© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

PROLOG

Programmation Logique

Yannick Estève
Université du Maine

Cours 2

- Théorie
  - Unification
  - Unification en Prolog
  - Recherche de preuve
- Exercices
  - Exercices du chapitre 2 (LPN)
  - Travaux pratiques

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

But de cette séance

- Traiter de l'**unification** en Prolog
  - Montrer comme l'unification de Prolog diffère de l'unification standard

- Expliquer la stratégie de recherche que Prolog utilise lorsqu'il tente de déduire de nouvelles informations à partir d'anciennes à l'aide du *modus ponens*

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Unification

- Rappel : exemple précédent, pour lequel nous avons dit que Prolog unifie

  femme(X)

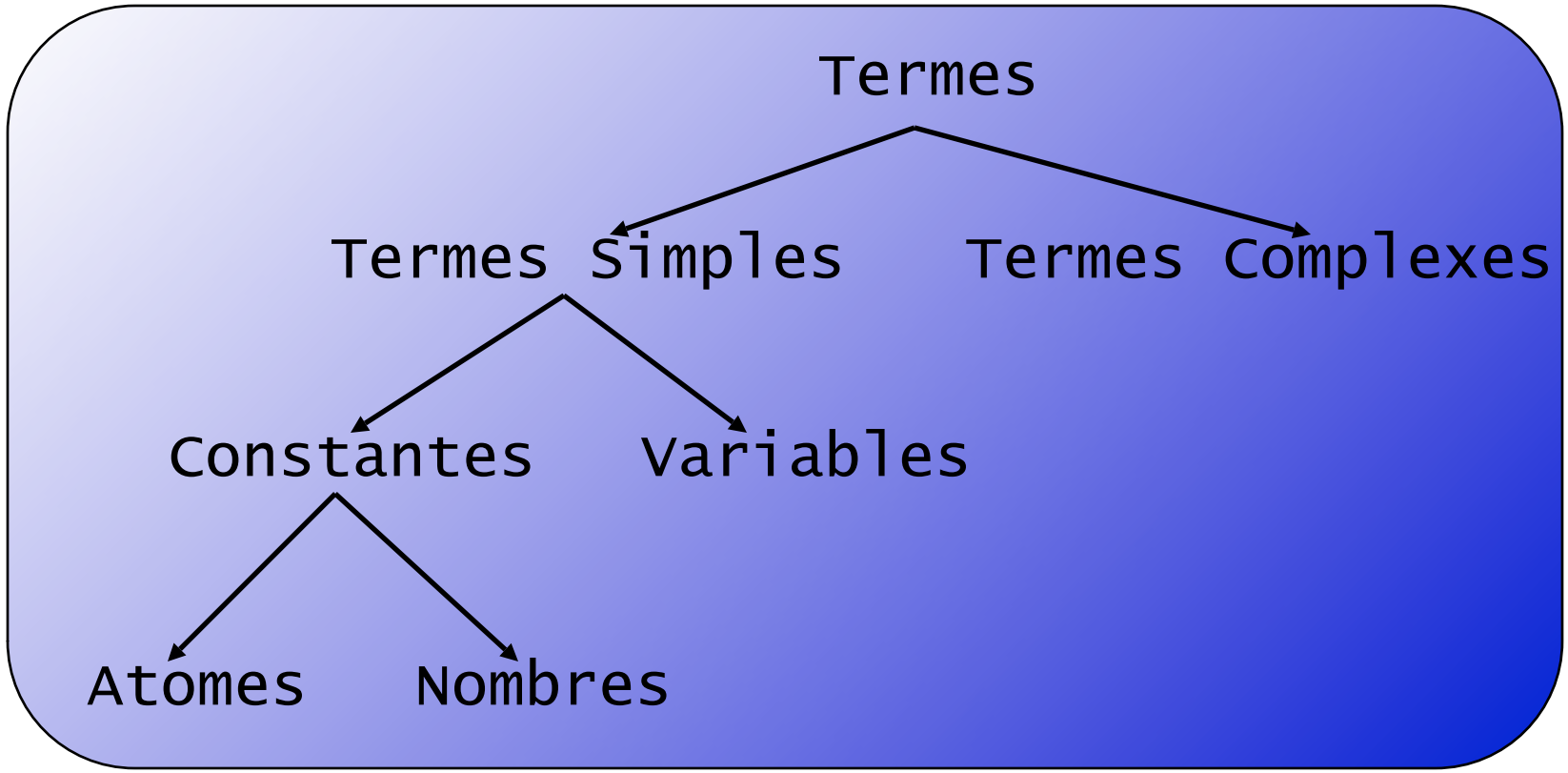
  avec

  femme(mia)

  en instanciant ta variable **X** avec l'atome **mia**.

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Rappel : termes de Prolog



© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Unification

- Définition de départ :
  - Deux termes sont unifiés si ils sont le même terme, ou si ils contiennent des variables qui peuvent uniformément s'instancier avec des termes de telle manières que les termes résultants sont égaux

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Unification

- Ceci signifie que :
  - **mia** and **mia** s'unifient
  - **42** and **42** s'unifient
  - **woman(mia)** et **woman(mia)** s'unifient
- Ceci signifie également que :
  - **vincent** et **mia** ne s'unifient pas
  - **woman(mia)** et **woman(jody)** ne s'unifient pas

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Unification

- Que pouvez-vous dire des termes :
  - **mia** et **X**

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Unification

- Que pouvez-vous dire des termes :
  - **mia** et **X**
  - **femme(Z)** et **femme(mia)**

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Unification

- Que pouvez-vous dire des termes :
  - **mia** et **X**
  - **femme(Z)** et **femme(mia)**
  - **aime(mia,X)** et **aime(X,vincent)**

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Instanciations

- Quand Prolog unifie deux termes, il effectue toutes les instanciations nécessaires, afin de rendre les termes égaux

- Ceci fait que l'unification est un mécanisme de programmation puissant

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Définition révisée 1/3

1. Si T_1 and T_2 sont des constantes, alors T_1 and T_2 s'unifient si il sont le même atome ou le même nombre.

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Définition révisée 2/3

1. Si T_1 and T_2 sont des constantes, alors T_1 and T_2 s'unifient si il sont le même atome ou le même nombre.
- Si T_1 est une variable et T_2 est n'importe quel type de terme, alors T_1 et T_2 s'unifient, et T_1 est instancié par T_2. (et vice versa)

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Définition révisée 3/3

1. Si T_1 and T_2 sont des constantes, alors T_1 and T_2 s'unifient si il sont le même atome ou le même nombre
- Si T_1 est une variable et T_2 est n'importe quel type de terme, alors T_1 et T_2 s'unifient, et T_1 est instancié par T_2. (et vice versa)
- Si T_1 et T_2 sont des termes complexes, ils s'unifient si :
  - Ils ont les mêmes foncteur et arité, et
  - Tous leur arguments s'unifient, et
  - Les instanciations de variables sont compatibles

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Unification Prolog : =/2

```
?- mia = mia.
yes
?-
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Unification Prolog : =/2

```
?- mia = mia.
yes
?- mia = vincent.
no
?-
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Unification Prolog : =/2

```
?- mia = X.
X=mia
yes
?-
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Comment Prolog répondra ?

?- X=mia, X=vincent.

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Comment Prolog répondra ?

?- X=mia, X=vincent.

no

?-

Pourquoi ? Après avoir travaillé avec la premier but, Prolog a instancié X avec **mia**, et ne peut donc pas l'unifier aussi avec **vincent :** le second but échoue

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Exemple avec des termes

```
?- k(s(g),Y) = k(X,t(k)).
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Example avec des termes

```
?- k(s(g),Y) = k(X,t(k)).
X=s(g)
Y=t(k)
yes
?-
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Example avec des termes

?- k(s(g),t(k)) = k(X,t(Y)).

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Example avec des termes

?- k(s(g),t(k)) = k(X,t(Y)).

X=s(g)

Y=k

yes

?-

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Un dernier exemple

?- aime(X,X) = aime(marsellus,mia).

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Prolog et unification

- Prolog n'utilise pas un algorithme d'unification standard
- Considérons la requête suivante :

  ?- pere(X) = X.

- Est-ce que ces termes s'unifient ?

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Termes Infinis

?- pere(X) = X.

X=pere(pere(pere(pere(pere(pere(pere(pere
(pere(pere(pere(pere(pere(pere(pere(pere
(pere(pere(pere(pere(pere(pere(pere(pere
(pere(pere(pere(pere(pere(pere(pere(pere
(pere(pere(pere(pere(pere(pere(pere(pere
(pere(pere(pere(pere(pere(pere(pere(pere
(pere(pere(pere(pere(pere(pere(pere(

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Termes Infinis

```
?- pere(X) = X.
X=pere(pere(pere(…))))
yes
?-
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Test d'occurrence

- Un algorithme d'unification standard effectue un test d'occurrence
- Si on lui demande d'unifier une variable avec un autre terme, il vérifiera d'abord la présence de cette variable dans ce terme
- Il existe un prédicat prédifini en Prolog:

```
?- unify_with_occurs_check(pere(X), X).
no
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Test d'occurrence

- L'algorithme d'unification standard est pessimiste : le test d'occurrence est systématique
- Prolog est optimiste : aucun test d'occurrence n'est effectué (efficacité)

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Programmation par Unification

```
vertical( ligne(point(X,Y),
                point(X,Z))).

horizontal(ligne(point(X,Y),
                 point(Z,Y))).
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Programmation par Unification

```
vertical(ligne(point(X,Y),
               point(X,Z))).

horizontal(ligne(point(X,Y),
                 point(Z,Y))).
```

```
?-
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Programmation par Unification

```
vertical(ligne(point(X,Y),
                    point(X,Z))).

horizontal(ligne(point(X,Y),
                      point(Z,Y))).
```

```
?- vertical(ligne(point(1,1),point(1,3))).
yes
?-
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Programmation par Unification

```
vertical(ligne(point(X,Y),
                    point(X,Z))).

horizontal(ligne(point(X,Y),
                        point(Z,Y))).
```

```
?- vertical(ligne(point(1,1),point(1,3))).
yes
?- vertical(ligne(point(1,1),point(3,2))).
no
?-
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Programmation par Unification

```
vertical(ligne(point(X,Y),
               point(X,Z))).

horizontal(ligne(point(X,Y),
                 point(Z,Y))).
```

```
?- horizontal(ligne(point(1,1),point(1,Y))).
Y = 1;
no
?-
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Programmation par Unification

```
vertical(ligne(point(X,Y),
                    point(X,Z))).

horizontal(ligne(point(X,Y),
                        point(Z,Y))).
```

```
?- horizontal(ligne(point(2,3),Point)).
Point = point(_554,3);
no
?-
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Exercice: unification

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Recherche de Preuve

- Nous savons ce qu'est l'unification, nous pouvons comprendre comment Prolog recherche dans sa base de connaissances si une requête peut être satisfaite (et comment elle le peut).
- En d'autres termes : nous sommes prêts à comprendre la notion de recherche de preuve

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Exemple

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

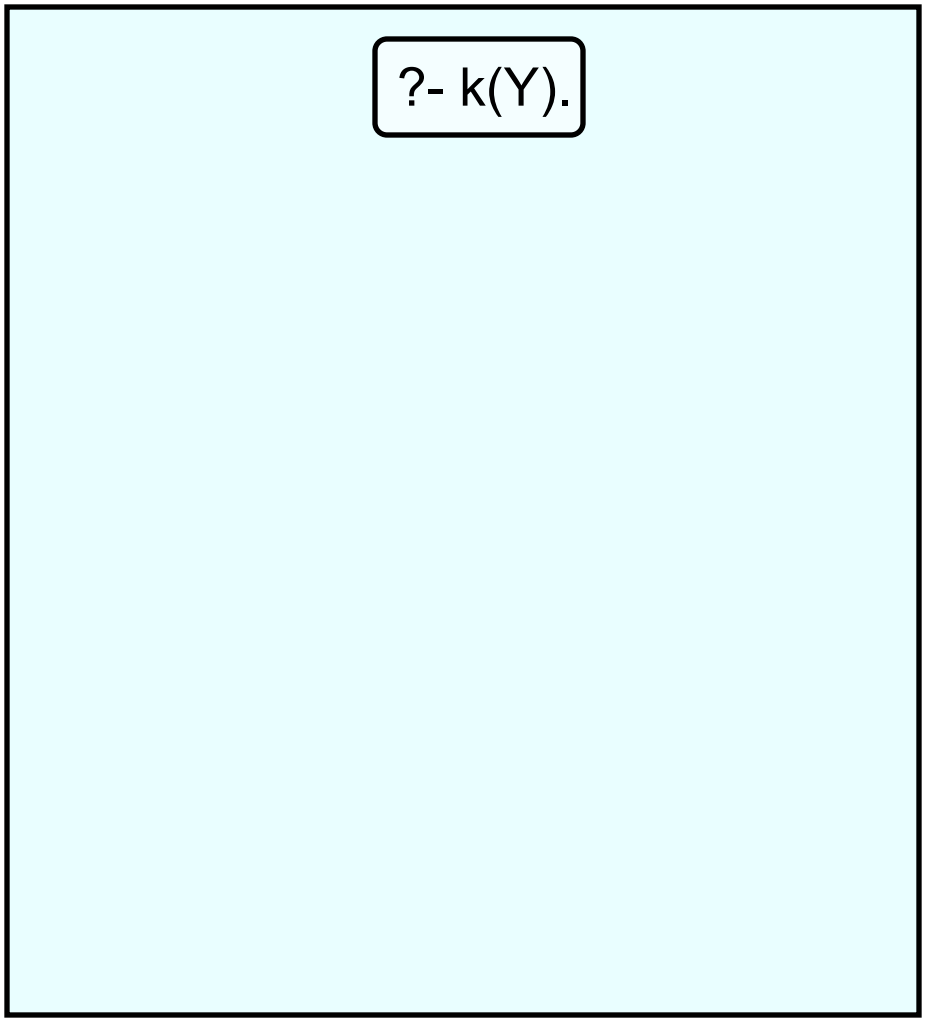
```
?- k(Y).
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Exemple: arbre de recherche

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

```
?- k(Y).
```

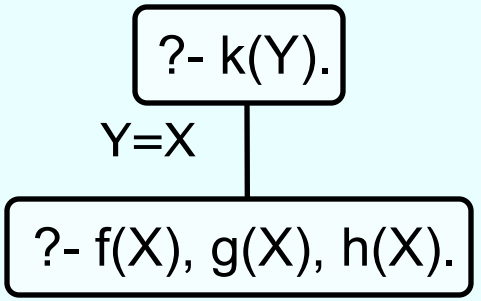


© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Exemple: arbre de recherche

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

```
?- k(Y).
```

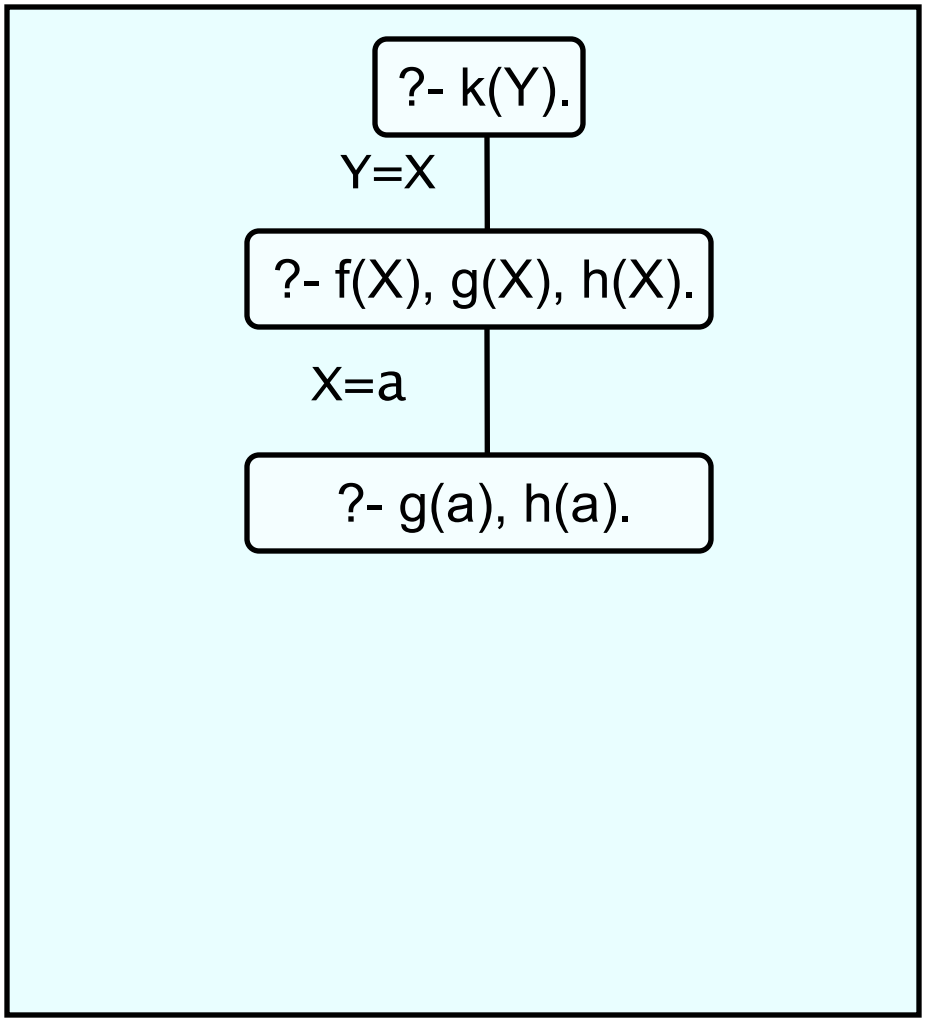


© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Exemple: arbre de recherche

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

```
?- k(Y).
```

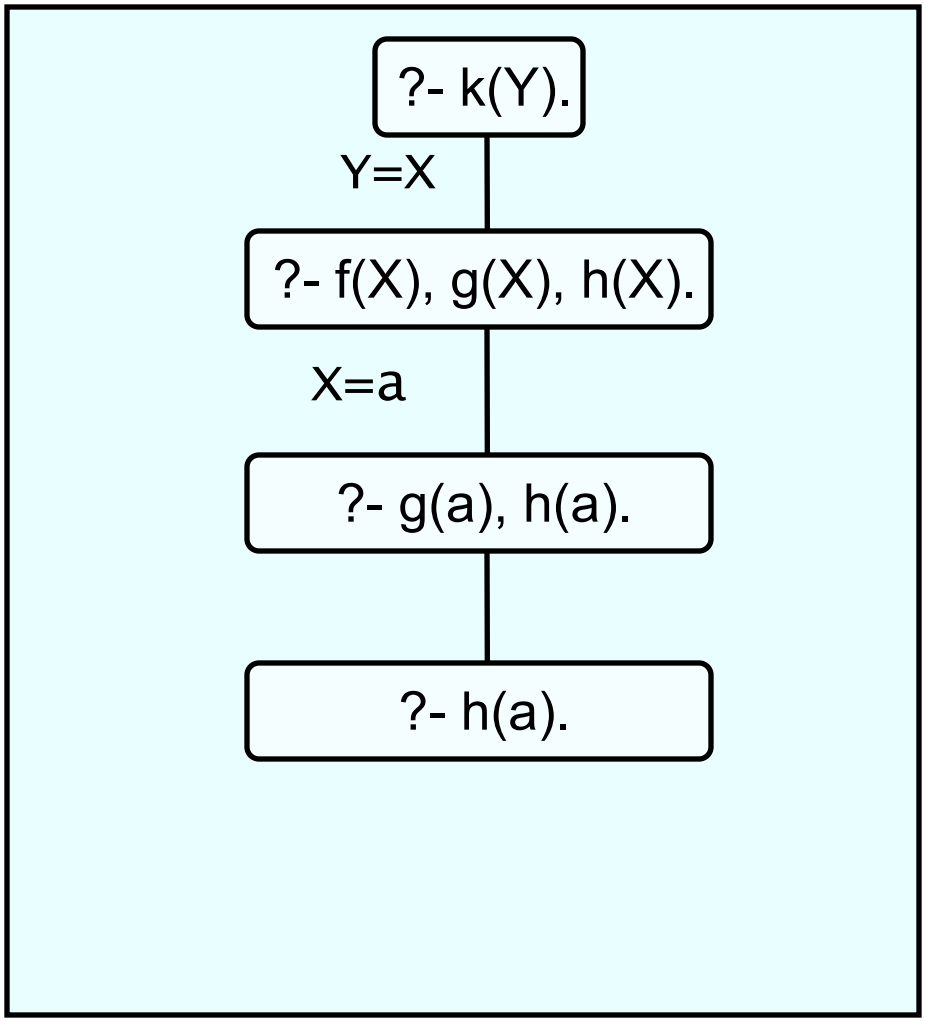


© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Exemple: arbre de recherche

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

```
?- k(Y).
```

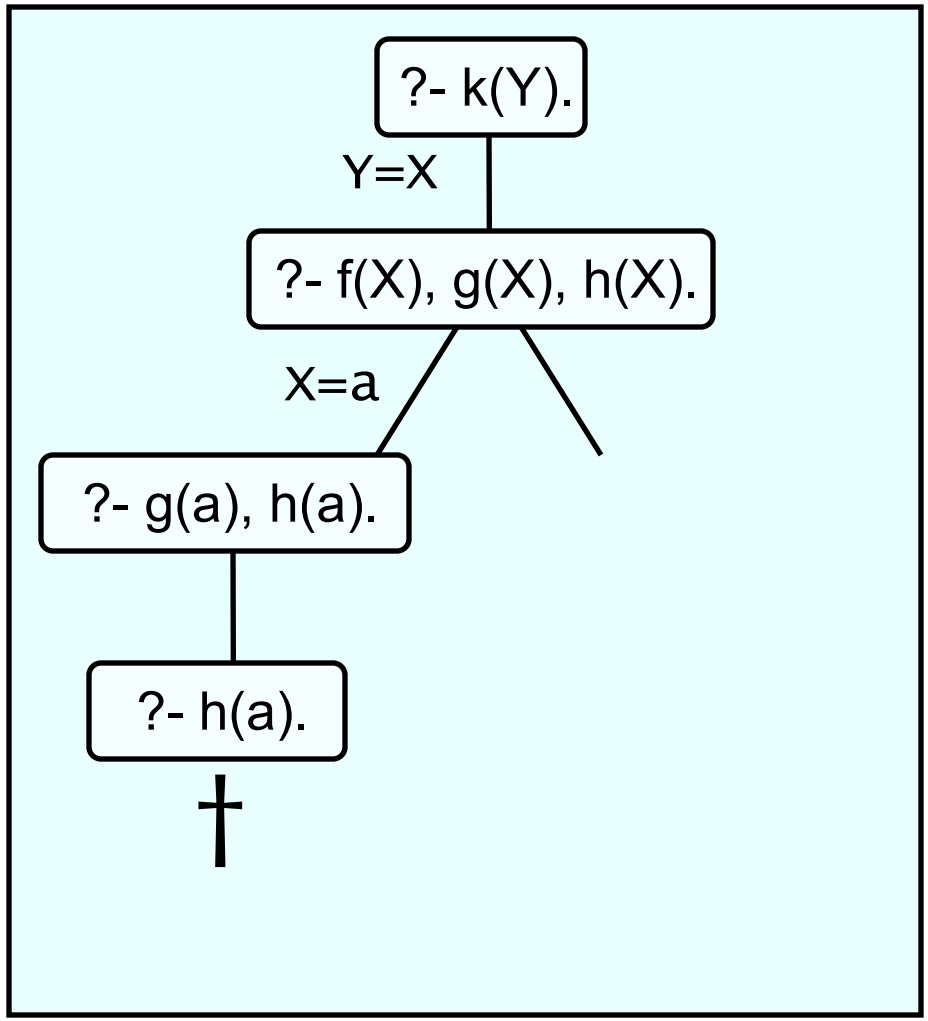


© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Exemple: arbre de recherche

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

```
?- k(Y).
```

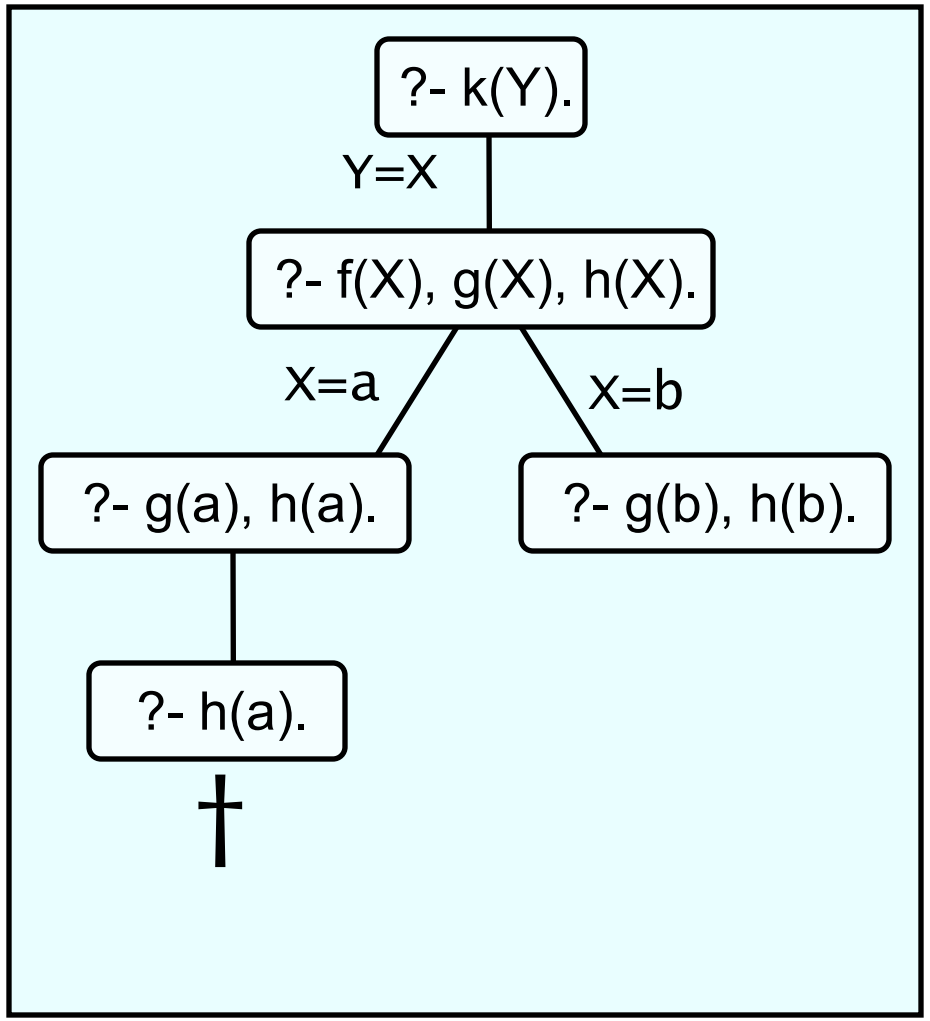


© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Exemple: arbre de recherche

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

```
?- k(Y).
```

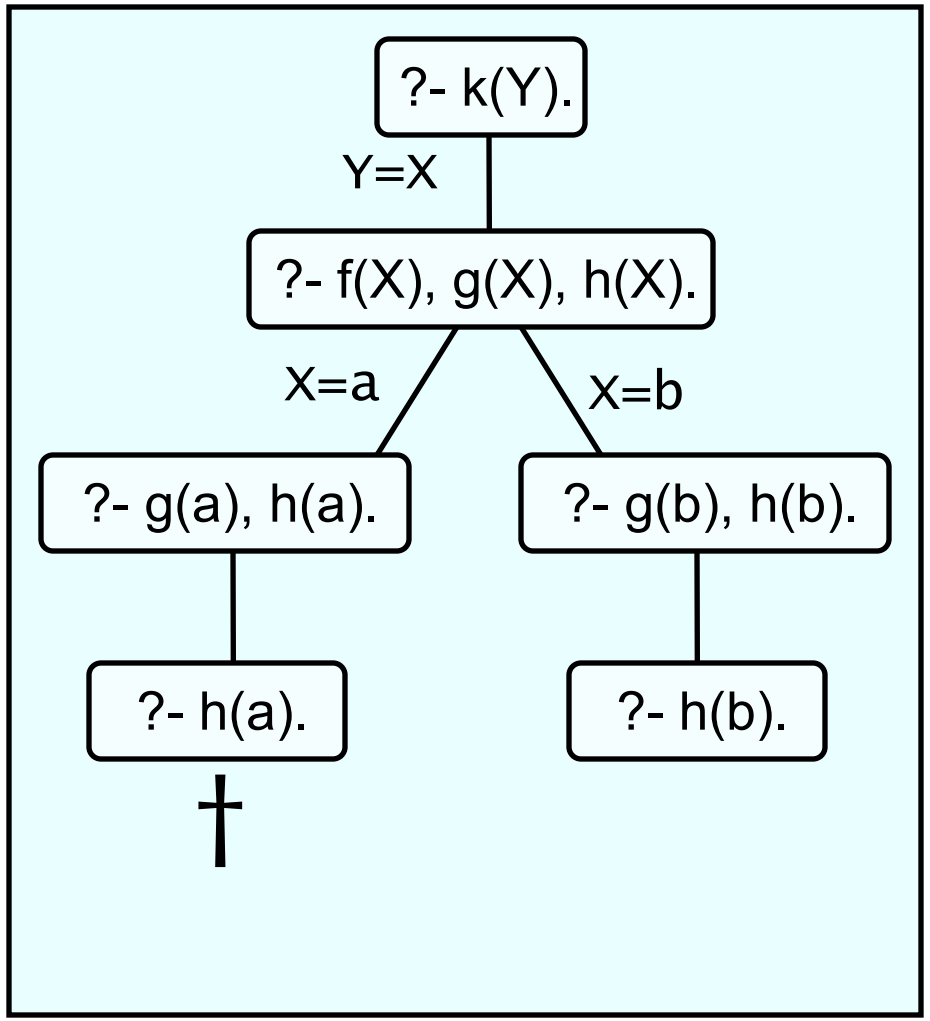


© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Exemple: arbre de recherche

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

```
?- k(Y).
```

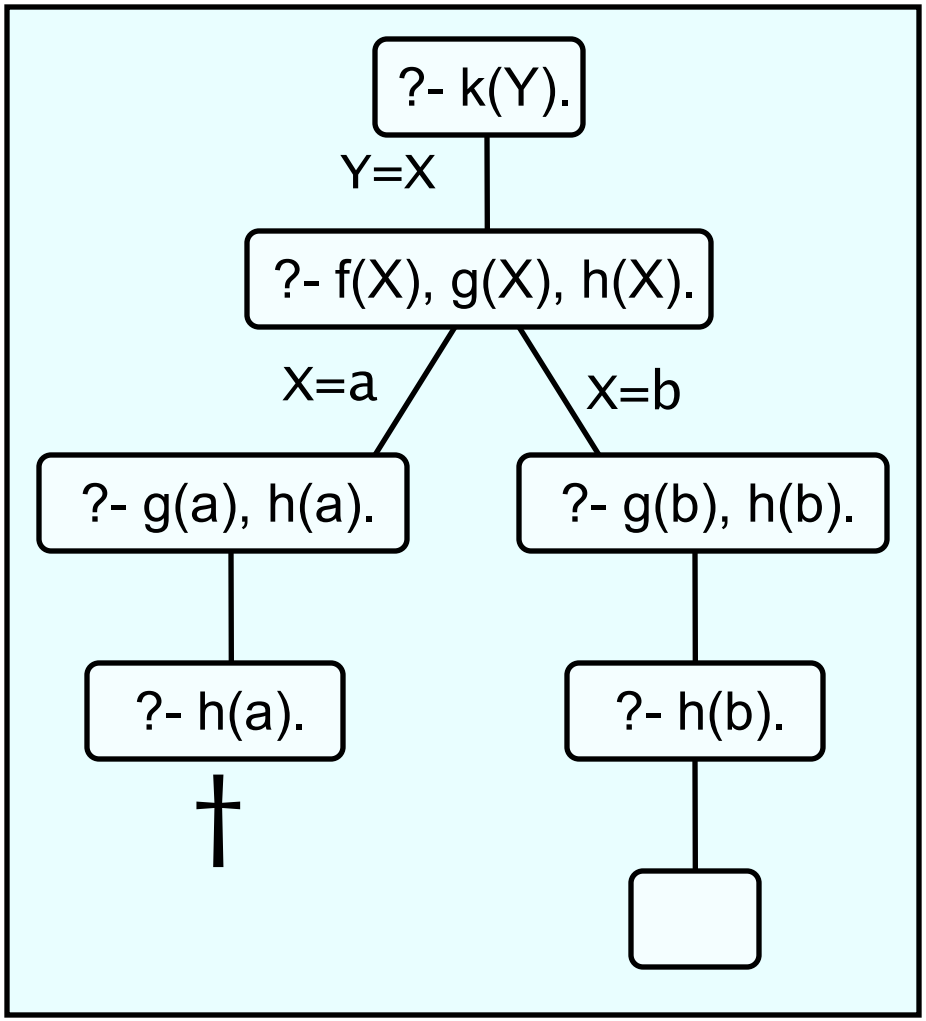


© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Exemple: arbre de recherche

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

```
?- k(Y).
Y=b
```

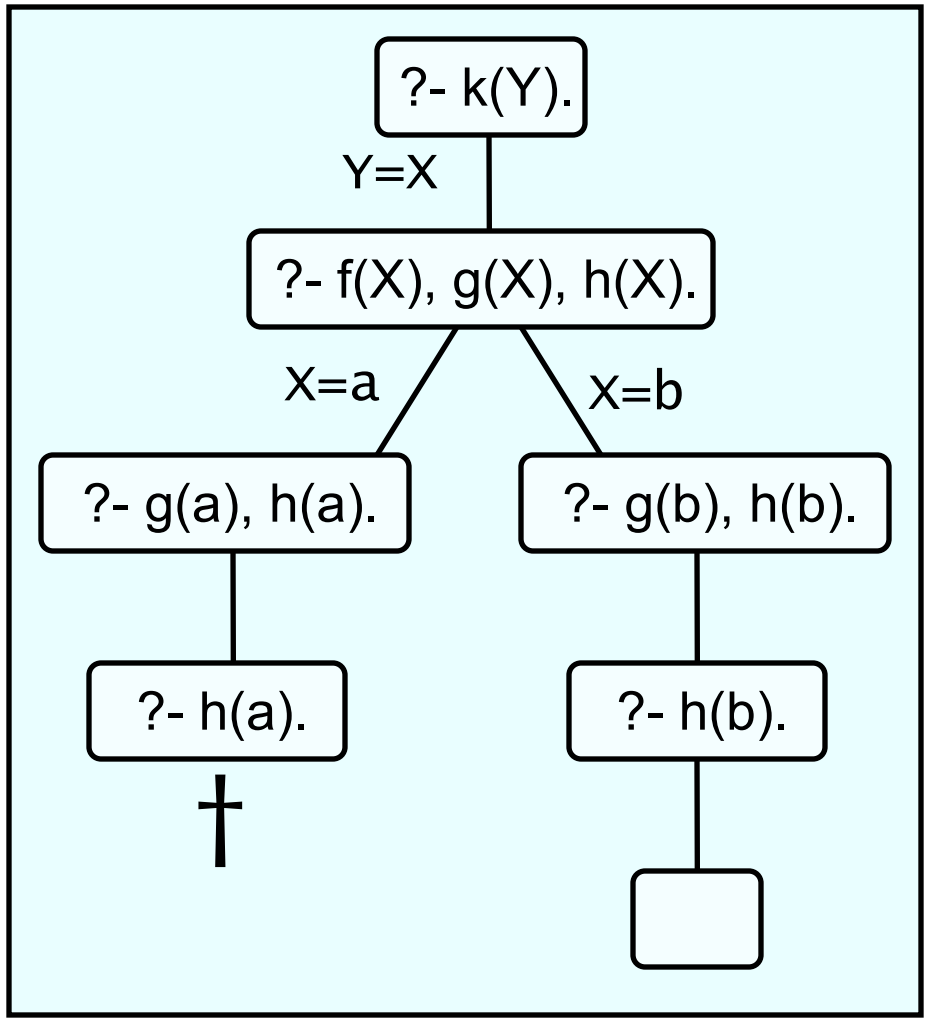


© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Exemple: arbre de recherche

```
f(a).
f(b).
g(a).
g(b).
h(b).
k(X):- f(X), g(X), h(X).
```

```
?- k(Y).
Y=b;
no
?-
```



© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Un autre exemple

```
aime(vincent,mia).
aime(marsellus,mia).

jaloux(A,B):-
    aime(A,C),
    aime(B,C).
```

```
?- jaloux(X,Y).
```

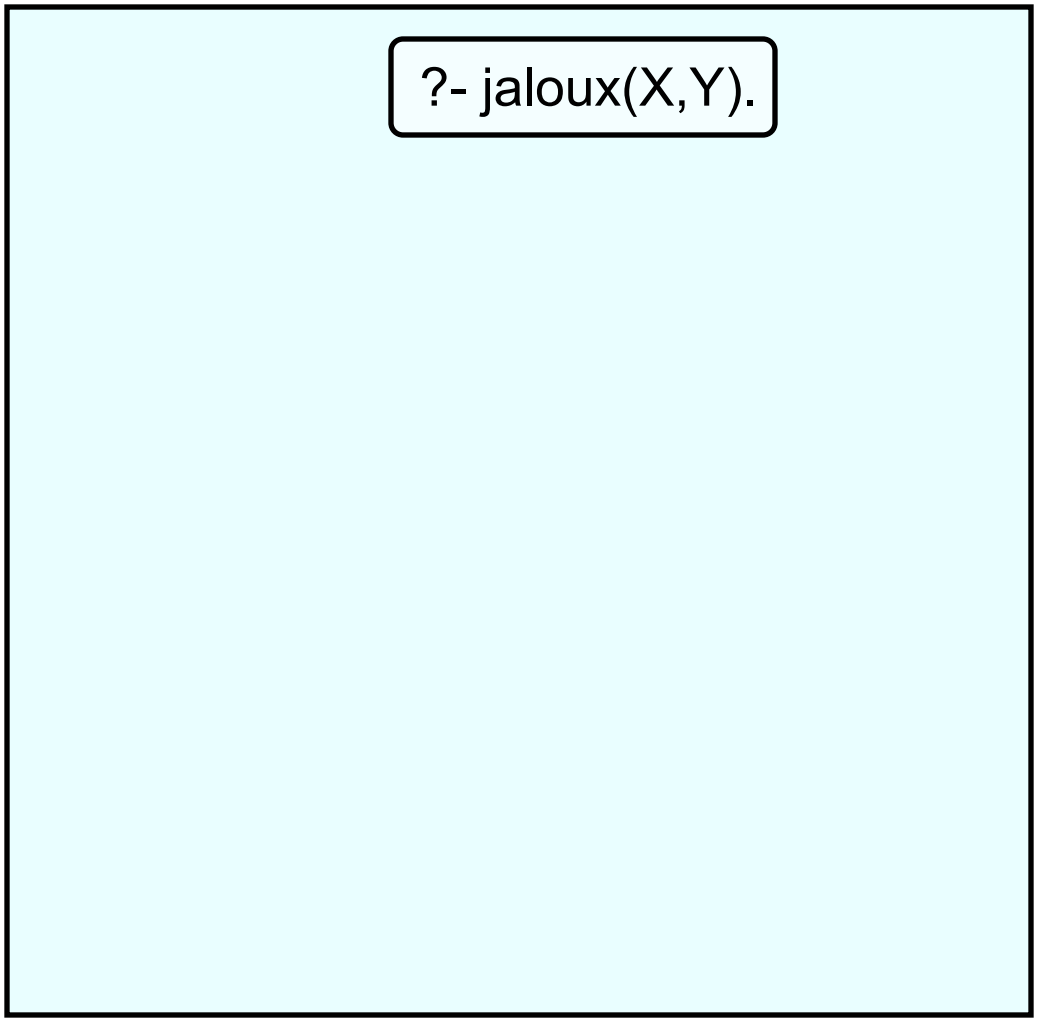
© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Un autre exemple

```
aime(vincent,mia).
aime(marsellus,mia).

jaloux(A,B):-
    aime(A,C),
    aime(B,C).
```

```
?- jaloux(X,Y).
```



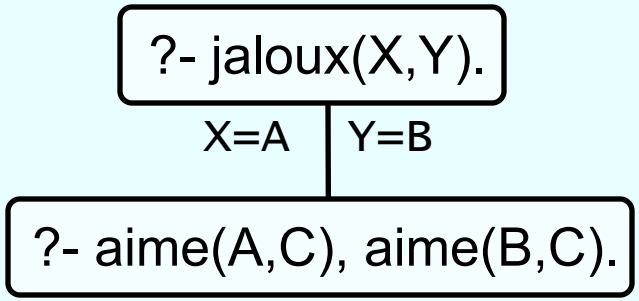
© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Un autre exemple

```
aime(vincent,mia).
aime(marsellus,mia).

jaloux(A,B):-
    aime(A,C),
    aime(B,C).
```

```
?- jaloux(X,Y).
```



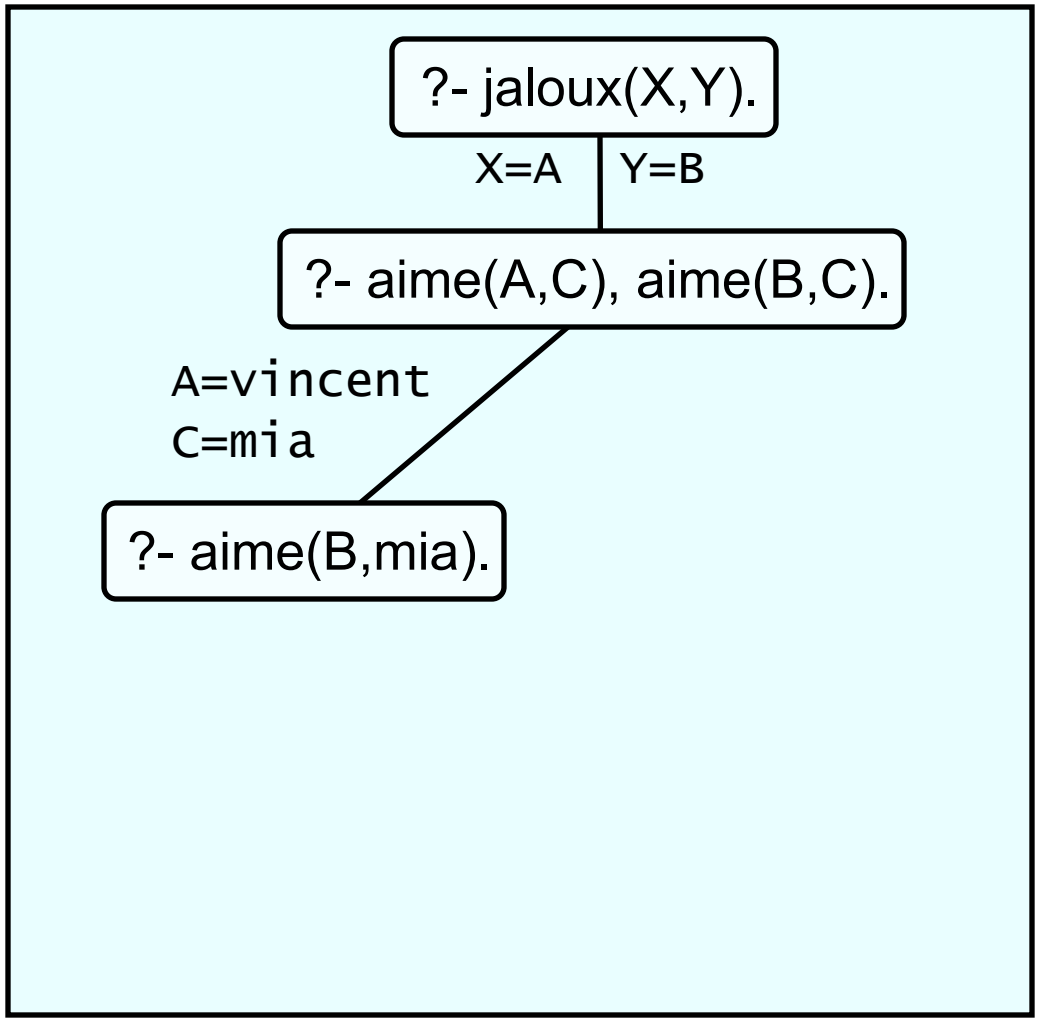
© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Un autre exemple

```
aime(vincent,mia).
aime(marsellus,mia).

jaloux(A,B):-
    aime(A,C),
    aime(B,C).
```

```
?- jaloux(X,Y).
```



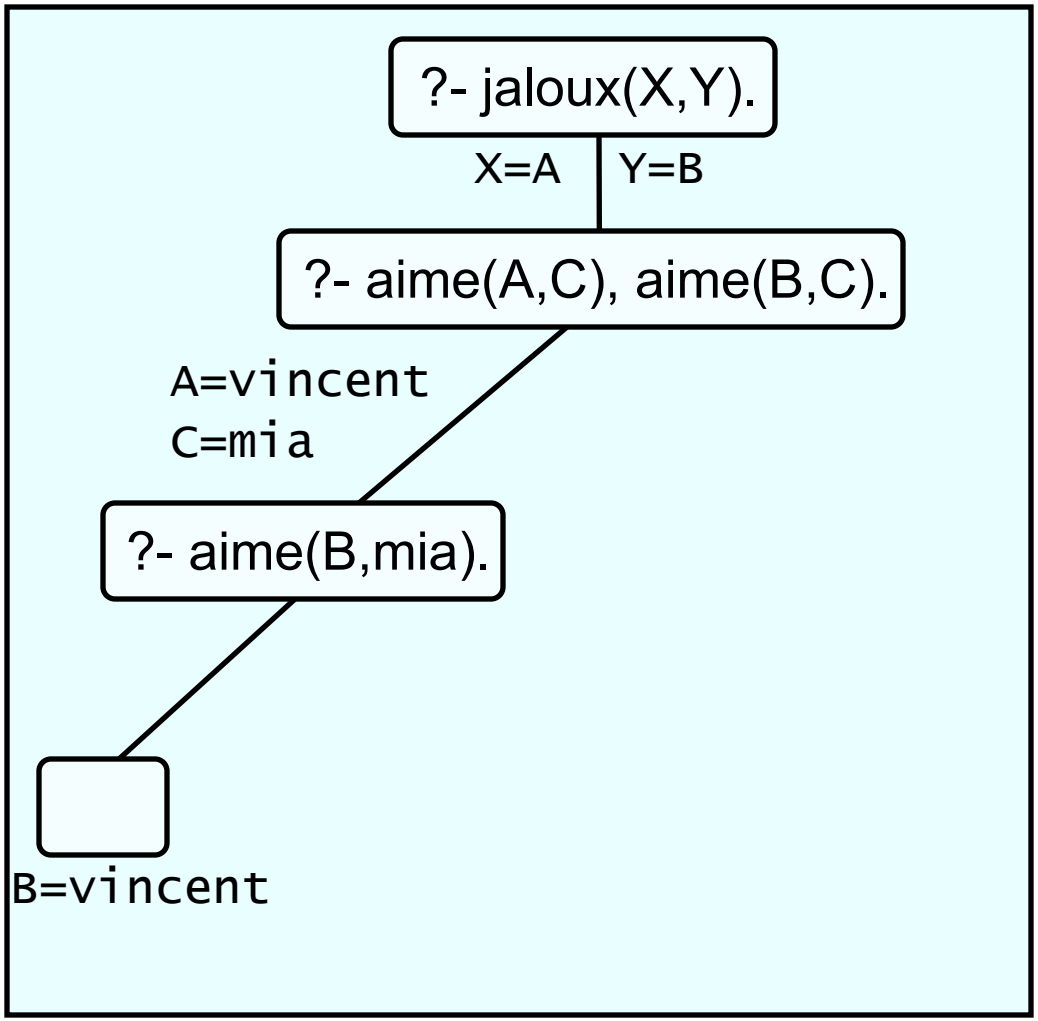
© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Un autre exemple

```
aime(vincent,mia).
aime(marsellus,mia).

jaloux(A,B):-
    aime(A,C),
    aime(B,C).
```

```
?- jaloux(X,Y).
X=vincent
Y=vincent
```



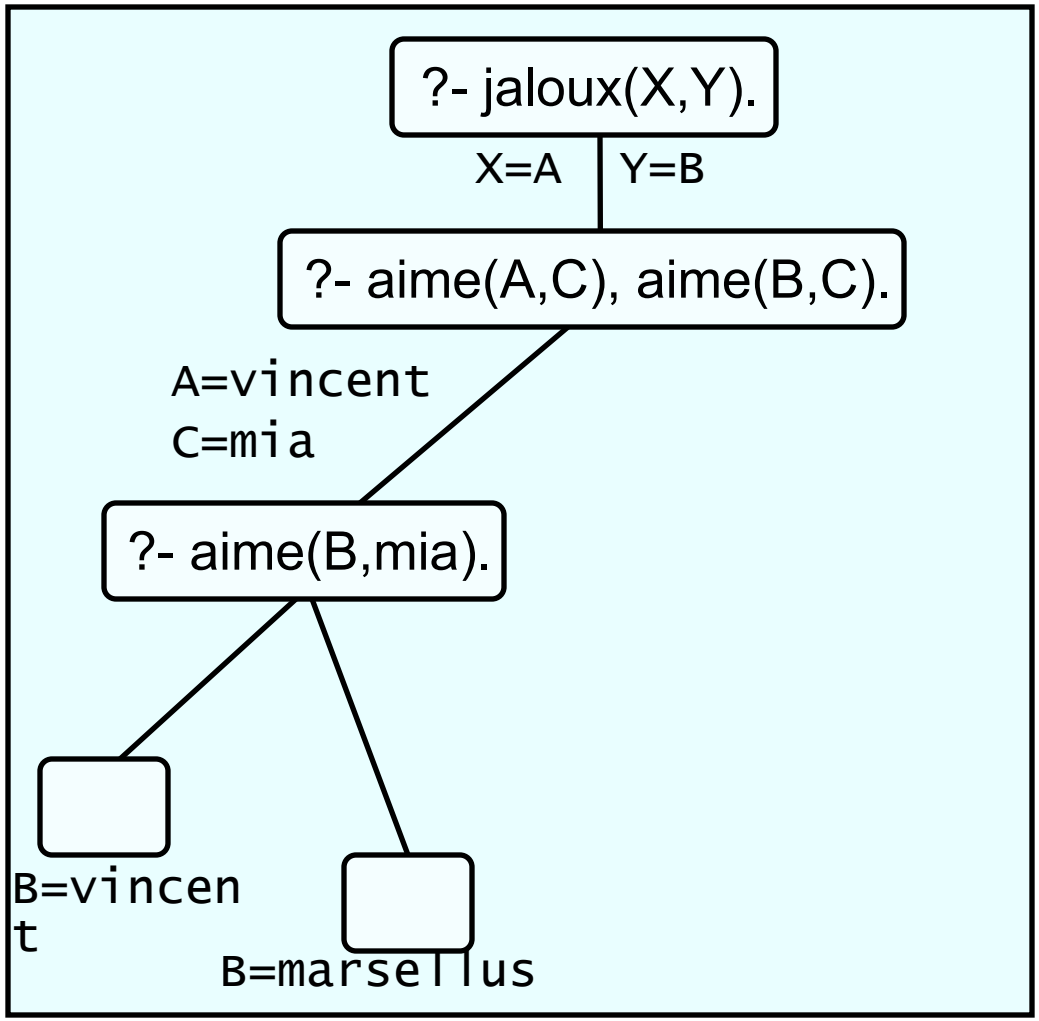
© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Un autre exemple

```
aime(vincent,mia).
aime(marsellus,mia).

jaloux(A,B):-
    aime(A,C),
    aime(B,C).
```

```
?- jaloux(X,Y).
X=vincent
Y=vincent;
X=vincent
Y=marsellus
```



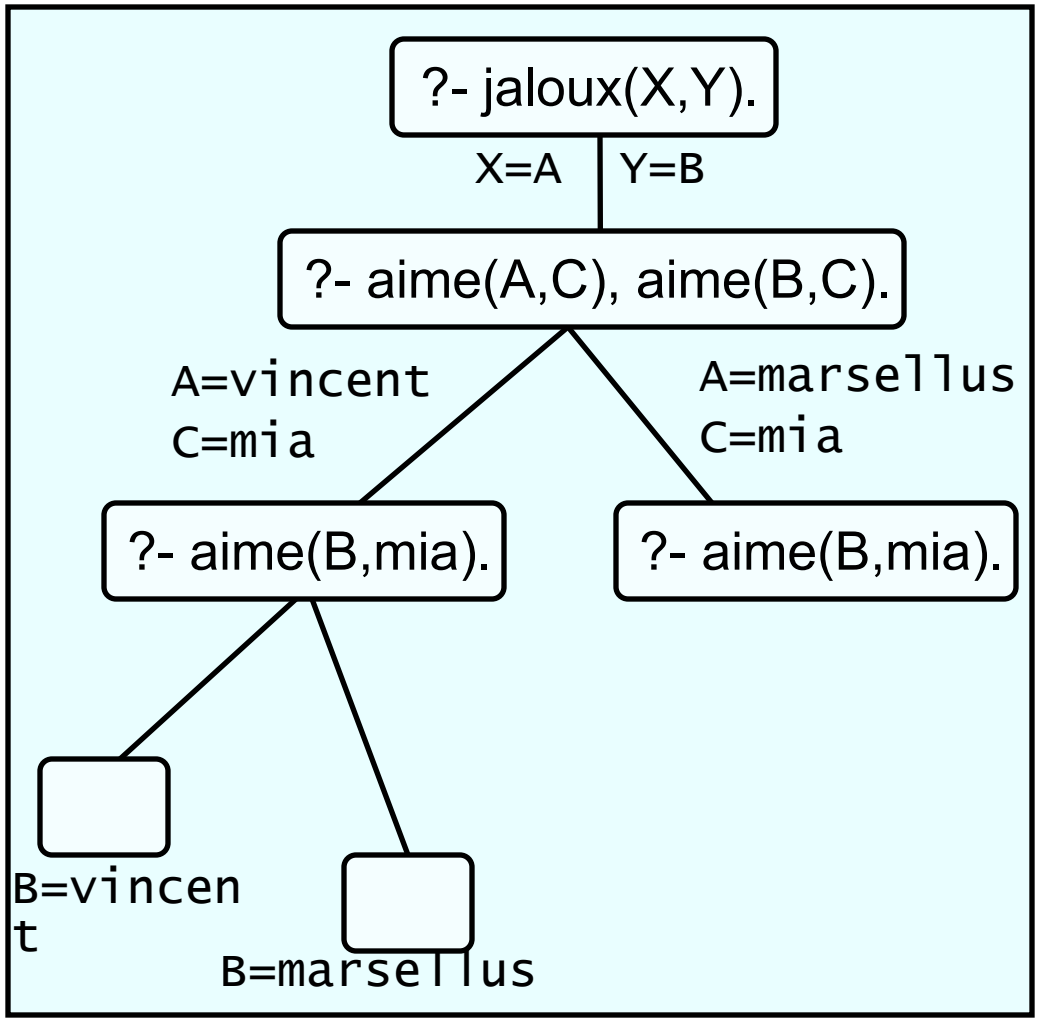
© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Un autre exemple

```
aime(vincent,mia).
aime(marsellus,mia).

jaloux(A,B):-
    aime(A,C),
    aime(B,C).
```

```
?- jaloux(X,Y).
X=vincent
Y=vincent;
X=vincent
Y=marsellus;
```



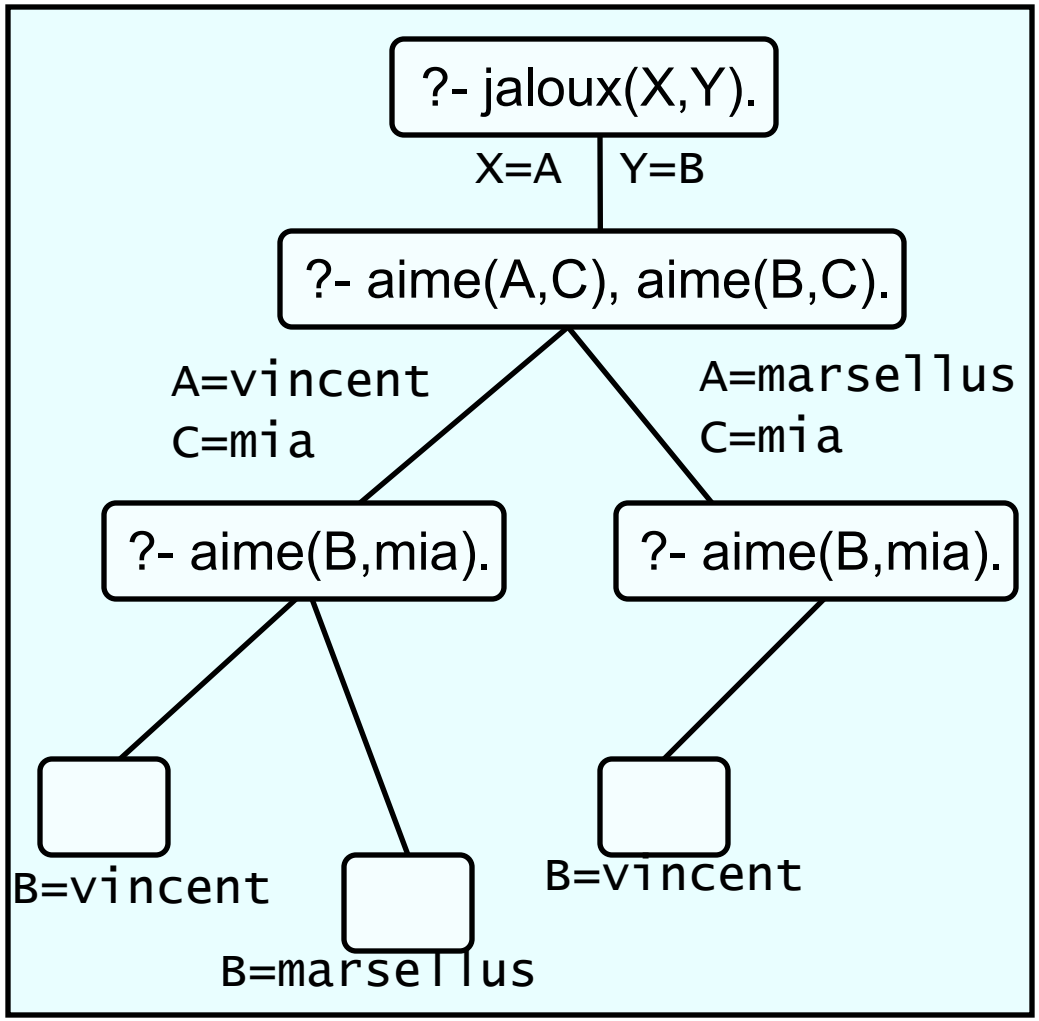
© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Un autre exemple

```
aime(vincent,mia).
aime(marsellus,mia).

jaloux(A,B):-
    aime(A,C),
    aime(B,C).
```

```
....
X=vincent
Y=marsellus;
X=marsellus
Y=vincent
```



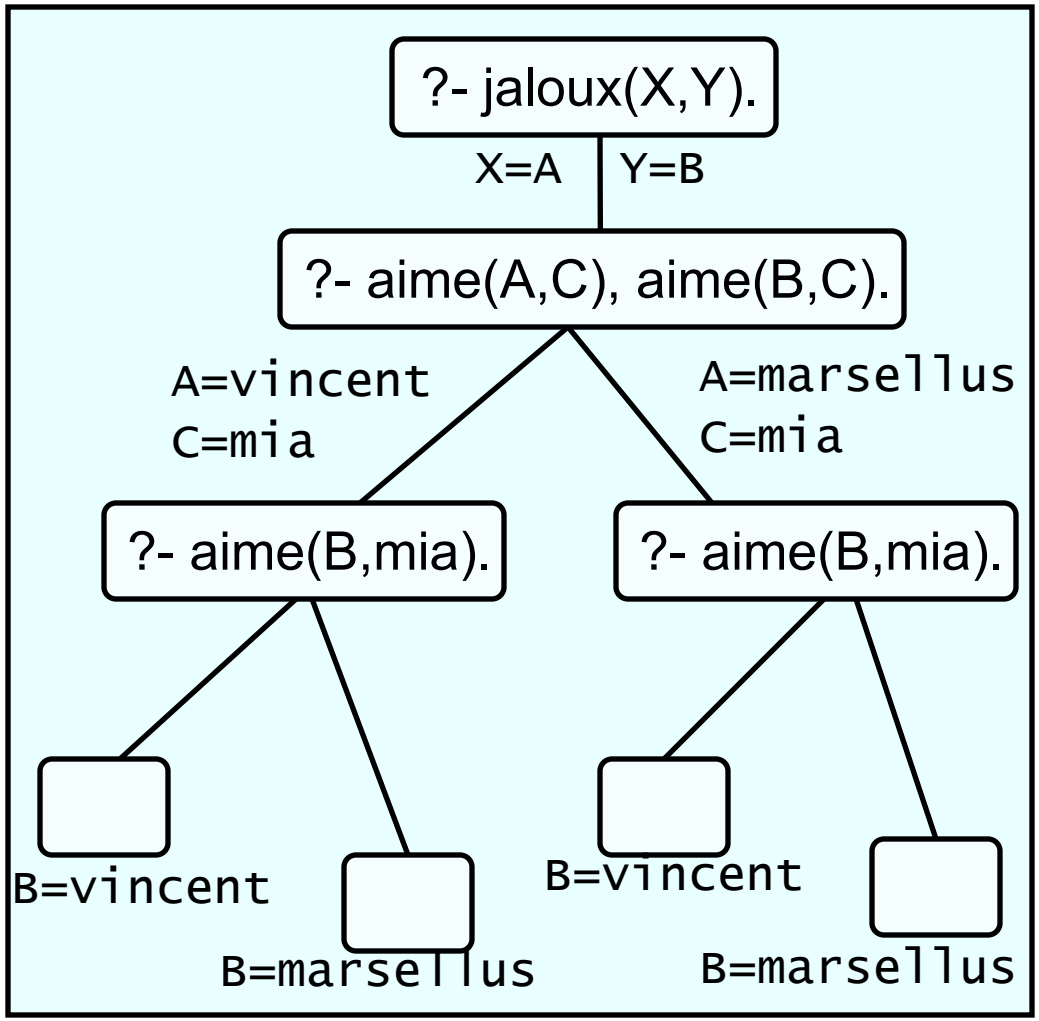
© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Un autre exemple

```
aime(vincent,mia).
aime(marsellus,mia).

jaloux(A,B):-
    aime(A,C),
    aime(B,C).
```

```
....
X=marsellus
Y=vincent;
X=marsellus
Y=marsellus
```



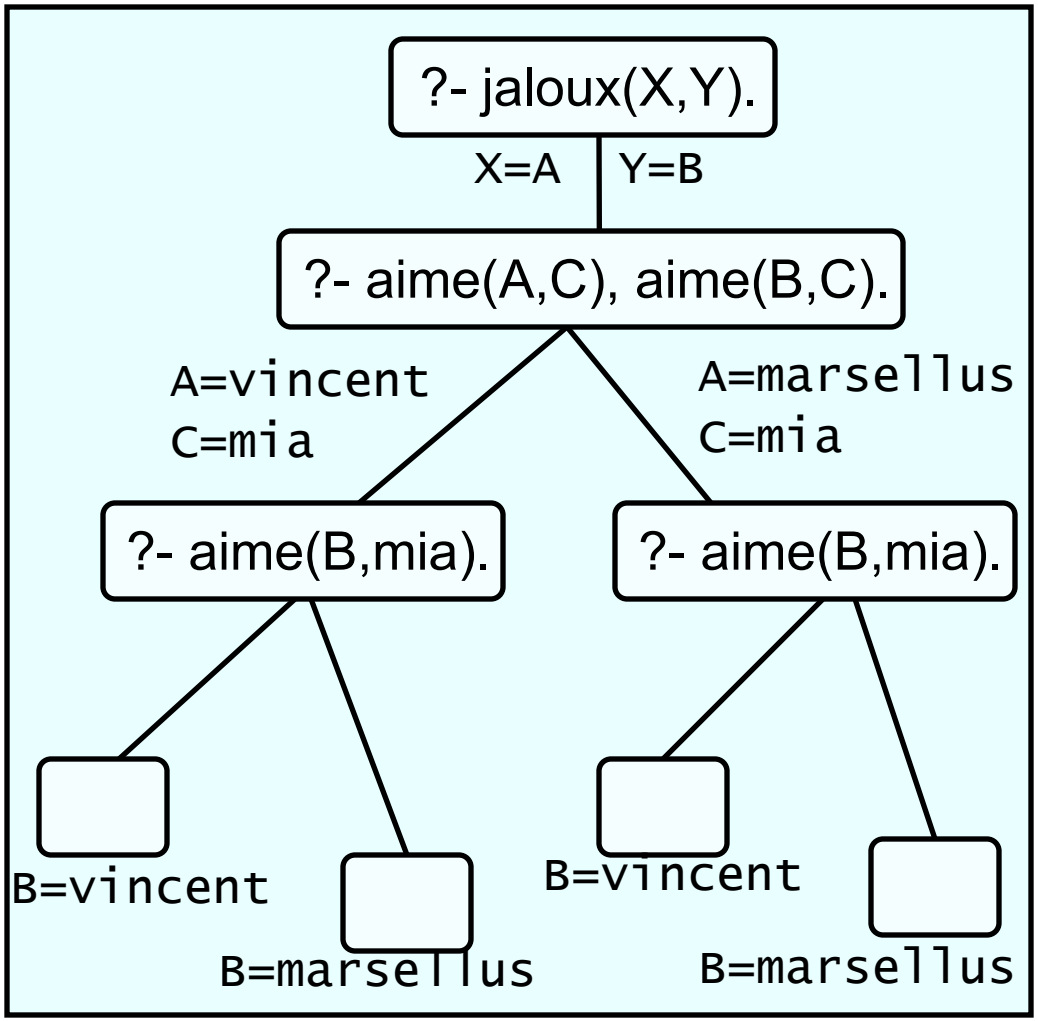
© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Un autre exemple

```
aime(vincent,mia).
aime(marsellus,mia).

jaloux(A,B):-
    aime(A,C),
    aime(B,C).
```

```
....
X=marsellus
Y=vincent;
X=marsellus
Y=marsellus;
no
```



© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Exercices

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Résumé de la séance

- Nous avons
  - Défini ce qu'est l'unification
  - Vu la différence entre l'unification standard et celle de Prolog
  - Présenter les arbres de recherche

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève

Prochaine séance

- La récursivité en Prolog
  - Les définitions récursives en Prolog
  - Les différences entre l'approche déclarative d'un programme en Prolog et une approche procédurale

© Patrick Blackburn, Johan Bos & Kristina Striegnitz. Trad: Yannick Estève