EXERCICE

Enoncés :

A1 :Un étudiant qui a la grippe ne doit pas venir en cours

A2: Un étudiant qui a de la fièvre et qui tousse a la grippe

A3: Un étudiant qui a une température supérieure à 38 a de la fièvre

A4: Ali tousse et a une température supérieure à 38

B: Ali ne doit pas venir en cours

Peut-on établir B à partir de A1, A2, A3 et A4 (utiliser la résolution)?

Prédicats

grippe(x) : x a la grippe

pasvenir(x) : x ne doit pas venir en cours

fievre(x) : x a de la fièvre

tousse(x) : x tousse

temp(x,t) : x a la température t

sup(t,T) : t est supérieure à T

Modélisation

A1: $\forall x$ (grippe(x)$\Rightarrow$ pasvenir(x))

A2: $\forall x$ ((fievre(x) $\wedge$ tousse(x)) $\Rightarrow$ grippe(x))

A3: $\forall x \forall t$ ((temp(x,t) $\wedge$ sup(t,38)) $\Rightarrow$ fievre(x))

A4: tousse(Ali) $\wedge \exists$ t (temp(Ali,t) $\wedge$ sup(t,38))

B: pasvenir(Ali) $\neg$ B= $\neg$ pasvenir(Ali)

{A1,A2,A3,A4,$\neg$ B} a un modèle ou pas ?

EXERCICE

Prenex

∀x (¬ grippe(x) v pasvenir(x))

∀x (¬ fievre(x) v ¬ tousse(x) v grippe(x))

∀x ∀t (¬ temp(x,t) v ¬ sup(t,38) v fievre(x))

∃ t (tousse(Ali) ^ (temp(Ali,t) ^ sup(t,38))
¬ pasvenir(Ali)

cste a

tousse(Ali) ^ (temp(Ali,a) ^ sup(a,38)

Passage à la forme causale

C1: ¬ grippe(x) v pasvenir(x)

C2: ¬ fievre(x) v ¬ tousse(x) v grippe(x)

C3: ¬ temp(x,t) v ¬ sup(t,38) v fievre(x)

C4: tousse(Ali)
C5: temp(Ali,a)
C6: sup(a,38)
C7: ¬ pasvenir(Ali)

Résolution

C8: ¬ grippe(Ali) Res(C1,C7)

C9: ¬ fievre(Ali) v ¬ tousse(Ali) Res(C8,C2)

C10: ¬ tousse(Ali)v ¬ temp(Ali,t)v ¬ sup(t,38)
Res(C9,C3)

C11: ¬ temp(Ali,t) v ¬ sup(t,38) Res(C10,C4)

C12: ¬ sup(a,38) Res(C11,C5)

C13: ? Res(C12,C6)

Conclusion :

B est conséquence de {A1,A2,A3,A4}

B est établi à partir de {A1,A2,A3,A4}

EXERCICE

Enoncés :

A1 :pour tout crime, il y a quelqu'un qui l'a commis

A2: seuls les gens malhonnêtes commettent des crimes

A3: ne sont arrêtés que les gens malhonnêtes

A4: les gens malhonnêtes arrêtés ne commettent pas de crimes

A5: il y a des crimes

B: il y a des gens malhonnêtes non arrêtés

Peut-on établir B à partir de A1, A2, A3 , A4 et A5 (utiliser la résolution)?

Prédicats

ar(y) : y est arrêté

ma(y): y est malhonnête

co(y,x) : y commet x

cr(x) : x est un crime

Modélisation

A1: $\forall x\ (cr(x) \Rightarrow \exists y\ co(y,x))$

A2: $\forall y\ \forall x\ ((cr(x) \wedge co(y,x)) \Rightarrow ma(y))$

A3: $\forall y\ (ar(y) \Rightarrow ma(y))$

A4: $\forall y\ ((ma(y) \wedge ar(y)) \Rightarrow \neg \exists x\ (cr(x) \wedge co(y,x)))$

A5: $\exists x\ cr(x)$

B : $\exists y\ (ma(y) \wedge \neg ar(y))$

{A1,A2,A3,A4,A5,¬ B} a un modèle ou pas ?

EXERCICE

Prenex

A1: ∀x ∃ y (¬ cr(x) v co(y,x))

A2: ∀y ∀x (¬ cr(x) v¬ co(y,x) v ma(y))

A3: ∀y (¬ ar(y) v ma(y))

A4: ∀y ∀ x(¬ ma(y) v¬ ar(y) v¬ cr(x)v ¬ co(y,x))

A5: ∃ x cr(x)

¬ B : ∀y (¬ ma(y) v ar(y)))

Mise sous forme de clauses

f1: ¬ cr(x) v co(f(x),x))

f2: ¬ cr(x) v ¬ co(y,x) v ma(y))

f3: ¬ ar(y) v ma(y)

f4: ¬ ma(y) v¬ ar(y) v¬ cr(x) v¬ co(y,x)

f5: cr(a)

f6: ¬ ma(y) v ar(y)

Résolution

f7: co(f(a),a)) res(f5,f1) avec θ=(x|a)

f8: ¬ cr(a) v ma(f(a)) res(f7,f2) (x|a)(y|f(x))

f9: ma(f(a)) res(f8,f5)

f10: ¬ ma(f(a)) v¬ ar(f(a)) v¬ cr(a) res(f7,f4) (x|a)(y|f(x))

f11: ¬ ma(f(a)) v¬ ar(f(a)) res(f10,f5)

f12: ¬ ar(f(a)) res(f11,f9)

f13: ¬ ma(f(a)) res(f12,f6) (y|f(a))

f14: □ res(f13,f9)

B est conséquence de {A1,A2,A3,A4,A5}

B est établi à partir de {A1,A2,A3,A4,A5}

Vers Prolog

➢Prolog peut être considéré à la fois comme un système de démonstration automatique et comme un langage de programmation basé sur la logique des prédicats.

➢L'utilisateur définit une base de connaissances

➢l'interpréteur Prolog utilise cette base de connaissances pour répondre à des questions

programmation logique	**programmation impérative**
ensemble de règles	programme
preuve	exécution
question (but)	appel de procédure
substitution, unification	passage de paramètres

CLAUSES DE HORN ET PROLOG

➢On appelle clause de Horn toute clause de la Logique du 1er ordre ayant au plus une formule atomique (littéral) positive

Type1	⌉ **H1** ∨ ⌉ **H2** ∨ ∨ ⌉ **Hn** ∨ **C**
Type 2	C
Type 3	⌉ H1 ∨ ⌉ H2 ∨ ∨ ⌉ Hn

➢ PROLOG est formé sur le sous-ensemble de la Logique du 1er ordre formé par les clauses de Horn.

Type1: règles	**Implication : production de nouvelles connaissances** **H1 ^ H2 ^ ^ Hn → C**
Type 2: faits	Affirmation de connaissances supposées vraies à priori C
Type 3: buts	Question à laquelle on souhaite répondre par déduction C1 ^ C2 ^ ^ Cn

CLAUSES DE HORN ET PROLOG

- **Clauses de Horn**

Règles	**H1 ^ H2 ^ ^ Hn → C**
Faits	C
Questions	C? C1 ^ C2 ^ ^ Cn ?

- **Prolog**

Règles	**C : - H1 , H2 , , Hn .**
Faits	C.
Questions	?- C. ?- C1, C2, , Cn.

Vers Prolog

➢**Syntaxe**

- Termes
  Atomes (constantes) : objets définis du problème
  Variables : objets indéfinis du problème
- Prédicats : jugement ou relations logiques élémentaires

➢**Clauses**

- Faits : connaissances de base établies à priori
  fait.
  pere(ali, omar).
  Clause de Horn réduite à un littéral positif
- Règles : règle de manipulation de la connaissance
  conclusion :- premisse.
  conclusion :-premisse_1,..,premisse_n.
  grandpere(X,Y) :- pere(X,Z), pere(Z,Y).
  Clause de Horn complète
- Questions (buts) : problème à résoudre
  ?-but.
  ?-but_1, ... but_n.
  pere(ali,X), mere(sarrah,X).
  Clause de Horn sans littéral positif

Vers Prolog

Stratégie de résolution Prolog

➢ **Chaînage arrière résolution de buts**

Résolution d'un but B : Recherche d'une clause ayant B en conclusion.

- fait B. Le but B est démontré
- règle B :- P1, ..., Pn B remplacé par P1, … Pn

➢ **Profondeur d'abord**

Si un but se décompose en plusieurs sous-buts

B :- P1, ..., Pn

Prolog tente toujours de résoudre P1 et ainsi de suite

➢**Régime par tentative**

Si plusieurs clauses peuvent résoudre le même but :

B :- P1, ..., Pp B :- R1, ..., Rr

- L'interpréteur choisit la 1ère clause du programme
- En cas d'échec, on effectue un retour-arrière
- S'il ne reste aucune règle utilisable, on remonte au point de choix précédent.
- Si aucune possibilité n'est envisageable au final : échec

Vers Prolog

Stratégie de résolution Prolog

➢**Résolution Prolog avec variables : unification**

Résolution d'un but B par décomposition en sous-buts :

- Recherche d'une clause ayant une conclusion s'unifiant avec le but.
- Les sous-buts obtenus sont les substitués, par l'unificateur trouvé, des prémisses de la clause

➢Si la formule à prouver contient des variables, Prolog cherche toutes les valeurs des variables pour lesquelles la formule est prouvable.

Exemple. Si les formules données sont :

- p(a, b)
- p(b, b)
- p(c, a)
- q(X) :- p(X, b)

et si la formule à prouver est q(Y)

Prolog fournira comme réponse Y = a et Y = b

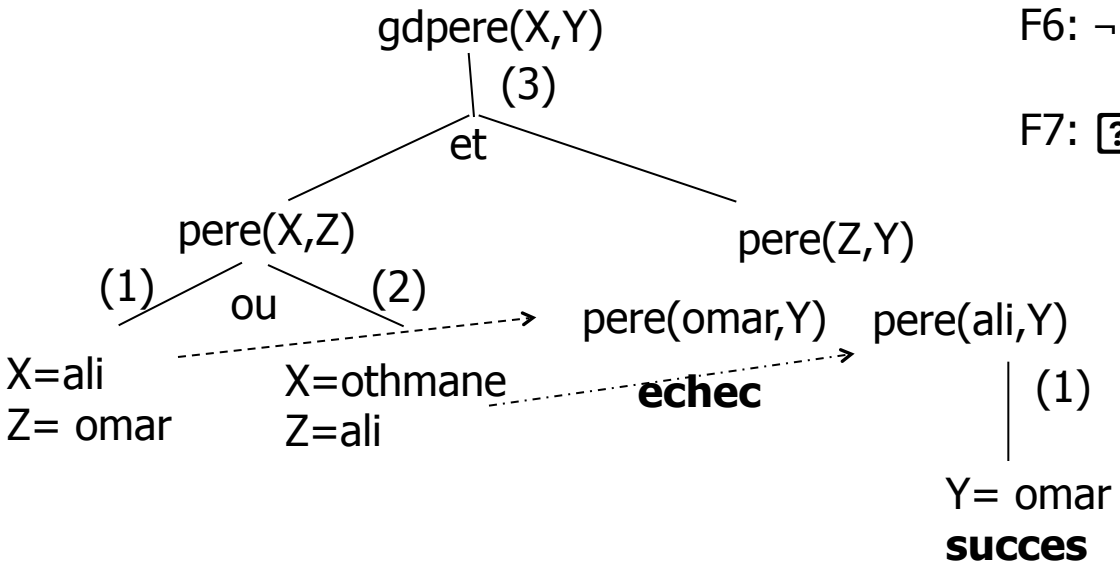
En effet : q(a) et q(b) sont prouvables respectivement à partir de :

p(a, b), p(a, b) → q(a) et p(b, b), p(b, b) → q(b)

Vers Prolog

Exemple

(1) pere(ali, omar).
(2) pere(othmane, ali).
(3) gdpere(X,Y) :- pere(X,Z), pere(Z,Y).
Question : gdpere(X,Y).
Réponse : X=othmane, Y= omar



f1:pere(ali, omar)
f2:pere(othmane, ali)
f3:¬pere(X,Z)v ¬pere(Z,Y)v gdpere(X,Y)
F4:¬gdpere(X,Y)
F5: ¬pere(X,Z)v ¬pere(Z,Y) (res f3 et f4)
F6: ¬pere(ali,Y) (res f2 et f5)
(X|othmane)(Z|ali)
F7: □ res f1 et f6 (Y| omar)

Vers Prolog

Exemple

Les chevaux sont plus rapides que les chiens, et il y a un lévrier qui est plus rapide que tous les lapins. On sait qu'Harry est un cheval et que Ralph est un lapin , et que les lévriers sont des chiens.
Déduire qu'Harry est plus rapide que Ralph.

Modélisation

- les chevaux sont plus rapides que les chiens
- il existe un lévrier plus rapide que tout lapin
- Harry est un cheval
- Ralph est un lapin
- les lévriers sont des chiens

Vers Prolog

Exemple

Traduction

∀ x ∀ y cheval(x) ^ chien(y) ⇨ rapide(x, y)	**rapide(X, Y) :- cheval(X), chien(Y).**
∃ y lévrier(y) ^ (∀ z lapin(z) ⇨ rapide(y, z))	rapide(a, Z) :- lapin(Z). lévrier(a).
cheval(Harry)	cheval(Harry).
lapin(Ralph)	lapin(Ralph).
∀y lévrier(y) ⇨ chien(y)	chien(Y):-lévrier(Y).
∀x ∀ y ∀ z rapide(x, y) ^ rapide(y, z) ⇨ rapide(x, z)	rapide(X, Z) :- rapide(X, Y), rapide(Y, Z).

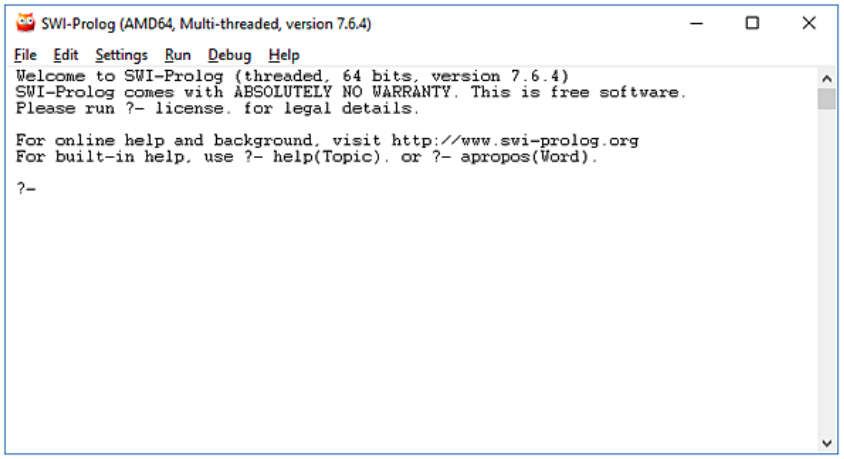
Vers Prolog

Exemple

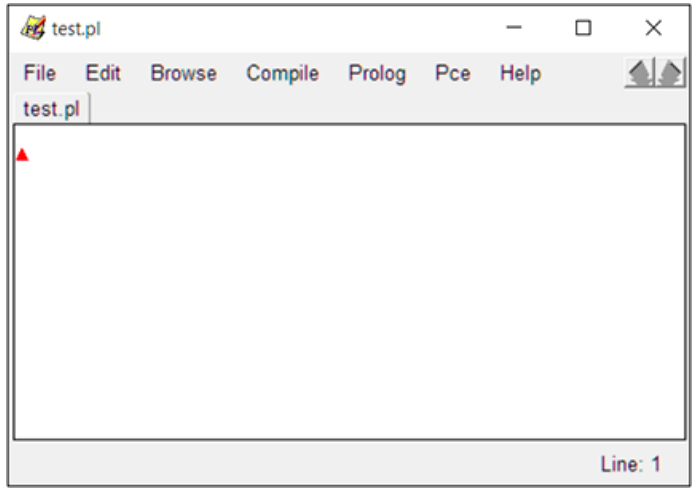
SWI-Prolog : une implémentation libre du langage Prolog.

installer SWI-Prolog
http://www.swi-prolog.org/download/stable

Au lancement du logiciel, il se présente sous Windows comme suit :



Pour créer un nouveau projet :**File** puis **New**.
Le fichier (vide initialement) contenant les règles et prédicats s'ouvre dans une autre fenêtre :



Vers Prolog

Exemple

Base de faits : (fichier texte exemple.pl)

```
animal(chien).
animal(chat).
prenom(ali).
prenom(omar).
prenom(othmane).
```

Ouvrir exemple.pl dans l'interpréteur Prolog :

```
consult(exemple).
```

```
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- prenom(ali).
true.

?- prenom(youssef).
false.

?- animal(X).
X = chien ;
X = chat.

?- prenom(X).
X = ali;
X = omar ;
X = othmane.

?-
```

SYSTÈME EXPERT

Qu'est-ce qu'un système expert ?

"Un système expert est un programme conçu pour simuler le comportement d'un humain qui est un spécialiste ou un expert dans un domaine très restreint" P.Denning (1986)

➢ Aide à la décision
Reproduire un **raisonnement en tentant d'analyser un problème** comme le ferait un **expert humain dans un domaine précis**

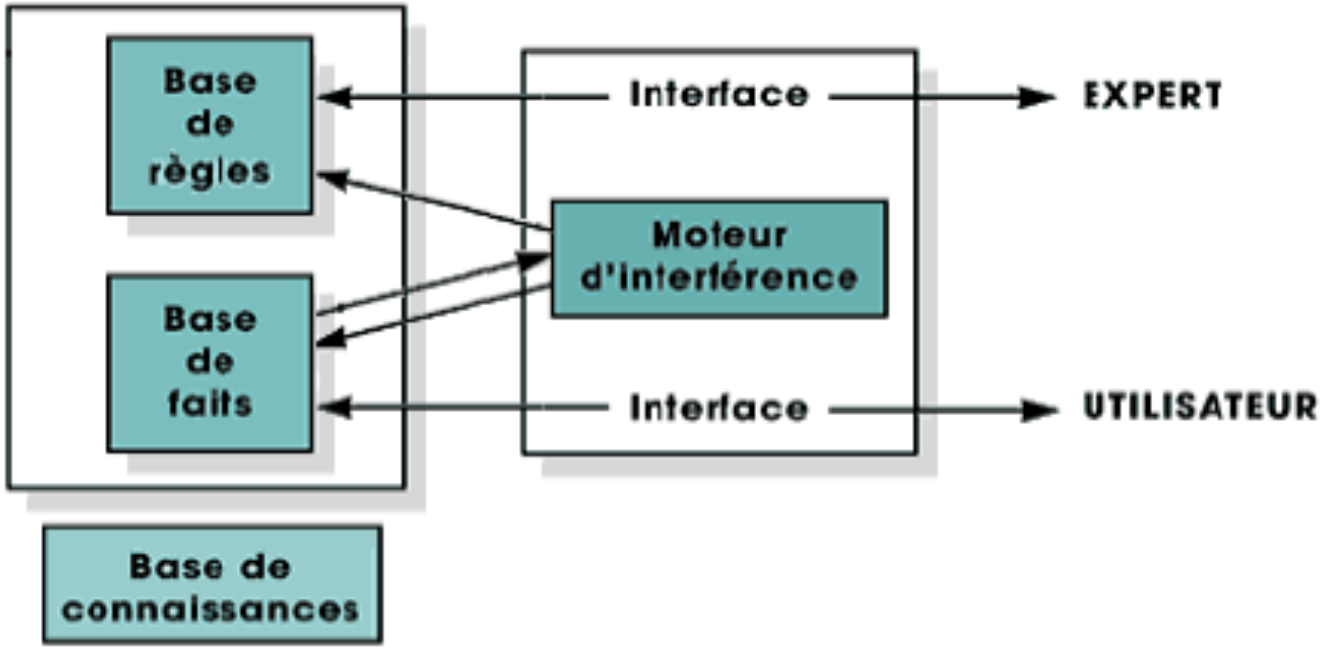
➢ Exemples :

- Recherche de solutions dans des domaines **peu formalisés**
  diagnostic médical, prescription thérapeutique , …
  ex : MYCIN "thérapie antibiotique"
- Détection de **pannes**
- **Automatisation de procédures (juridique, loi du travail, …)**
- Régulation d'échanges boursiers

…

SYSTÈME EXPERT

Composants d'un système expert



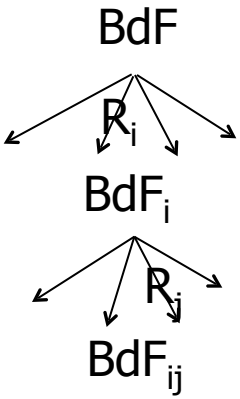
SYSTÈME EXPERT

Composants d'un système expert

Une base de connaissances

➢ Base de Faits

- contient les informations concernant le problème traité
- Elle est la mémoire de travail du système expert
- Elle est variable au cours de l'exécution
- Au début de la session, elle contient les faits initiaux. Puis, elle est complétée par les faits déduits par le moteur ou demandés à l'utilisateur.



SYSTÈME EXPERT

Composants d'un système expert

➢ Base de Règles

- Rassemble la connaissance et le savoir-faire de l'expert
- Elle peut varier au cours de l'exécution uniquement via l'interface avec l'expert
- Une règle est composée d'un ensemble de prémisses (conditions d'activation) et d'un ensemble de conclusions (actions à exécuter)

Un moteur d'inférences

- Raisonne à partir des informations contenues dans la BF et dans la BR
- effectue des déductions

Une interface utilisateur et une interface avec l'expert

- Possibilité d'évolution du SE au cours de l'exécution

SYSTÈME EXPERT

Moteur d'inférence

Les moteurs d'inférence fonctionnent presque tous en deux phases :

La phase d'évaluation

➢ le moteur d'inférence va choisir dans l'ensemble de toutes les règles un ss-ensemble qui, à priori, mérite d'être utilisé

➢ parmi le ss-ensemble de règles choisies, on ne garde que les règles dont les conditions sont effectivement satisfaites et qui formeront ce que l'on appelle l'ensemble de conflit.

➢ on résout le conflit

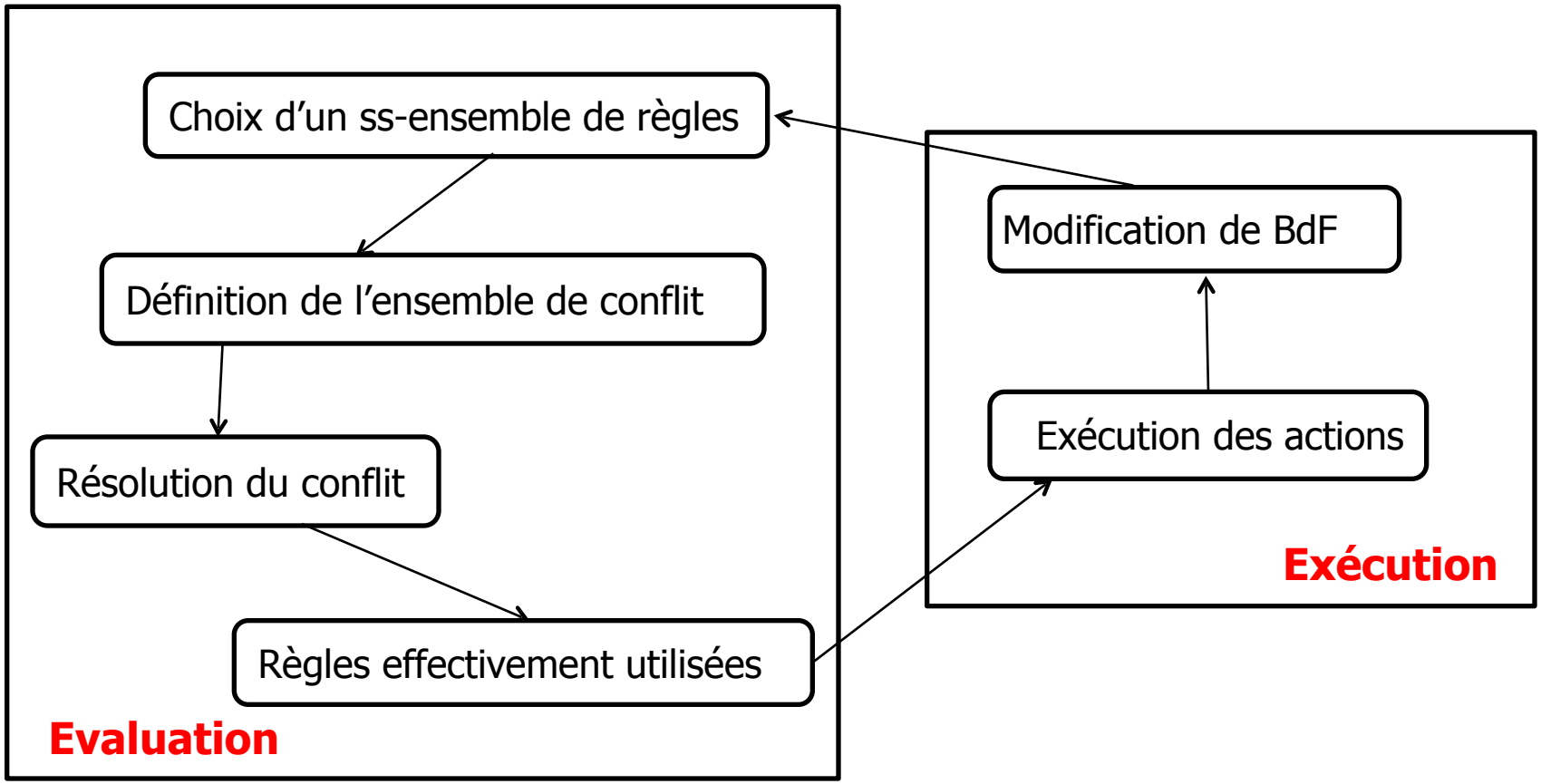
La phase d'exécution

➢ on exécute effectivement la partie action des règles déclenchées

➢ on modifie en conséquence la base de faits

SYSTÈME EXPERT

Moteur d'inférence



SYSTÈME EXPERT

Moteur d'inférence

Formalisme

➢ **Moteur d'ordre 0:**

❖logique propositionnelle
❖aucune notion de variable n'est autorisée
❖les seules actions autorisées sont l'ajout de faits
❖toute règle appliquée est éliminée
❖exemple : si l'animal est un mammifère et l'animal possède un pelage rayé de noir et de blanc
alors l'animal est un zèbre

➢ **Moteur d'ordre 0+ :**

❖moteur d'ordre 0 autorisé à modifier des faits (par exemple un compteur que l'on incrémente).
❖une règle appliquée n'est pas nécessairement effacée.
❖exemple: si la température est < 30°C et la consigne est ≤ 50 alors ajouter 2 à la consigne

➢ **Moteur d'ordre 1:**

❖Logique des prédicats du 1er ordre
❖notion de variable et unification
❖une règle appliquée reste toujours possiblement applicable
❖exemple: si X est un homme alors X est mortel

SYSTÈME EXPERT

Moteur d'inférence

Mécanisme

- le chaînage avant (modus ponens) : il est dirigé par les données
- le chaînage arrière (modus tollens) : il est dirigé par les buts
- le chaînage mixte