Examen de *Programación declarativa* del 15 de Febrero de 2007

José A. Alonso Jiménez

Grupo de Lógica Computacional
Dpto. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, 25 de febrero de 2007

Ejercicio 1 (2.5 puntos) *Se considera la siguiente gramática*

```
s --> [].
s --> i,s,d.
i --> [x].
d --> [y].
```

El programa Prolog correspondiente a la gramática es

```
s(A, A).
s(A, D) :- i(A, B), s(B, C), d(C, D).
i([x|A], A).
d([y|A], A).
```

Construir el árbol de resolución correspondiente al programa anterior con la pregunta

```
?- s([X, Y], []).
```

Explicar las respuestas de Prolog correspondiente al programa anterior con la pregunta

```
?- s(L, []).
```

Solución: El árbol de resolución se muestra en la figura 1. La respuesta de Prolog a la pregunta

```
?- s([X, Y], []).
```

es `X=x, Y=y`.

Las respuestas de Prolog a la pregunta

```
?- s([X, Y], []).
```

son `L=[]`, `L=[x,y]`, `L=[x,x,y,y]`, `L=[x,x,x,y,y,y]`, etc. Es decir las listas de la forma $[x^n, y^n]$ formada por n veces el elemento `x` y a continuación n veces el elemento `y`.

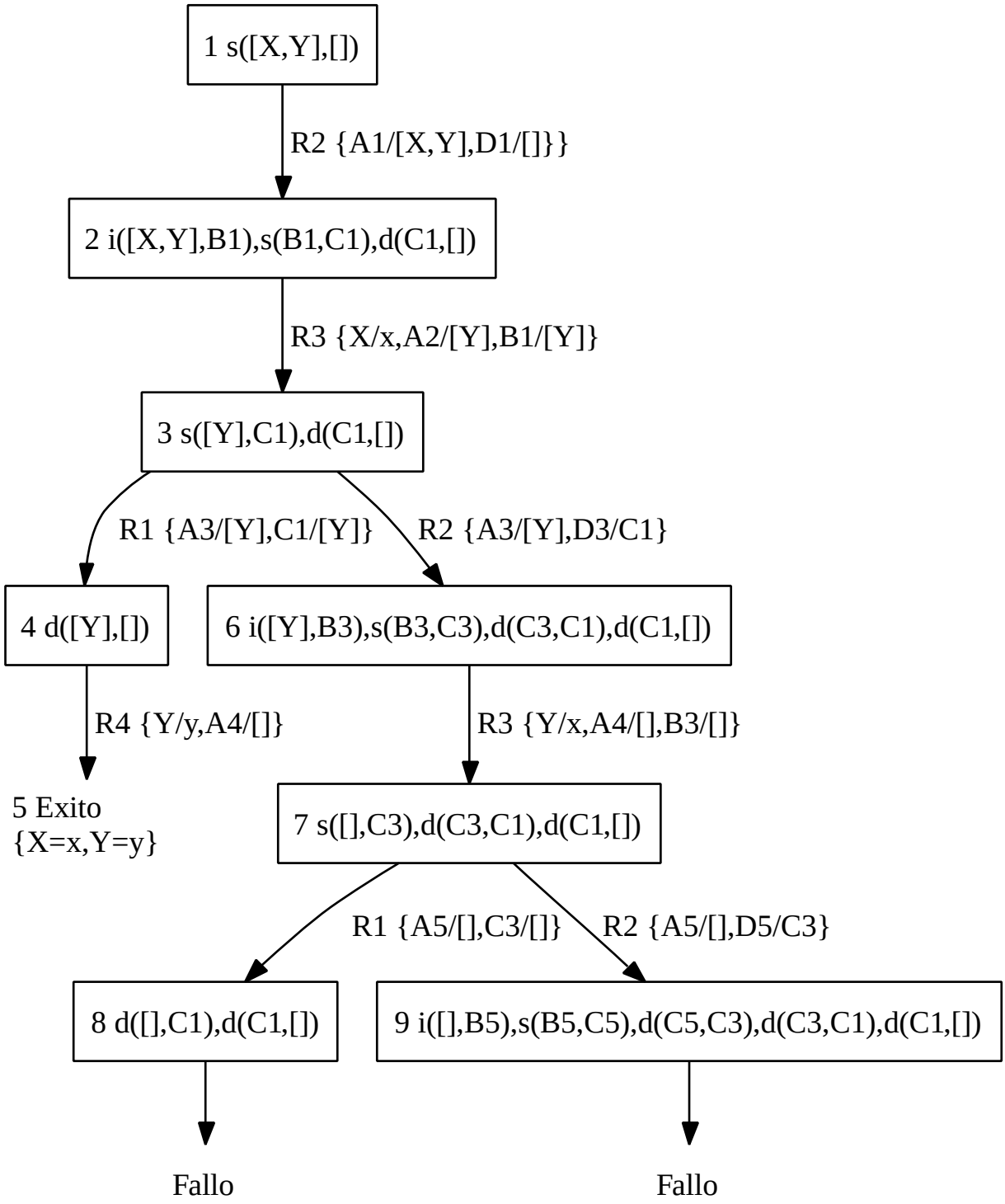

Figura 1: Árbol de resolución

Ejercicio 2 (2.5 puntos) *Determinar todas las respuestas de Prolog a cada una de las siguientes preguntas y por qué.*

1. `?- assert(p(a,1)), asserta(p(b,2)), assert(p(a,2)), findall(X,p(X,Y),L).`

2. `?- assert(p(a,1)), asserta(p(b,2)), assert(p(a,2)), setof(X,p(X,Y),L).`

3. `?- assert(p(a,1)), asserta(p(b,2)), assert(p(a,2)), setof(X,Y^p(X,Y),L).`

4. *`?- use_module(library(clpr)).`*
   `Yes`
   `?- {A >= 0, A+2 =< B, A+5 =< C, B+1 =< C}, minimize(C).`

5. `?- p(1,2) =.. [X,A,B],  p(3,4) =.. [X,C,D], T =.. [X,A+B,C+D].`

Solución:

1. La respuesta a

   ```
   ?- assert(p(a,1)), asserta(p(b,2)), assert(p(a,2)), findall(X,p(X,Y),L).
   ```

   es

   ```
   L = [b, a, a]
   ```

2. Las respuestas a

   ```
   ?- assert(p(a,1)), asserta(p(b,2)), assert(p(a,2)), setof(X,p(X,Y),L).
   ```

   son

   ```
   Y = 1
   L = [a] ;
   Y = 2
   L = [a, b] ;
   No
   ```

3. La respuesta a

   ```
   ?- assert(p(a,1)), asserta(p(b,2)), assert(p(a,2)), setof(X,Y^p(X,Y),L).
   ```

   es

   ```
   L = [a, b]
   ```

4. La respuesta a

   ```
   ?- use_module(library(clpr)).
   ?- {A >= 0, A+2 =< B, A+5 =< C, B+1 =< C}, minimize(C).
   ```

```
A = 0.0
{B=<4.0}
{B>=2.0}
C = 5.0
```

5. La respuesta a

```
?- p(1,2) =.. [X,A,B],  p(3,4) =.. [X,C,D], T =.. [X,A+B,C+D].
```

es

```
X = p
A = 1
B = 2
C = 3
D = 4
T = p(1+2, 3+4)
```

Ejercicio 3 (2.5 puntos) *Definir recursivamente la relación* `al_menos(+N,+P,+L)` *que se verifique si al menos* `N` *elementos de la lista* `L` *cumplen la propiedad* `P`*. Por ejemplo,*

```
?- al_menos(2,number,[a,3,b,c,3,d]).
Yes
?- al_menos(4,number,[a,3,b,c,3,d]).
No
?- al_menos(1,number,[a,3,b,c,3,d]).
Yes
```

Solución: La definición de `al_menos` es

```
al_menos(0,_,_) :- !.
al_menos(N,P,[X|L]) :-
   % N > 0,
   apply(P,[X]), !,
   N1 is N-1,
   al_menos(N1,P,L).
al_menos(N,P,[_|L]) :-
   % N > 0,
   % not(apply(P,[X])),
   al_menos(N,P,L).
```

Ejercicio 4 (2.5 puntos) *Se considera el proceso de sumar un número con el obtenido invirtiendo sus cifras hasta obtener un número capicúa. Por ejemplo, dado el número 37 se obtiene*

```
  37 +  73 = 110
 110 + 011 = 121
```

Luego, las transformaciones de 37 son [37, 110, 121].

1. *Definir la relación* `transformaciones(+N,-L)` *que se verifique si* `L` *es la lista de las transformaciones del número* `N`*. Por ejemplo,*

```
?- transformaciones(37,L).
L = [37, 110, 121]
?- transformaciones(59,L).
L = [59, 154, 605, 1111]
```

2. *Definir la relación* `número_transformaciones(+N,-X)` *que se verifique si* `X` *es el número de transformaciones del número* `N`*. Por ejemplo,*

```
?- número_transformaciones(37,X).
X = 3
?- número_transformaciones(59,X).
X = 4
```

3. *Definir la relación* `números_con_mayor_número_de_transformaciones(+N,+M,-L)` *que se verifique si* `L` *es la lista de los números entre* `N` *y* `M` *con mayor número de transformaciones. Por ejemplo,*

```
?- números_con_mayor_número_de_transformaciones(10,30,L).
L = [19, 28]
```

Solución:

1. La definición de `transformaciones` es

```
transformaciones(N,[N]) :-
   capicúa(N), !.
transformaciones(N,[N|L]):-
   % not(capicúa(N)),
   simétrico(N,M),
   N1 is N+M,
   transformaciones(N1,L).
```

donde `capicúa(+N)` se verifica si `N` es capicúa y está definida por

```
capicúa(N) :-
   name(N,L),
   reverse(L,L).
```

y `simétrico(+N1,-N2)` se verifica si `N2` es el simétrico de `N1` y está definida por

```
simétrico(N1,N2) :-
   name(N1,L1),
   reverse(L1,L2),
   name(N2,L2).
```

2. La definición de `número_transformaciones` es

```
número_transformaciones(N,A) :-
   transformaciones(N,L),
   length(L,A).
```

3. La definición de `números_con_mayor_número_de_transformaciones` es

```
números_con_mayor_número_de_transformaciones(N1,N2,L) :-
   findall(X-NT, (between(N1,N2,X), número_transformaciones(X,NT)), L1),
   member(_-N,L1),
   not((member(_-M,L1), N<M)), !,
   findall(X,member(X-N,L1),L).
```