CHAPITRE V

Approche de développement des systèmes experts

Cycle de développement

Cycle de développement

- **Phase 1 : Initialisation du projet**
  - Définition du problème
  - Analyse des besoins
  - Evaluation des solutions alternatives
  - Vérifier si l'approche SE est appropriée
  - Considérer les problèmes de gestion

Cycle de développement

- **Phase 2 : Conception et analyse du système**
  - Définir un design conceptuel
  - Définir une stratégie de développement
  - Identifier les sources de connaissance (knowledge) et assurer une coopération
  - Sélectionner les ressources informatiques
  - Assurer un coût bénéfique de l'analyse

Cycle de développement

- **Phase 3 : Prototypage**
  - Construire un petit prototype
  - Le tester, améliorer et étendre
  - Analyser le faisabilité
  - Compléter le design

Cycle de développement

- **Phase 4 : Développement du système**
  - Construire la base de connaissances
  - Tester, évaluer et améliorer cette base
  - Planifier l'intégration

Cycle de développement

- **Phase 5 : Implémentation**
  - Assurer l'acceptation des utilisateurs
  - Installer et déployer le système
  - Orienter et entrainer les utilisateurs
  - Assurer la sécurité
  - Fournir la documentation

Cycle de développement

- **Phase 5 : Post-Implémentation**
  - Maintenance
  - Mise à jour
  - Evaluation périodique

Conception et implémentation

- **Conception conventionnelle (programmation du moteur, etc…)**
  - Langage de programmation (C, C#, Java, Python, ..)
- **Utilisation de générateurs de SE (moteurs d'inférences nus)**
  - M1, OPS5, MP-LRO
- **Usage de langages de programmation pour l'IA**
  - Langages fonctionnels : LISP, ML, CAML
  - Langages logiques : PROLOG
  - Interpréteurs : CLIPS, JESS
  - Outils graphiques : ES Builder

Caractéristiques

- Prolog est logique : Le programme peut être vu comme une suite d'axiomes qui décrivent un problème.

- Prolog utilise une véritable notion de variables : les variables Désignent des objets non connus en recherche. Elles sont gérées par le système qui leur attribut des valeurs et il les défait.

- Prolog est non déterministe : Les fonctions définies peuvent avoir plusieurs valeurs, et la recherche de ces valeurs, c'est Prolog qu'il l'a fait en revenant en arrière là où il faut.

Raisonnement PROLOG

Structure d'un programme

Un programme prolog est composé de trois parties : Faits, Règles et requêtes (buts)

Exemple :

%faits

```
personne(léon,35).
personne(lucie,27).
personne(louis,40).
personne(pauline,9).
personne(luc,27).
```

%règles

```
individu(x) :- personne(x,_).
majeur(x) :- personne(x,y),y>=18.
mineur(x) :- personne(x,y),y<18.
```

Implémentation (requêtes)

individu(pauline).
→true

individu(jacque).
→ false

personne(X,27).
→ X=lucie;
→ X=luc.

personne(louis,X)
→ X=40.

mineur(X).
→ X=pauline.

Propriétés de Prolog

Remarques :

- Le mécanisme de Prolog est **correcte :**

Ne donne que des réponses logiquement correctes.

- Le mécanisme de Prolog est **complet :**

Lorsqu'il s'arrête, on peut être sure qu'il a donné toutes les réponses qu'il faut.

Raisonnement PROLOG

Sens des règles de la section *clauses (règles)*

- Clause simples (faits):

Exemple : Personne(léon,35).

Se lit : *Léon est une personne de 35 ans.*

- Clause complètes : qui contiennent le symbole :-

Elles correspondent à des affirmations générales comprenant des variables.

- Le symbole '**:-** ' : Signifie **Si**, peut être remplacé par ***if.***
- Le symbole ', ' : Signifie ***ET***, peut être remplacé par ***and.***
- Les variables : représentent des objets quelconques

  Exemple : x,y,Personne
- Le symbole ' _ ' : variable anonyme (signifie ∀)

Raisonnement PROLOG

Sens des règles de la section *clauses (règles)*

- **Signification :**

- personne(léon,35). Signifie :

il est vrai que Léon est une personne de 35 ans **OU**
Le but Personne(Léon,35) est satisfait.

- individu(x):-personne(x,_). Signifie :

x est un individu s'il existe une donnée (fait) de personne(x,_). **OU**
Le but individu(x) est satisfait pour chaque x tel que Personne(x,_) est satisfait.

- mineur(x) :- personne(x,y),y<18. signifie :

x est un mineur si x est une personne d'âge y et que y<18 **OU**
Le but Mineur(x) est satisfait pour chaque x tel que le but Personne(x,y) est satisfait avec un y<18.

Raisonnement PROLOG

Sens des règles de la section *clauses*

- **Autres précisions :**

- Le symbole ';' peut être utilisé dans le corps d'une clause, il signifie ***OU*** :

a(x):-b(x);c(x). Cette règle peut être écrite :

a(x):-b(x).

a(x):-c(x).

- Le sens logique d'une clause est celui d'une implication entre le corps de la clause et sa tête et dont les variables sont toutes quantifiées universellement.

Exemples :

a(x):-b(x),c(x). signifie: $\forall$ x b(x) $\wedge$ c(x) $\Rightarrow$ a(x)

a(x):-b. signifie : b $\Rightarrow \forall$ x a(x)

a(x):-b(x,y). signifie : $\forall$ x ($\exists$ y b(x,y)) $\Rightarrow$ a(x)

Raisonnement Prolog

Principe de raisonnement :

1- S'occuper de but le plus à gauche.

2- Pour satisfaire le but dont on s'occupe :

+ Rechercher dans l'ordre du programme, la première règle non déjà essayée à ce point et dont la tête est compatible avec le but.
+ Puis remplacer dans la liste des buts, le premier but par le corps de cette règle en faisant toutes les substitutions.

3- A chaque fois qu'un blocage à lieu (échec), revenir en arrière à la plus récentes liste des buts où une règle pourrait être utilisée et ne la pas était (retour en arrière : **backtracking**).

Remarque : ce type de recherche systématique se présente très bien avec les arbres. Chaque nœud est une liste de buts et le parcours est selon la technique :

Gauche – Droite – En profondeur d'abord avec Backtracking

Exemple:

R1: Est-de-bonne-humeur(x):-A-de-l-argent(x),Est-en-vacances(x),Il-y-a-du-soleil.

R2: Est-de-bonne-humeur(x):-Réussit-dans-le-travail(x),Réussit-dans-sa-famille(x).

R3: A-de-l-argent(Jean).

R4: A-de-l-argent(Alain).

R5: Est-en-vacances(Jean):-On-est-en(Aout).

R6: Est-en-vacances(Alain):-On-est-en(Juillet).

R7: On-est-en(Juillet).

R8: Il-y-a-du-soleil:-On-est-en(Aout).

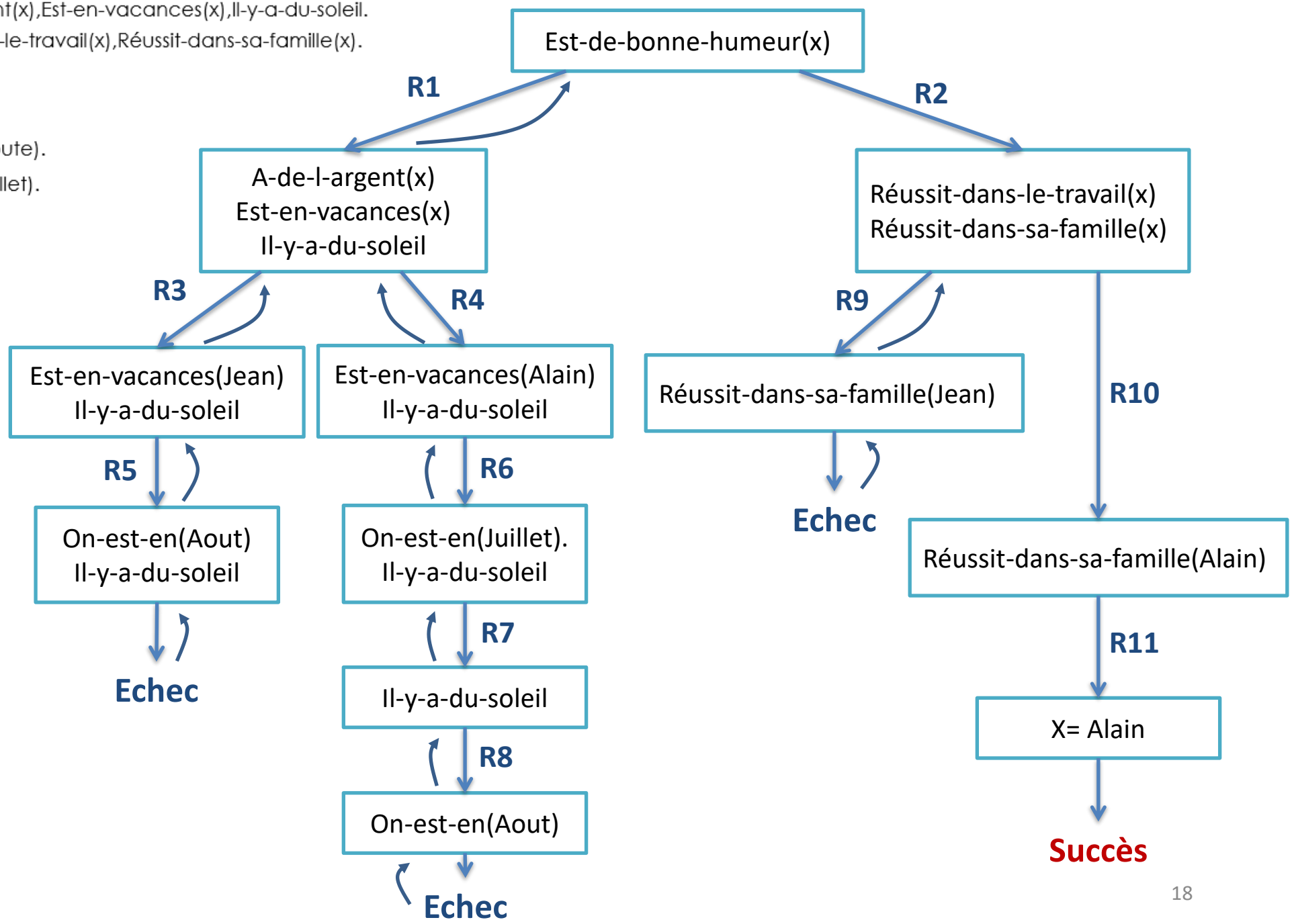
R9: Réussit-dans-le-travail(Jean).

R10: Réussit-dans-le-travail(Alain).

R11: Réussit-dans-sa-famille(Alain).

But: Est-de-bonne-humeur(x) ?

R1: Est-de-bonne-humeur(x):-A-de-l-argent(x),Est-en-vacances(x),Il-y-a-du-soleil.
R2: Est-de-bonne-humeur(x):-Réussit-dans-le-travail(x),Réussit-dans-sa-famille(x).
R3: A-de-l-argent(Jean).
R4: A-de-l-argent(Alain).
R5: Est-en-vacances(Jean):-On-est-en(Aoute).
R6: Est-en-vacances(Alain):-On-est-en(Juillet).
R7: On-est-en(Juillet).
R8: Il-y-a-du-soleil:-On-est-en(Aoute).
R9: Réussit-dans-le-travail(Jean).
R10: Réussit-dans-le-travail(Alain).
R11: Réussit-dans-sa-famille(Alain).



Exercice :

Refaire le raisonnement en remplaçant *On-est-en(Juillet)* par *On-est-en(Aoute)*

Raisonnement PROLOG

Traitement de la négation

- A la rencontre des négations, Prolog raisonne avec le principe : **Négation par l'échec**.

- Si le raisonnement Prolog appliqué au but P donne un succès Alors le but ¬P est considéré amenant à un échec.
- Si le raisonnement Prolog appliqué au but P donne un échec Alors le but ¬P est considéré amenant à un succès.

- A chaque fois qu'un but de la forme ¬P doit être traité :

- Construire un arbre de raisonnement auxiliaire ayant come racine P.
- Si cet arbre donne un succès, revenir à l'arbre principal en marquant ¬P avec un échec.
- Si cet arbre donne un échec, revenir à l'arbre principal en marquant ¬P par succès.

Raisonnement PROLOG

Négation par l'échec

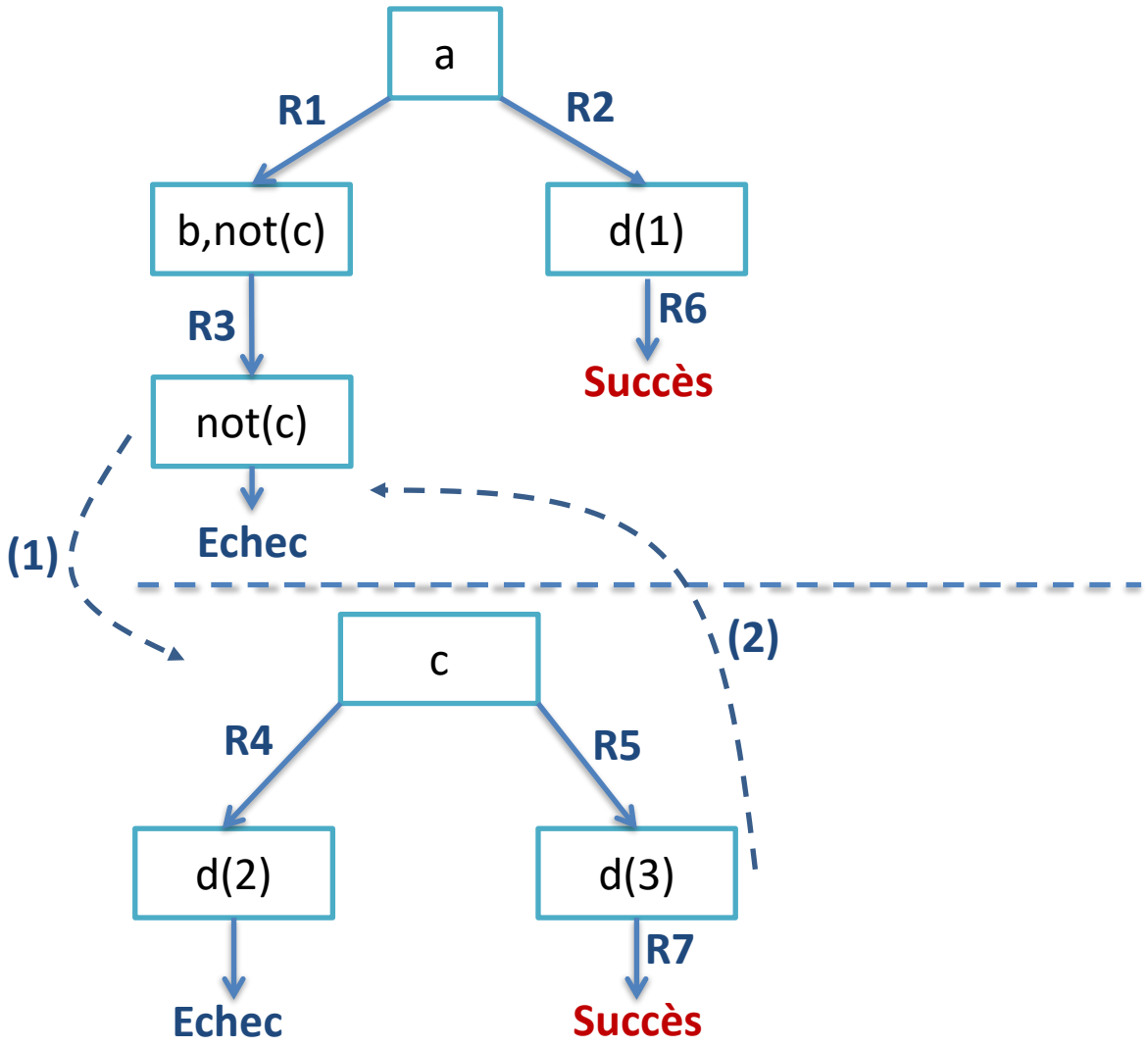
Raisonnement PROLOG

Négation par l'échec

Exemple 1 :

R1: a:-b,not(c).
R2: a:-d(1).
R3: b.
R4: c:-d(2).
R5: c:-d(3).
R6: d(1).
R7: d(3).

Question : a ?



Raisonnement PROLOG

Négation par l'échec

Exemple 2 :

R1: a:-b,not(c).
R2: a:-d(1).
R3: b.
R4: c:-d(2).
R5: c:-d(3).
R6: d(1).
R7: d(3).
R8: e:-d(1),not(a).
R9: e:-not d(2),d(3).

Question : e ?

Raisonnement PROLOG

La coupure « ! »

Le **cut « ! »** est un prédicat très spécial, il est toujours satisfait (vrai), et il agit sur la façon dont l'arbre de raisonnement est parcouru.

- Dans le parcours de l'arbre de raisonnement, si lors d'une nécessité on tombe sur une liste de buts commençant par un cut « ! » :

- Il faut remonter directement au dessus du but qui a introduit ce cut.
- Lorsque le cut s'efface, il coupe tous les choix partant du but qui a introduit le cut.

Raisonnement PROLOG

La coupure « ! »

Exemple : A ?

A:-B,C.
A:-D.
A:-E.
B:-F,!.
B:-H.
B:-I.
E.
F:-G.
F:-J.
F:-K.
G.
G:-L.
G:-M.