Programación Declarativa

3er Curso, Grado en Ingeniería en Informática
Universidad Rey Juan Carlos

Programación Lógica

Construcción paso a paso de un Árbol de Resolución con cortes y negaciones

---

Considere el programa Prolog

```
p(A,B) :- q(A), !, r(B,A).          |          q(1).
r(A,B) :- q(A).                     |          q(2).
t(2).                               |          q(3).
```

y la consulta `?- p(Y,X), \+ t(X).` (cuyo objetivo es averiguar si, dado el conocimiento expresado en el programa anterior, existen objetos `Y` y `X` tales que el primero está relacionado con el segundo mediante la relación "`p`" de forma que el segundo, además, no cumple la propiedad "`t`").

En lo que sigue se detalla la construcción, paso a paso, del árbol de Resolución correspondiente a la consulta anterior y se explica cómo, a partir de él, se deduce qué respuesta(s) ofrecería Prolog ante esa consulta, y en qué orden facilitaría dichas respuestas.

Los árboles de Resolución de Prolog, descritos en el apartado 3.2. del tema PL-2 de los apuntes, se construyen en profundidad por la izquierda y con retroceso. Como el programa y la consulta dados incorporan tanto un corte (`!`) como una negación (`\+`), es necesario conocer además los efectos que estos dos predicados tienen sobre la construcción de los árboles de Resolución. Ambos predicados están descritos, respectivamente, en los apartados 8 y 1 de los temas PL-2 y PL-3.

Construcción del Árbol de Resolución

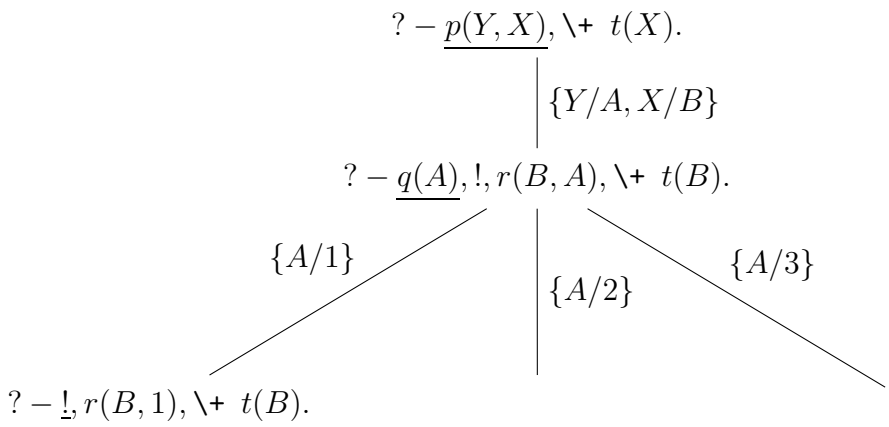
La raíz del árbol no es otra cosa que la consulta, que tendrá tantos hijos como cláusulas haya en el programa cuya cabeza unifique con el predicado más a la izquierda de la consulta, es decir, `p(Y,X)` (Prolog siempre resuelve sus objetivos empezando por el sub-objetivo de más a la izquierda). Sólo hay una cláusula en el programa, la primera, cuya cabeza unifica con `p(Y,X)`, y el nodo hijo será la cláusula resolvente obtenida al aplicar la Regla de Resolución entre la consulta y esa cláusula. Un unificador de máxima generalidad (u.m.g.) que permite hacer lo anterior es $\{Y/A, X/B\}$, obteniéndose la cláusula resolvente `?- q(A), !, r(B,A), \+ t(B)` (este u.m.g. no es el único posible, se podría haber elegido, por ejemplo, $\{Y/A, B/X\}$, en cuyo caso la cláusula resolvente sería `?- q(A), !, r(X,A), \+ t(X)`, pero los resultados finales serían los mismos). Por lo tanto, el primer nivel del árbol de Resolución es el siguiente:

$$? - \underline{p(Y,X)}, \texttt{\textbackslash+}\ t(X).$$

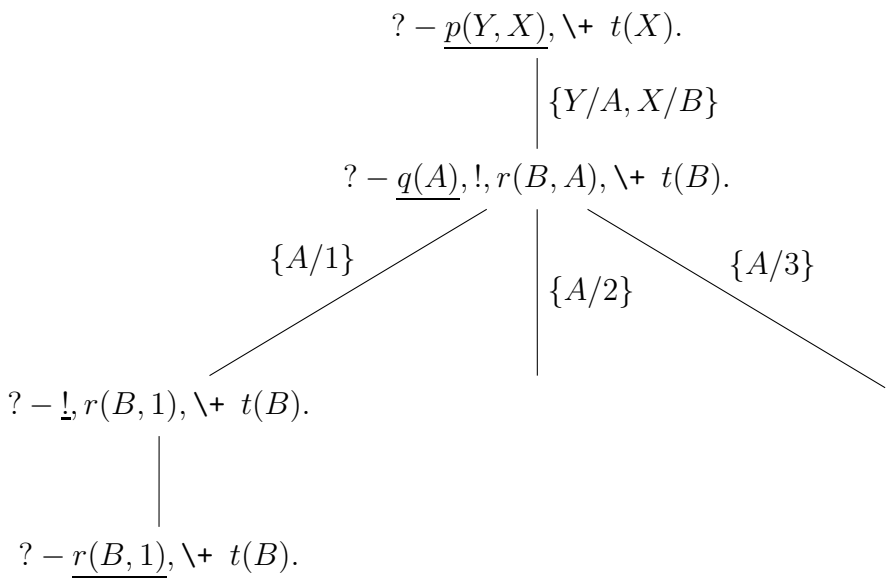
$$\Big| \{Y/A, X/B\}$$

$$? - \underline{q(A)}, !, r(B,A), \texttt{\textbackslash+}\ t(B).$$

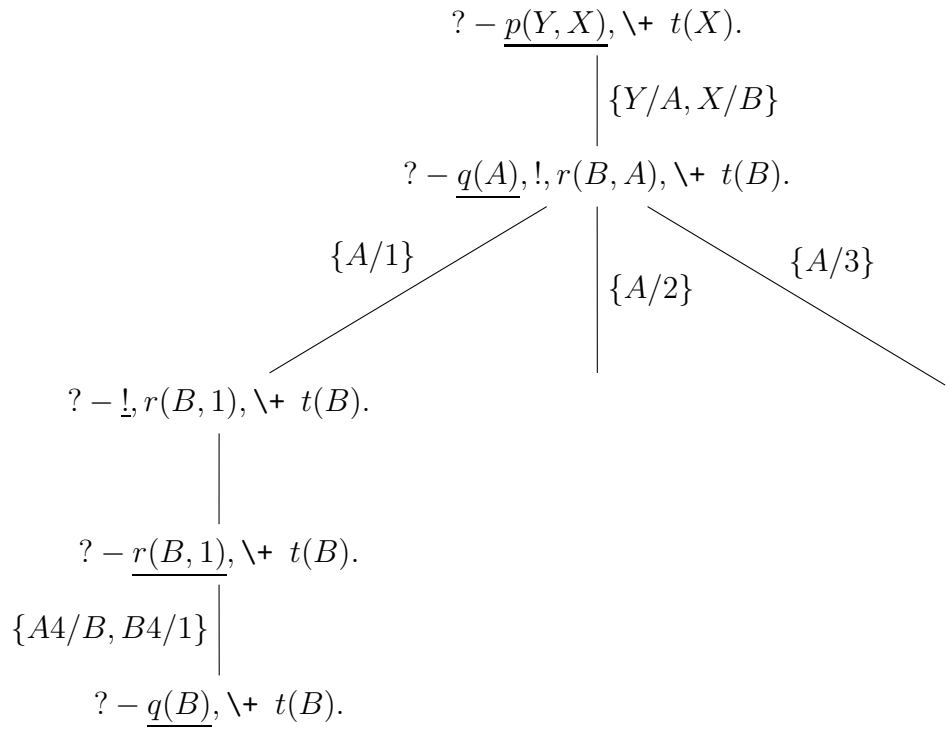
En el siguiente nivel del árbol habrá tres nodos, puesto que el objetivo de más a la izquierda, `q(A)`, unifica con la cabeza de las tres últimas cláusulas del programa, mediante u.m.g.'s $\{A/1\}$, $\{A/2\}$ y $\{A/3\}$, respectivamente. Como Prolog desarrolla el árbol en profundidad por la izquierda y es posible que algunas de las ramas de la derecha acaben siendo podadas, en este momento sólo se calcula el nodo correspondiente a la rama de más a la izquierda (aunque se dejan trazadas las demás ramas por si hubiese que desarrollarlas al retroceder en la construcción del árbol). El árbol resultante es el siguiente:



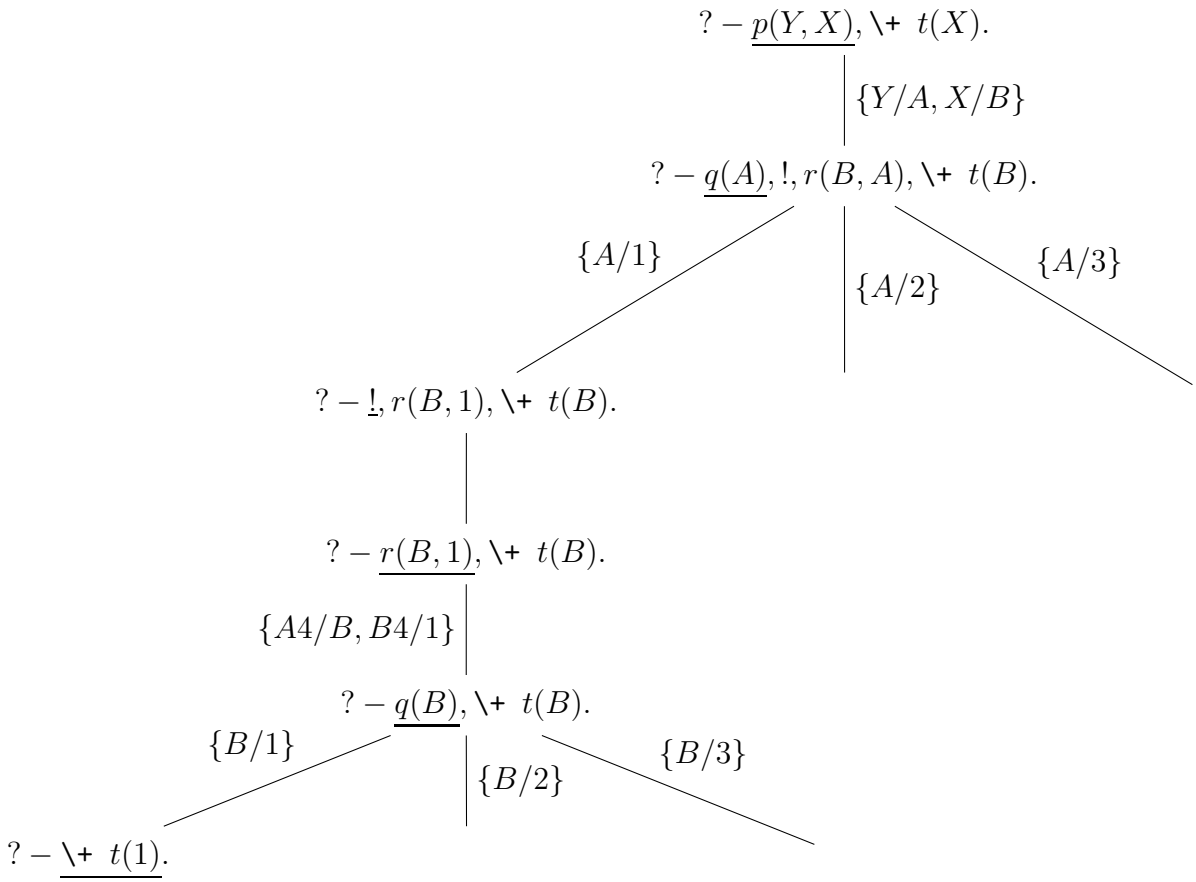
Siguiendo el desarrollo del árbol en profundidad por la izquierda, el nuevo nodo a expandir es `?- !, r(B,1), \+ t(B)`, que tiene un único hijo, igual a él pero sin el corte inicial (el corte es un predicado que es siempre cierto; sus efectos se producirán más adelante, al retroceder en la construcción del árbol).



De forma similar, se trata ahora de expandir el nodo `?- r(B,1), \+ t(B)`, viendo con qué cabezas de las cláusulas del programa unifica el sub-objetivo de más a la izquierda, `r(B,1)`. Éste sólo unifica con la segunda cláusula, que habrá que renombrar puesto que hay conflicto de variables (la variable `B` aparece tanto en la regla del programa como en `r(B,1)`). Como estamos en el cuarto nivel del árbol, renombramos la cláusula del programa como `r(A4,B4) :- q(A4)`, y mediante u.m.g. $\{A4/B, B4/1\}$ se obtiene la cláusula resolvente `?- q(B), \+ t(B)`:



El sub-objetivo de más a la izquierda, `q(B)`, unifica con las tres últimas cláusulas del programa, por lo que el nodo tendrá tres hijos, aunque, como siempre, por el momento basta con calcular el de más a la izquierda, que será `?- \+ t(1)` ya que el u.m.g. es $\{B/1\}$.

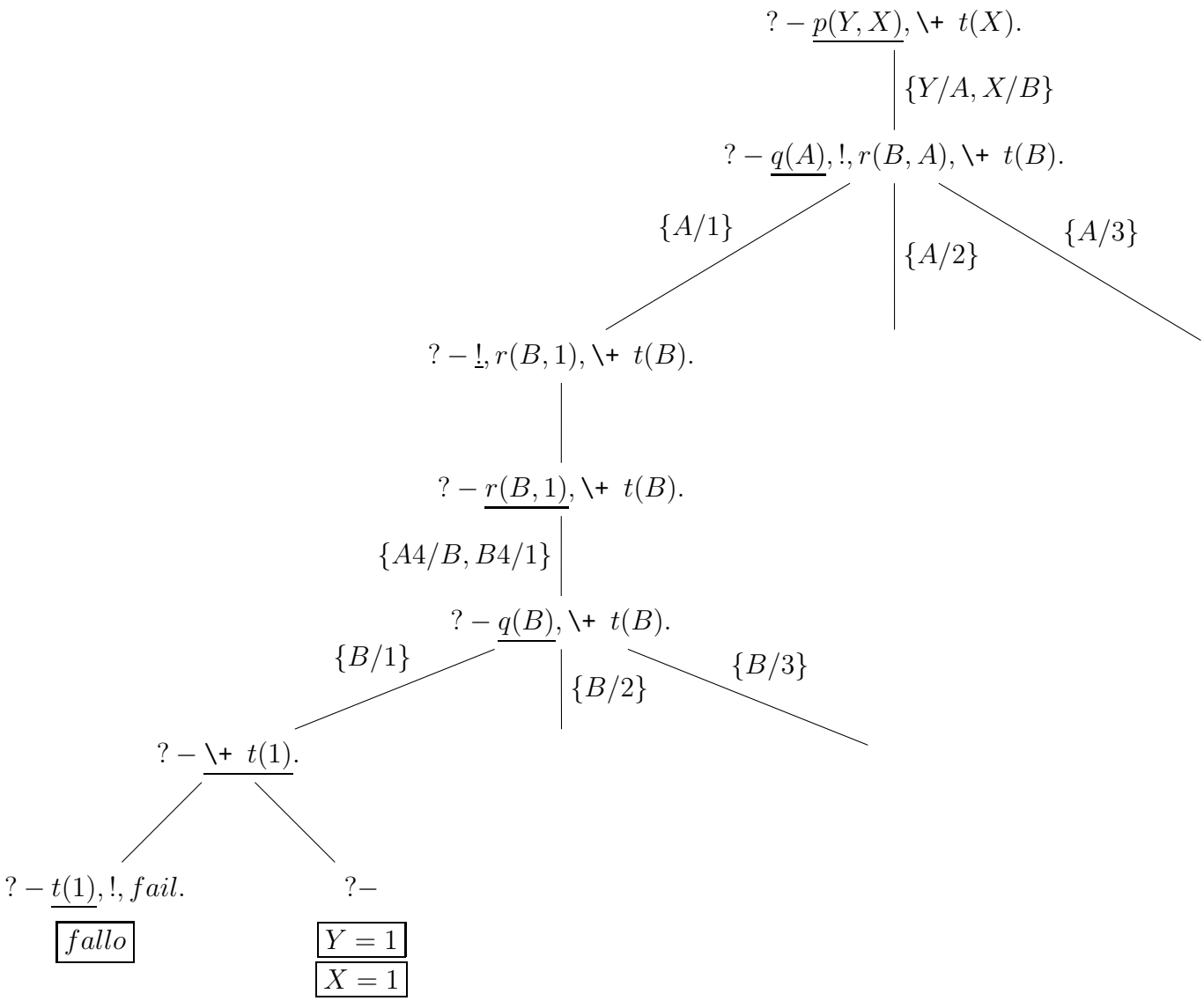


Hay que desarrollar ahora el nodo `?- \+ t(1)`, que, al ser un objetivo negado, tendrá dos hijos (ver tema PL-3), de los cuales por el momento solo se calcula el de la izquierda, que es `?- t(1), !, fail.` Este último resulta ser un nodo fallo, puesto que su primer sub-objetivo, `t(1)`, no unifica con la cabeza de ninguna cláusula del programa. Al llegar a un nodo fallo, Prolog retrocede para seguir desarrollando el árbol en profundidad. El siguiente hijo del nodo padre `?- \+ t(1)` es directamente un nodo éxito, por lo que Prolog calcula la solución asociada aplicando a las variables de la consulta, `Y` y `X`, todos los u.m.g.'s de la rama, en orden, empezando desde la raíz:

$$Y\{Y/A, X/B\}(\{A/1\}\{A4/B, B4/1\}\{B/1\}) = A\{A/1\}(\{A4/B, B4/1\}\{B/1\}) = 1$$

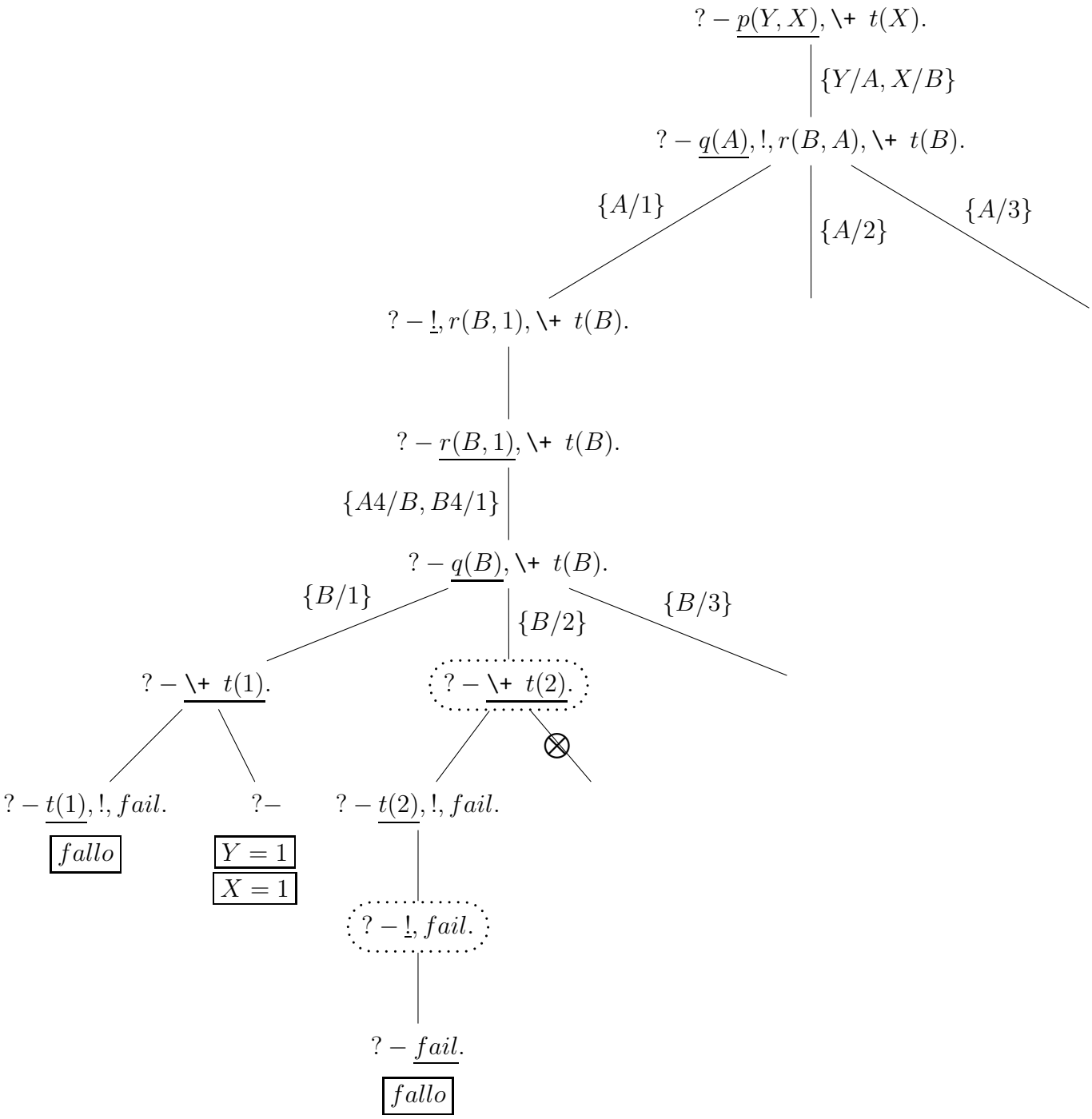
$$\begin{aligned} X\{Y/A, X/B\}(\{A/1\}\{A4/B, B4/1\}\{B/1\}) &= B\{A/1\}(\{A4/B, B4/1\}\{B/1\}) \\ &= B\{A4/B, B4/1\}(\{B/1\}) = B\{B/1\} = 1 \end{aligned}$$

El árbol resultante hasta el momento es por lo tanto el siguiente:

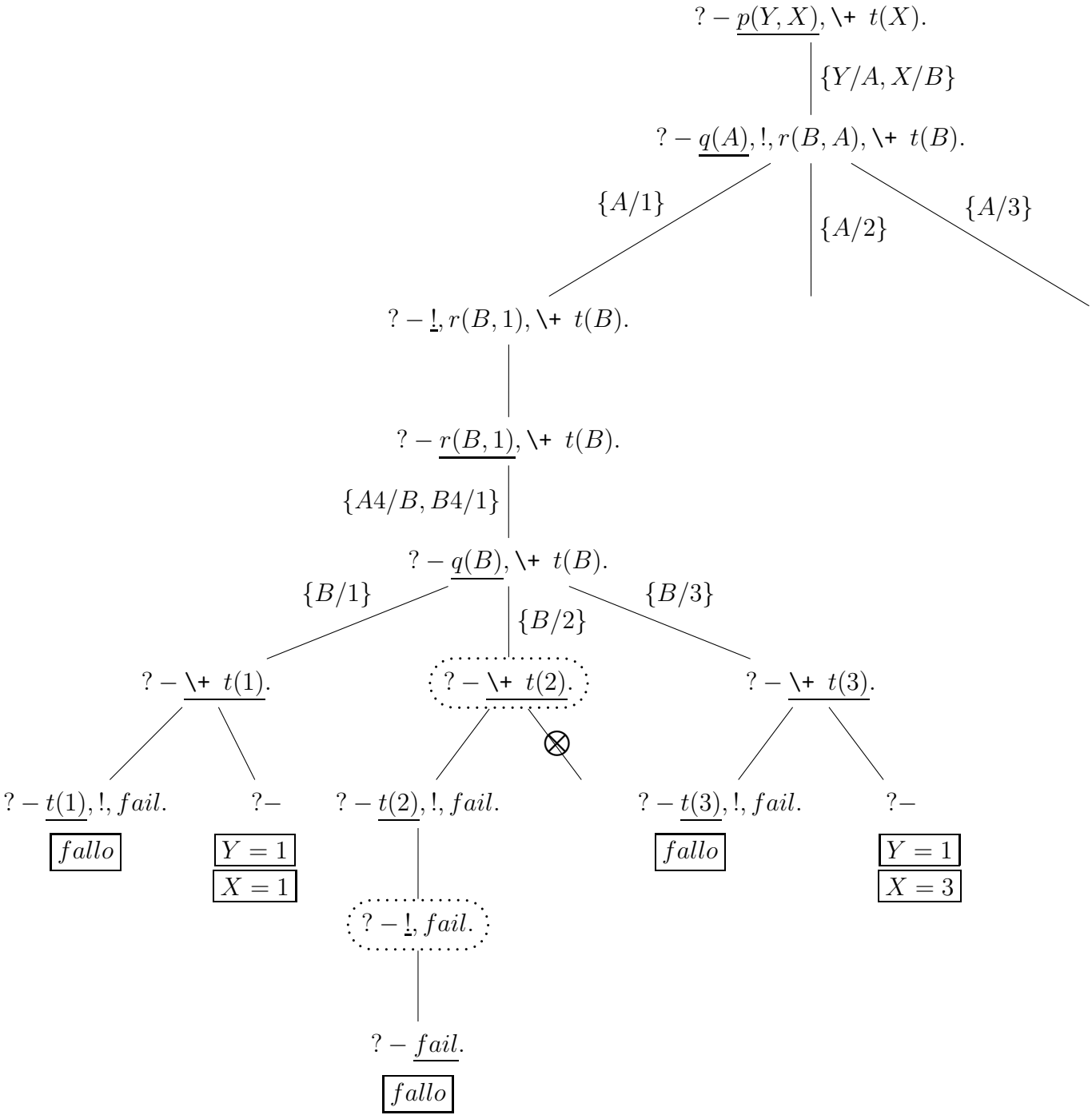


Al seguir desarrollando el árbol en profundidad, Prolog retrocede hasta el siguiente nodo pendiente de cálculo, el segundo hijo del nodo `?- q(B), \+ t(B)`, que se obtiene me-

diante la cláusula `q(2)` y la sustitución $\{B/2\}$, dando lugar al nodo `?- \+ t(2).` Este último, al empezar por un objetivo negado, tendrá dos hijos, siendo el de la izquierda `?- t(2), !, fail.` Como `t(2)` es cierto, este nodo tiene como único hijo `?- !, fail` que a su vez, al ser el corte siempre cierto, da lugar al nodo `?- fail`, que falla (el predicado predefinido `fail` siempre falla). El fallo anterior hace que Prolog retroceda por la rama, y como al hacerlo encuentra un nodo empezando por corte, en el retroceso poda todas las ramas comprendidas entre el nodo que empieza por corte y su ancestro más cercano que no contiene corte (ambos marcados en el árbol a continuación mediante una elipse intermitente). En este caso no hay más que una rama por podar, la marcada en el árbol mediante el símbolo $\bigotimes$.

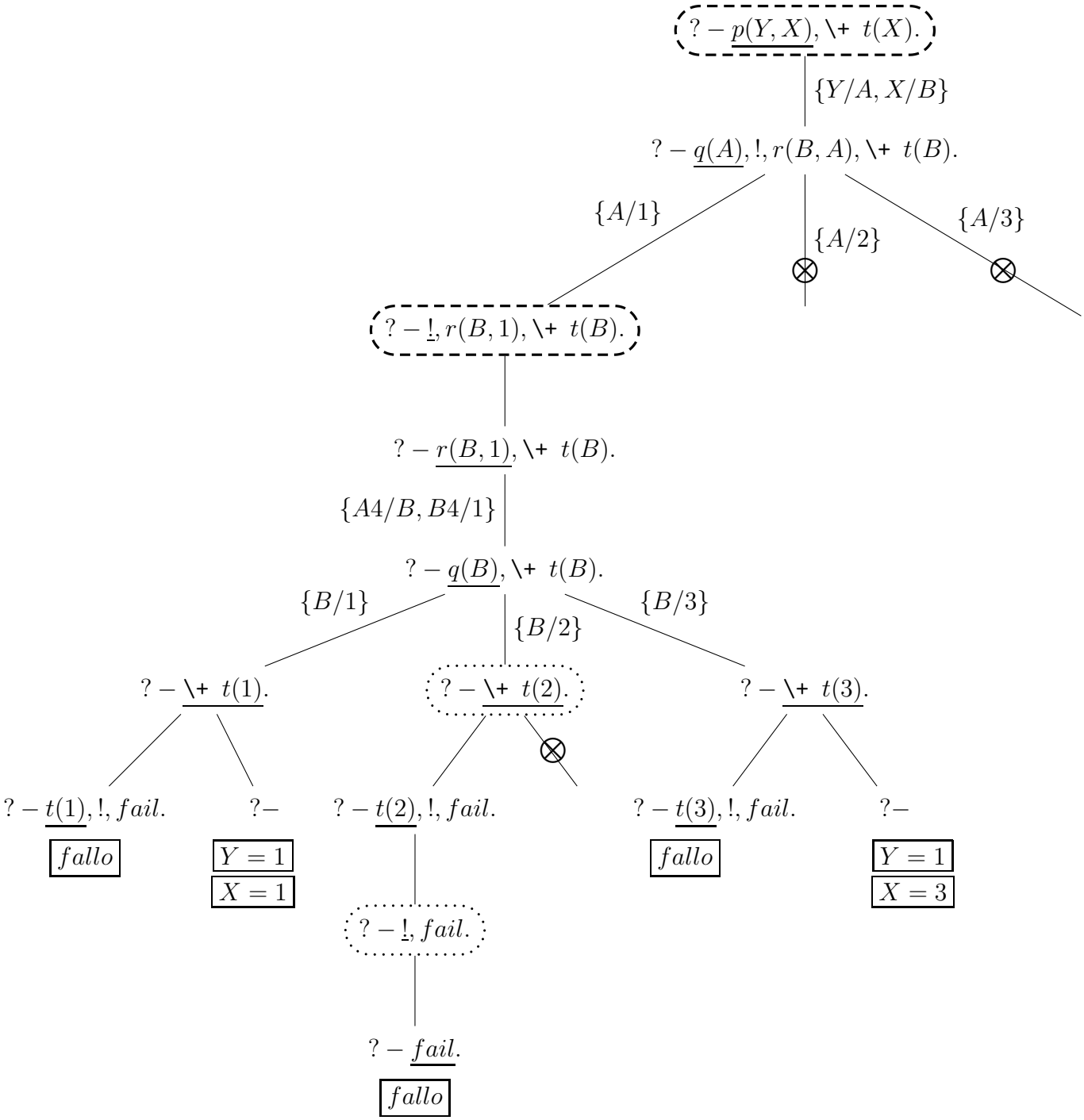


El siguiente nodo a expandir es el tercer hijo del nodo `?- q(B), \+ t(B)`, que resulta análogo al primer hijo, dando lugar al siguiente árbol:



Una vez terminada la rama en la que ha encontrado la solución `Y=1, X=3`, Prolog retrocede en busca del siguiente nodo pendiente de explorar. En el camino hacia arriba acaba llegando al nodo `?- !, r(B,1), \+ t(B)`, que empieza por corte, por lo que tiene que buscar el ancestro más cercano que no contiene el corte, que resulta ser la raíz del árbol, y podar todas las ramas que salgan a la derecha entre uno y otro. En el árbol que se dibuja a continuación los dos nodos que determinan la poda aparecen marcados con una elipse intermitente, mientras que las ramas podadas se marcan con el símbolo $\bigotimes$.

Ya no queda ningún nodo por expandir, por lo que la construcción del Árbol de Resolución termina. El árbol completo obtenido es el siguiente:



Respuestas de Prolog

Ante una consulta, Prolog construye el árbol de Resolución, en profundidad, y facilita las soluciones según las va encontrando. Por lo tanto, ante la consulta dada, y a la vista del árbol anterior, la respuesta de Prolog será la siguiente:

```
Y=1, X=1 ;    % primera solución encontrada
Y=1, X=3 ;    % segunda solución encontrada
false (o no)  % indicando que no quedan más soluciones
```