def. true
```

El condicional

- Árbol de deducción correspondiente a la pregunta `?- nota(6,Y)`.



Bibliografía

- J.A. Alonso y J. Borrego
Deducción automática (Vol. 1: Construcción lógica de sistemas lógicos)
(Ed. Kronos, 2002)
 - ▶ Cap. 2: Introducción a la programación lógica con Prolog, pp. 21–29
- I. Bratko *Prolog Programming for Artificial Intelligence (3 ed.)*
(Addison–Wesley, 2001)
 - ▶ Cap. 5: “Controlling backtracking”
- W.F. Clocksin y C.S. Mellish *Programming in Prolog (Fourth Edition)*
(Springer Verlag, 1994)
 - ▶ Cap. 4: “Backtracking and the cut”
- L. Sterling y E. Shapiro *The Art of Prolog (2nd Edition)* (The MIT Press, 1994)
 - ▶ Cap. 11: “Cuts and negation”