

Programmation par Contraintes

Module du Master "Systèmes Informatiques Intelligents" 2ème année

Chapitre V

Programmation logique

Mr ISLI

Département d'Informatique

Faculté d'Electronique et d'Informatique

Université des Sciences et de la Technologie Houari Boumediène

BP 32, El-Alia, Bab Ezzouar

DZ-16111 ALGER

http://perso.usthb.dz/~aisli/TA_PpC.htm

aisli@usthb.dz



CHAPITRE V

Programmation logique

Une première vue globale :

- Programme Prolog = ensemble de clauses de Horn
- Clause de Horn = affirmation sur des atomes logiques
- Atome logique = relation sur des termes
- Terme = objet de l'univers

Terme – atome logique – clause de Horn – programme Prolog



CHAPITRE V

Programmation logique

Les termes PROLOG

- Les variables
- Les termes élémentaires
- Les termes composés



CHAPITRE V

Programmation logique

Les variables PROLOG

- Chaîne sur l'alphabet alphanumérique augmenté du symbole de soulignement, commençant par une lettre majuscule ou par le caractère de soulignement (`_objet`, `_100`)
 - `Var`, `X`, `Var_longue_2`
 - `_objet`, `_100`
- Les variables commençant par "`_`" sont dites variables anonymes



CHAPITRE V

Programmation logique

Les termes élémentaires PROLOG

- Les nombres
 - entiers ou flottants
- Les identificateurs (aussi appelés atomes)
 - chaîne alphanumérique commençant par une minuscule
 - **Exemples** : alger bureau226 babezzouar
- Les chaînes de caractères entre guillemets
 - **Exemples** : "#@" "alger" "2012"



CHAPITRE V

Programmation logique

Les termes composés PROLOG

- $\text{foncteur}(t_1, \dots, t_n)$
- **foncteur** : chaîne alphanumérique commençant par une minuscule (identificateur)
- t_1, \dots, t_n : termes
- n : arité du foncteur
- **Exemples**
 - $\text{adresse}(18, \text{"rue Abane Ramdane"}, \text{"Alger"})$
 - $\text{cons}(a, \text{cons}(X, \text{nil}))$



CHAPITRE V

Programmation logique

Les listes : définition

La liste est un terme composé particulier, de symbole de fonction ``.`` et d'arité 2:

- le premier argument est l'élément de tête de la liste,
- le deuxième argument est la queue de la liste.
- la liste vide est notée `[]`.



CHAPITRE V

Programmation logique

Listes : notations

- la liste $.(X,L)$ est également notée $[X|L]$,
- la liste $.(X1, .(X2, L))$ est également notée $[X1, X2|L]$,
- la liste $.(X1, .(X2, \dots, .(Xn, L) \dots))$ est également notée $[X1, X2, \dots, Xn|L]$,
- la liste $[X1, X2, X3, \dots, Xn|[]]$ est également notée $[X1, X2, X3, \dots, Xn]$.
- Par exemple, la liste $[a,b,c]$ correspond à la liste $.(a,.(b,.(c,[])))$ et contient les 3 éléments a , b et c .
- La liste $[a,b|L]$ correspond à la liste $.(a,.(b,L))$ et contient les 2 éléments a et b , suivis de la liste (inconnue) L .



CHAPITRE V

Programmation logique

Listes : leur importance en Prolog

Une liste est une structure récursive: la liste $Liste = [X|L]$ est composée d'un élément de tête X et d'une queue de liste L qui est elle-même une liste. Par conséquent, les relations Prolog qui ``manipulent'' les listes seront généralement définies par

- une ou plusieurs clauses récursives, définissant la relation sur la liste $[X|L]$ en fonction de la relation sur la queue de liste L ,
- une ou plusieurs clauses non récursives assurant la terminaison de la manipulation, et définissant la relation pour une liste particulière (par exemple, la liste vide, ou la liste dont l'élément de tête vérifie une certaine condition...).

CHAPITRE V

Programmation logique

Les atomes logiques :

ou propositions atomiques

- `symbole_de_prédicat(t1, ..., tn)`
- `symbole_de_prédicat` : chaîne alphanumérique commençant par une minuscule (identificateur)
- `t1, ..., tn` : termes
- Exemples
 - `pere(rachid,ines)`
 - `habite(X,adresse(18,"rue Abane Ramdane","Alger"))`



CHAPITRE V

Programmation logique

Les clauses

- Une clause
 - affirmation inconditionnelle (fait)
 - affirmation conditionnelle (règle)



CHAPITRE V

Programmation logique

Les clauses

- fait : atome logique A
 - la relation définie par A est vraie (sans condition)
 - Exemple
 - pere(rachid,ines)
 - la relation "rachid est le père de ines" est vraie (sans condition)



CHAPITRE V

Programmation logique

Les clauses

- règle : $A_0 :- A_1, \dots, A_n$.
 - A_0, A_1, \dots, A_n : atomes logiques
 - La relation A_0 est vraie si les relations A_1, \dots, A_n sont vraies
 - A_0 : tête de la clause
 - A_1, \dots, A_n : corps de la clause



CHAPITRE V

Programmation logique

Les clauses

- Une variable apparaissant dans la tête d'une règle (et éventuellement dans son corps) est quantifiée universellement
- Une variable apparaissant dans le corps d'une clause mais pas dans sa tête est quantifiée existentiellement



CHAPITRE V

Programmation logique

Les clauses : exemple

- `meme_pere(X,Y) :- pere(P,X), pere(P,Y).`
- pour tout X et pour tout Y
 - `meme_pere(X,Y)` est vrai s'il existe un P tel que `pere(P,X)` et `pere(P,Y)` soient vrais



CHAPITRE V

Programmation logique

Un programme prolog

- Suite de clauses regroupées en paquets
- L'ordre dans lequel sont définis les paquets n'est pas significatif
- Paquet :
 - Suite de clauses ayant le même atome de tête
 - L'arité de l'atome de tête (d'un même paquet) est la même au niveau de toutes les clauses du paquet
 - Un paquet définit un prédicat (dont le nom est l'atome de tête du paquet)
 - Un paquet définit une disjonction de clauses
 - L'ordre des clauses dans un paquet est donc important, voire crucial
 - L'efficacité de la résolution en dépend
- Un programme prolog définit une conjonction de paquets



CHAPITRE V

Programmation logique

Un programme prolog : exemple

personne(X) :- homme(X).

personne(X) :- femme(X).

- Un paquet à deux clauses
- pour tout X
 - personne(X) est vrai si homme(X) est vrai ou femme(X) est vrai



CHAPITRE V

Programmation logique

Exécution d'un programme prolog

- Exécuter un programme prolog revient à poser une question à l'interprète PROLOG
- **Question** (également appelée **but** ou **activant**)
 - Suite d'atomes logiques séparés par des virgules : A_1, \dots, A_n
- Réponse de l'interprète PROLOG à une question
 - **yes** si la question est une conséquence logique du programme
 - **no** sinon (si la question n'est pas une conséquence logique du programme)



CHAPITRE V

Programmation logique

Exécution d'un programme prolog

- une question peut inclure des variables, qui sont alors quantifiées existentiellement
 - La réponse de Prolog à une telle question est l'ensemble des valeurs des variables pour lesquelles la question est une conséquence logique du programme
 - Toutes les instanciations satisfaisant la formule

programme \Rightarrow question

(\Rightarrow est l'implication logique)



CHAPITRE V

Programmation logique

Exécution d'un programme prolog

?- pere(rachid,X), pere(X,Y).

- Existe-t-il un X et un Y tels que pere(rachid,X) et pere(X,Y) soient vrais
- Réponse de prolog :
 - enfants et petits-enfants de rachid si rachid est grand-père



CHAPITRE V

Programmation logique

Dénotation d'un programme Prolog : sémantique logique

- La dénotation d'un programme Prolog P est l'ensemble de tous les atomes logiques A qui sont des conséquences logiques de P
- Lorsqu'on exécute un programme Prolog P en posant une question (ou but) Q :
 - La réponse de Prolog est l'ensemble des instances de la question Q qui font partie de la dénotation de P



CHAPITRE V

Programmation logique

Dénotation d'un programme Prolog : sémantique logique

La dénotation d'un programme Prolog P peut être calculée par une approche ascendante (en chaînage avant)

- Partir des faits
- Appliquer les règles pour déduire de nouveaux faits, jusqu'à fermeture



CHAPITRE V

Programmation logique

Dénotation d'un programme Prolog : sémantique logique

parent(rachid,ines).

parent(ines,mohamed).

parent(ali,younes).

parent(younes,said).

homme(rachid).

homme(ali).

pere(X,Y):-parent(X,Y),homme(X).

gdpere(X,Y):-pere(X,Z),parent(Z,Y).



CHAPITRE V

Programmation logique

Dénotation d'un programme Prolog : sémantique logique

- L'ensemble des faits de P (relations vraies sans condition) :
 - $E_0 = \{\text{parent}(\text{rachid}, \text{ines}),$
 $\text{parent}(\text{ines}, \text{mohamed}),$
 $\text{parent}(\text{ali}, \text{younes}),$
 $\text{parent}(\text{younes}, \text{said}),$
 $\text{homme}(\text{rachid}),$
 $\text{homme}(\text{ali})\}$
- On applique les règles de P à E_0 pour obtenir :
 - $E_1 = \{\text{pere}(\text{rachid}, \text{ines}),$
 $\text{pere}(\text{ali}, \text{younes})\}$



CHAPITRE V

Programmation logique

Dénotation d'un programme Prolog : sémantique logique

- On applique les règles de **P** à $E_0 \cup E_1$ pour obtenir :
 - $E_2 = \{\text{gdpere}(\text{rachid}, \text{mohamed}),$
 $\text{gdpere}(\text{ali}, \text{said})\}$
 - L'application des règles de **P** à $E_0 \cup E_1 \cup E_2$ ne permet pas de déduire de nouveaux faits
 - La dénotation de **P** est donc
$$E_0 \cup E_1 \cup E_2 = \{\text{parent}(\text{rachid}, \text{ines}), \text{parent}(\text{ines}, \text{mohamed}),$$
$$\text{parent}(\text{ali}, \text{younes}), \text{parent}(\text{younes}, \text{said}),$$
$$\text{homme}(\text{rachid}), \text{homme}(\text{ali}),$$
$$\text{pere}(\text{rachid}, \text{ines}), \text{pere}(\text{ali}, \text{younes}),$$
$$\text{gdpere}(\text{rachid}, \text{mohamed}),$$
$$\text{gdpere}(\text{ali}, \text{said})\}$$



CHAPITRE V

Programmation logique

Dénotation d'un programme Prolog : sémantique logique

La dénotation d'un programme peut être infinie :

plus(0,X,X).

plus(succ(X),Y,succ(Z)):-plus(X,Y,Z).

- **E={plus(0,X,X),
plus(succ(0),X,succ(X)),
plus(succ(succ(0)),X,succ(succ(X))),
plus(succ(succ(succ(0))),X,succ(succ(succ(X)))), ...}**



CHAPITRE V

Programmation logique

Sémantique opérationnelle :

Substitution

- Fonction s de l'ensemble des variables dans l'ensemble des termes
- $s = \{X \leftarrow Y, Z \leftarrow f(a, Y)\}$ est la substitution remplaçant X par Y et Z par $f(a, Y)$, et laisse inchangées les autres variables
- Une substitution peut être appliquée à un atome :
 - $s(p(X, f(Y, Z))) = p(s(X), f(s(Y), s(Z))) = P(Y, f(Y, f(a, Y)))$



CHAPITRE V

Programmation logique

Sémantique opérationnelle :

Instance

- Une instance d'un atome logique A est l'atome $s(A)$ obtenu par application d'une substitution s à A
- L'atome $\text{pere}(\text{rachid}, \text{ines})$ est une instance de l'atome $\text{pere}(\text{rachid}, X)$:
 - Appliquer, par exemple, la substitution $s = \{X \leftarrow \text{ines}\}$



CHAPITRE V

Programmation logique

Sémantique opérationnelle :

Unificateur

- Un unificateur de deux atomes logiques $A1$ et $A2$ est une substitution s telle que $s(A1)=s(A2)$
- $s=\{X\leftarrow a, Z\leftarrow f(a, Y)\}$ est un unificateur des deux atomes $A1=p(X, f(X, Y))$ et $A2=p(a, Z)$
- $s(A1)=s(A2)=p(a, f(a, Y))$
- Unifier deux atomes $A1$ et $A2$ revient à résoudre l'équation $A1=A2$



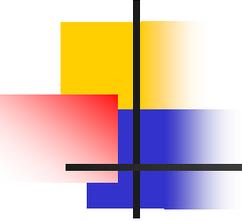
CHAPITRE V

Programmation logique

Sémantique opérationnelle :

Unificateur le plus général

- Un unificateur s de deux atomes logiques $A1$ et $A2$ est dit unificateur le plus général (**upg**) si pour tout autre unificateur s' , il existe une substitution s'' telle que $s' = s''(s)$
- $s = \{X \leftarrow Y\}$ est un upg des deux atomes $p(X, b)$ et $p(Y, b)$
- $s' = \{X \leftarrow a, Y \leftarrow a\}$ est un unificateur de $p(X, b)$ et $p(Y, b)$ **mais pas un upg**



CHAPITRE V

Programmation logique

Sémantique opérationnelle :

Algorithme d'unification de Robinson

- Entrée : deux atomes logiques A1 et A2
- Sortie :
 - un upg de A1 et A2 si A1 et A2 sont unifiables
 - Échec sinon



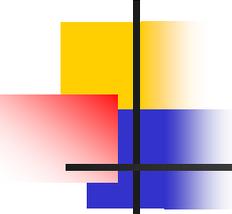
CHAPITRE V

Programmation logique

Sémantique opérationnelle :

Algorithme d'unification de Robinson : unification de deux termes

- Si T1 et T2 sont des termes élémentaires (nombres, identificateurs ou chaînes de caractères), alors T1 et T2 sont unifiables ssi ils consistent en le même terme élémentaire
- Si T1 est une variable et T2 est un terme quelconque, alors T1 et T2 sont unifiables, et T1 est instancié à T2
- Si T2 est une variable et T1 est un terme quelconque, alors T1 et T2 sont unifiables, et T2 est instancié à T1
- Si T1 et T2 sont des termes complexes, alors ils sont unifiables ssi :
 - Ils ont le même foncteur et la même arité,
 - Le $i^{\text{ème}}$ argument de T1 et le $i^{\text{ème}}$ argument de T2 sont unifiables, et
 - Les instanciations des variables sont compatibles (par exemple, il n'est pas possible d'instancier une variable X à **meriem** quand on unifie une paire d'arguments, et d'instancier X à **ania** quand on unifie une autre paire d'arguments)
- Deux termes sont unifiables ssi les quatre points ci-dessus permettent de montrer qu'ils le sont



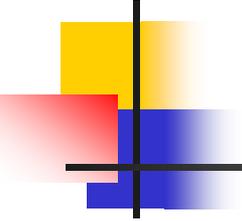
CHAPITRE V

Programmation logique

Sémantique opérationnelle :

Algorithme d'unification de Robinson : unification de deux atomes logiques

- Si A1 et A2 sont deux atomes logiques (propositions atomiques), alors ils sont unifiables ssi :
 - Ils ont le même symbole de prédicat et la même arité,
 - Le $i^{\text{ème}}$ argument de A1 et le $i^{\text{ème}}$ argument de A2 sont unifiables, et
 - Les instanciations des variables sont compatibles (par exemple, il n'est pas possible d'instancier une variable X à **meriem** quand on unifie une paire d'arguments, et d'instancier X à **ania** quand on unifie une autre paire d'arguments)



CHAPITRE V

Programmation logique

Sémantique opérationnelle :

Algorithme d'unification de Robinson : exemples

?- ines=ines.

true

?- ines=younes.

false

?- 54=68.

false

?- `ines`=ines.

true

?- `54`=54.

false

?- ines=X.

X=ines

?- X=Y.

X=Y

?- X=ines,Y=younes.

X=ines

Y=younes

?- f(g(h),Y)=f(X,i(f)).

X=g(h)

Y=i(f)

?- f(g(h),i(f))=f(X,i(Y)).

X=g(h)

Y=f

?- plus(X,X)=plus(2,9).

false



CHAPITRE V

Programmation logique

Sémantique opérationnelle :

Algorithme d'unification de Robinson :

Problème avec l'algorithme d'unification utilisé par Prolog :

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

```
X=pere(X)
```

```
?-
```

- L'algorithme d'unification de Prolog n'utilise pas la vérification d'occurrence (the occurs check) des algorithmes d'unification standard pour des raisons d'efficacité
- **Unification standard** : Si l'unification standard est confrontée à l'unification d'une variable avec un terme composé, elle commence par vérifier si la variable occure dans le terme (occurs check) : si tel est le cas, l'unification échoue.
- Prolog dispose néanmoins d'un prédicat prédéfini pour l'unification standard de deux termes (unification avec vérification d'occurrence) :

```
?- unify_with_occurs_check(pere(X),X).  
false  
?-
```



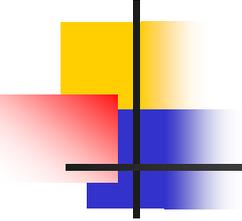
CHAPITRE V

Programmation logique

Sémantique opérationnelle :

Algorithme d'unification de Robinson : son utilisation par l'algorithme de résolution

- L'algorithme de résolution de Prolog teste si un but $B=A_1,\dots,A_n$ est une conséquence logique d'un programme P
- Un but est une suite d'atomes logiques séparés par des virgules, interprétés conjonctivement
- Atome logique=proposition atomique=symbole_de_predicat(terme1,...,termeN)
- But=Question=Activant
- Une action clé de l'algorithme de résolution :
 - sélectionner une clause de P dont l'atome de tête s'unifie avec le premier atome logique A_1 :
 $C_0:-C_1,\dots,C_m$
- Pour ne pas être confronté au problème de vérification d'occurrence, l'unification utilisée par l'algorithme de résolution a recours au renommage éventuel des variables de telle sorte qu'il n'y ait pas de variable commune au premier atome logique du but et à l'atome de tête de la clause sélectionnée



CHAPITRE V

Programmation logique

Sémantique opérationnelle

- L'ensemble de toutes les conséquences logiques d'un programme (dénotation) ne peut donc pas toujours être calculé
- MAIS on peut montrer si un but (question) est une conséquence logique d'un programme
 - But : suite d'atomes logiques (conjonction d'atomes logiques)

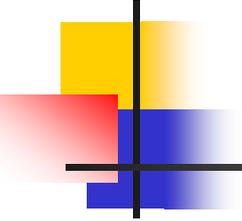


CHAPITRE V

Programmation logique

Sémantique opérationnelle

- Pour prouver si un but $B=A_1, \dots, A_n$ est une conséquence logique d'un programme P :
 - Approche descendante (en chaînage arrière)
 - Commencer par prouver que A_1 est une conséquence logique de P
 - Chercher une clause règle de P dont l'atome de tête s'unifie avec A_1
 $C_0: -C_1, \dots, C_m$ telle que $\text{upg}(A_1, C_0) = s$
 - Le nouveau but à prouver est
 $\text{But} = s(C_1), \dots, s(C_m), s(A_2), \dots, s(A_n)$
 - Répéter le processus jusqu'à ce que le but à prouver devienne vide



CHAPITRE V

Programmation logique

procedure prouver(But: liste d'atomes logiques)

si But = [] **alors**

/* le but initial est prouvé */

/* afficher les valeurs des variables du but initial */

sinon soit But = [A1, A2, .., An]

pour toute clause (C0 :- C1, C2, .., Cm) du programme:
(où les variables ont été renommées)

s=upg(A1,C0)

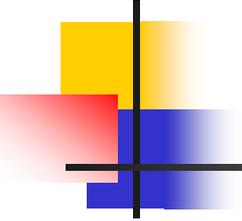
si s≠échec **alors**

prouver([s(C1), s(C2), .. s(Cm), s(A2), .. s(An)])

finsi

finpour

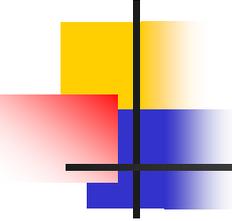
finsi



CHAPITRE V

Programmation logique

- Quand on pose une question à l'interprète Prolog, celui-ci exécute dynamiquement la procédure **prouver** :
 - L'arbre constitué de l'ensemble des appels récursifs à la procédure **prouver** est appelé *arbre de recherche*
- la stratégie de recherche dépend :
 - d'une part de l'ordre de définition des clauses dans un paquet : **si plusieurs clauses peuvent être utilisées pour prouver un atome logique, on considère les clauses selon leur ordre d'apparition dans le paquet**
 - et d'autre part de l'ordre des atomes logiques dans le corps d'une clause : **on prouve les atomes logiques selon leur ordre d'apparition dans la clause**



CHAPITRE V

Programmation logique

Sémantique opérationnelle : exemple 1

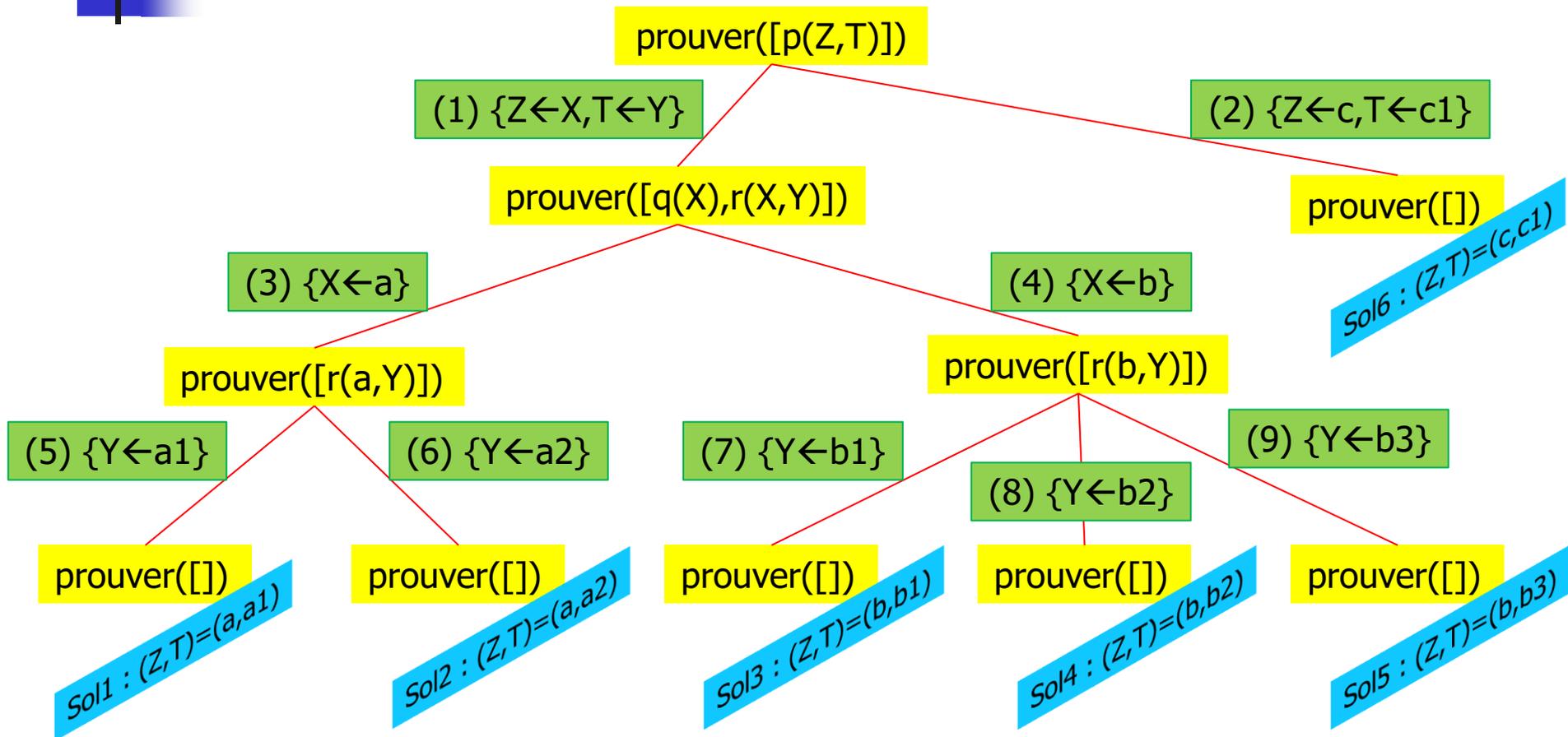
Programme :

$p(X,Y) :- q(X), r(X,Y).$	(1)
$p(c,c1).$	(2)
$q(a).$	(3)
$q(b).$	(4)
$r(a,a1).$	(5)
$r(a,a2).$	(6)
$r(b,b1).$	(7)
$r(b,b2).$	(8)
$r(b,b3).$	(9)

Question : $? p(Z,T).$

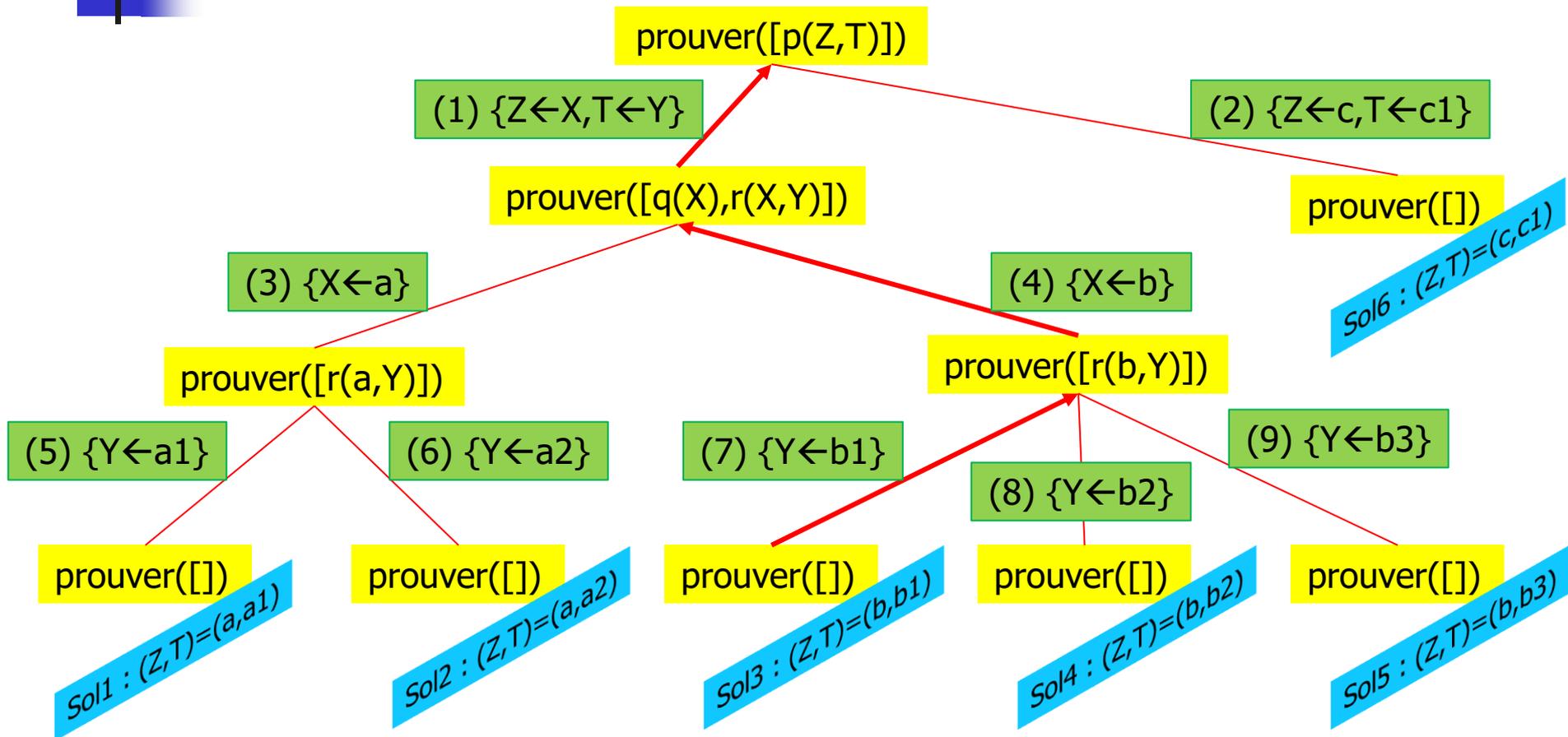
CHAPITRE V

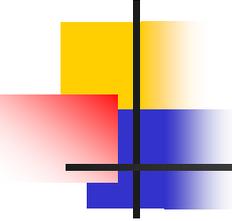
Programmation logique



CHAPITRE V

Programmation logique





CHAPITRE V

Programmation logique

Sémantique opérationnelle :

La coupure : élagage de branches de l'arbre de recherche

- **Signification opérationnelle de la coupure**

- La coupure, aussi appelée "cut", est notée !. La coupure est un prédicat sans signification logique (la coupure n'est ni vraie ni fausse), utilisée pour "couper" des branches de l'arbre de recherche
- La coupure est **toujours "prouvée" avec succès** dans la procédure prouver. La "preuve" de la coupure a pour effet de bord de modifier l'arbre de recherche :

elle coupe l'ensemble des branches en attente créées depuis l'appel de la clause qui a introduit la coupure



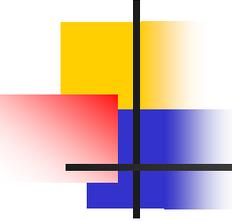
CHAPITRE V

Programmation logique

Sémantique opérationnelle :

La coupure : exemples d'utilisation

- Recherche de la première solution
- Exprimer le déterminisme
- La négation
- Le "si-alors-sinon"



CHAPITRE V

Programmation logique

Sémantique opérationnelle : exemple 2

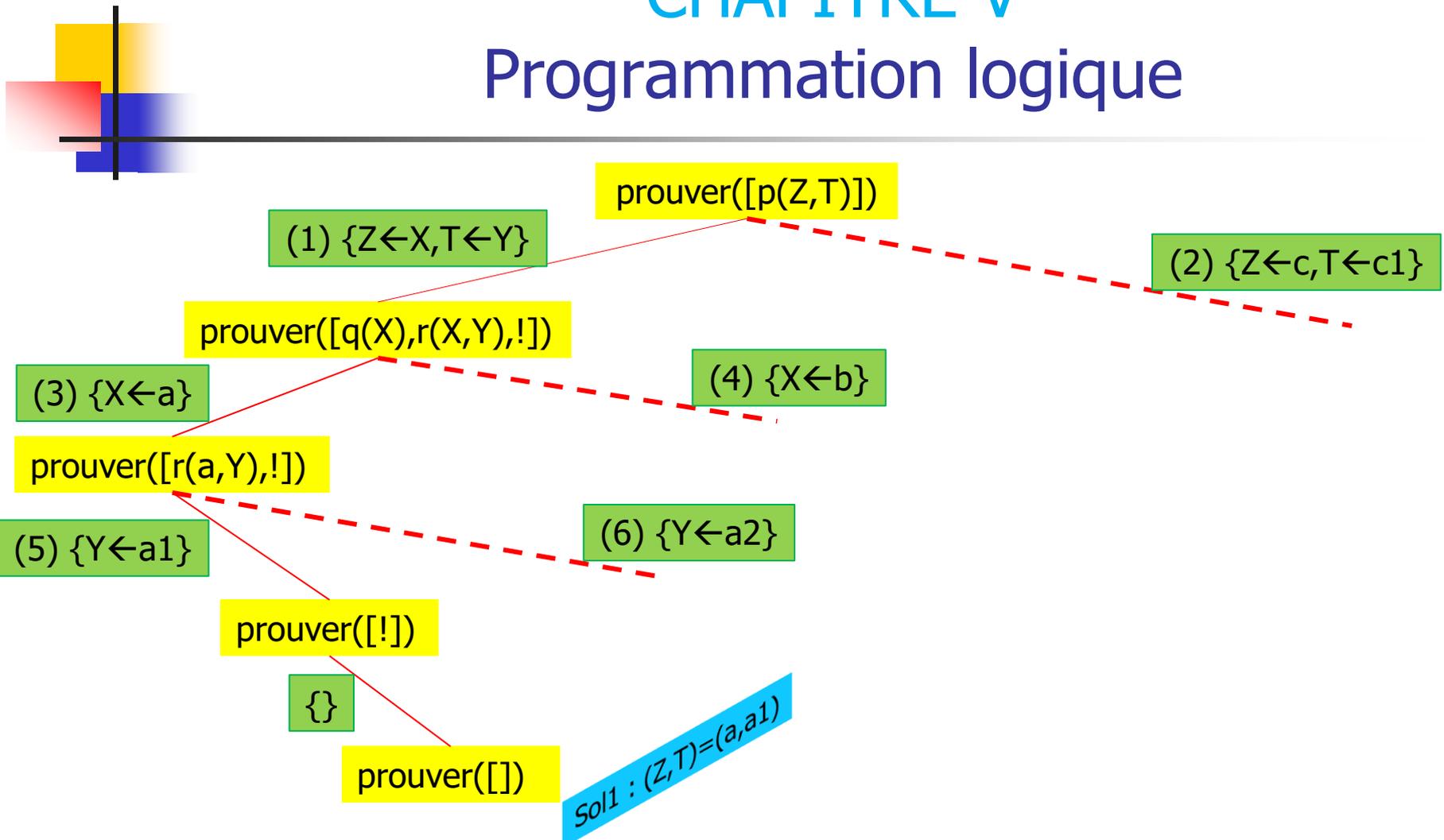
Programme :

$p(X,Y) :- q(X), r(X,Y), !.$	(1)
$p(c,c1).$	(2)
$q(a).$	(3)
$q(b).$	(4)
$r(a,a1).$	(5)
$r(a,a2).$	(6)
$r(b,b1).$	(7)
$r(b,b2).$	(8)
$r(b,b3).$	(9)

Question : $? p(Z,T).$

CHAPITRE V

Programmation logique





CHAPITRE V

Programmation logique

Sémantique opérationnelle : exemple 3

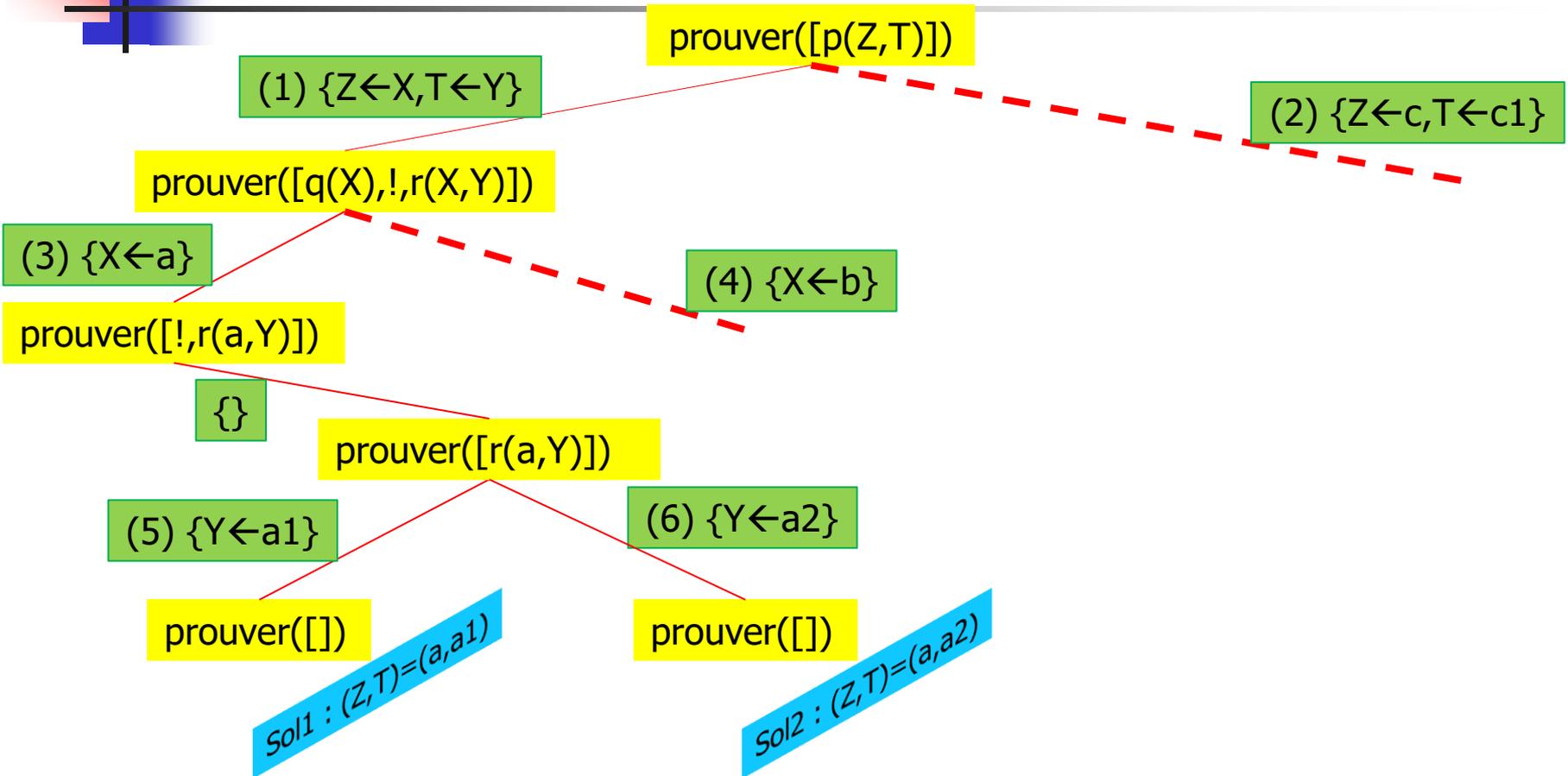
Programme :

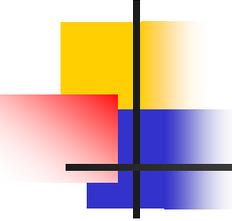
$p(X,Y) :- q(X), !,r(X,Y).$	(1)
$p(c,c1).$	(2)
$q(a).$	(3)
$q(b).$	(4)
$r(a,a1).$	(5)
$r(a,a2).$	(6)
$r(b,b1).$	(7)
$r(b,b2).$	(8)
$r(b,b3).$	(9)

Question : $? p(Z,T).$

CHAPITRE V

Programmation logique





CHAPITRE V

Programmation logique

Sémantique opérationnelle : exemple 4

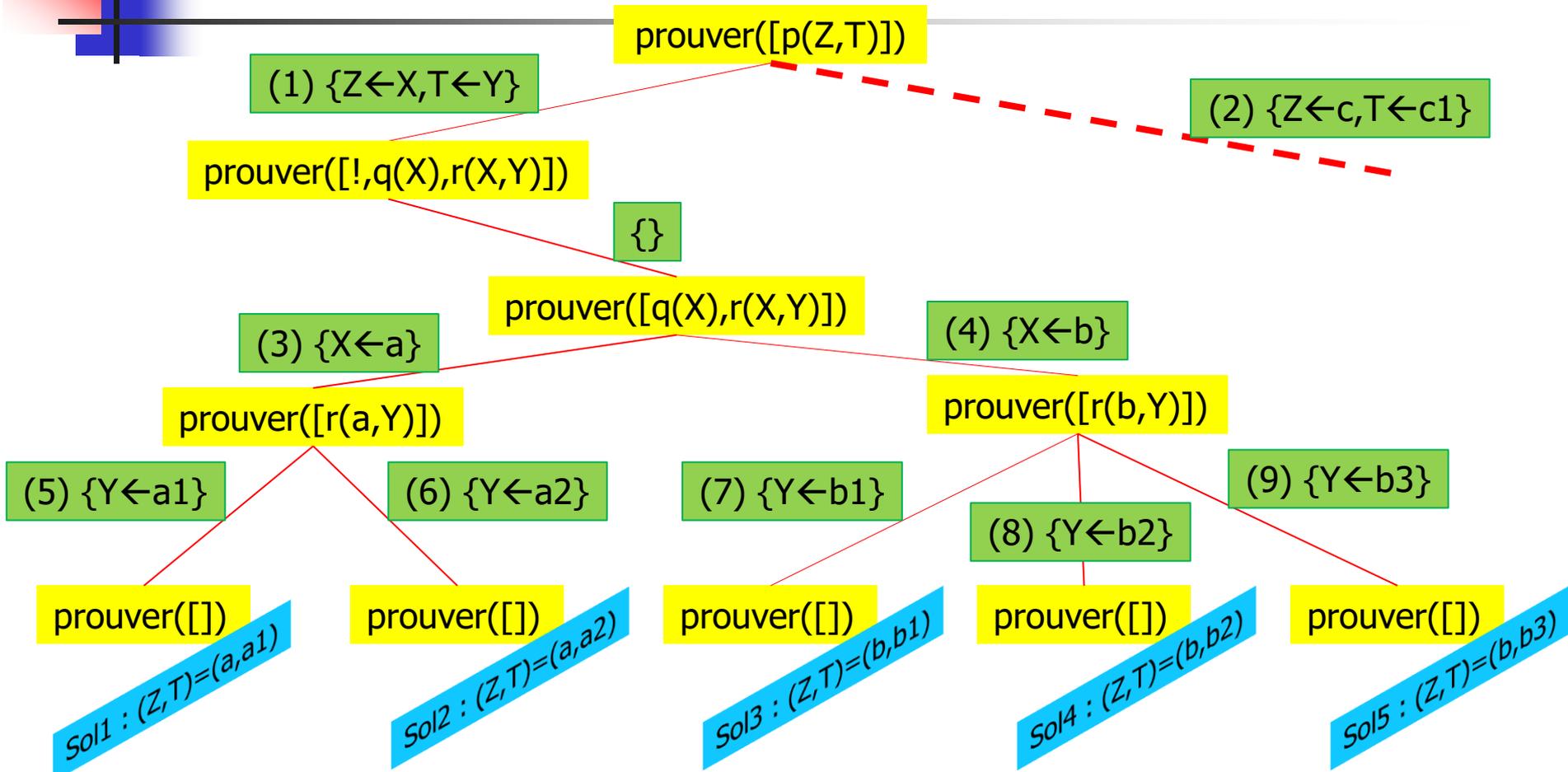
Programme :

$p(X,Y) :- !,q(X), r(X,Y).$	(1)
$p(c,c1).$	(2)
$q(a).$	(3)
$q(b).$	(4)
$r(a,a1).$	(5)
$r(a,a2).$	(6)
$r(b,b1).$	(7)
$r(b,b2).$	(8)
$r(b,b3).$	(9)

Question : $? p(Z,T).$

CHAPITRE V

Programmation logique



CHAPITRE V

Programmation logique par contraintes

Programmation logique → Programmation logique par contraintes :

- **Fixer une structure d'interprétation S représentant l'univers du discours**
- **Distinguer dans un programme les deux choses suivantes :**
 - **Le langage de contraintes défini sur S , supposé décidable**
 - **Le langage des prédicats défini par des formules logiques**
- **Les formules logiques permettant de définir des prédicats sont restreintes aux clauses de Horn de la forme :**

$A:-c_1, \dots, c_m, A_1, \dots, A_n.$

- **CLP(S)**