

## I/Stratégies de recherche :

### 1/Système expert :

Un **système expert** est un logiciel capable de simuler le raisonnement d'un expert humain dans un domaine spécifique.

Il repose sur une **représentation des connaissances**, souvent à l'aide de la **logique des prédicats**, pour formuler des règles et des faits.

### \*/Système à base de connaissances (SBC)

Un **SBC** est un système composé de deux éléments principaux :

1. Une **base de connaissances**
2. Un **moteur d'inférence**

#### ♦ Base de connaissances

Elle se divise en deux parties :

- **Base de faits** : ensemble d'assertions considérées comme **toujours vraies**. Ce sont des connaissances ponctuelles ou permanentes, utilisées comme **entrée** par le moteur d'inférence.
- **Base de règles** : ensemble de **règles de production** (de la forme *si... alors...*), permettant de **relier les faits entre eux**. Elle constitue la **seconde entrée** du moteur d'inférence.

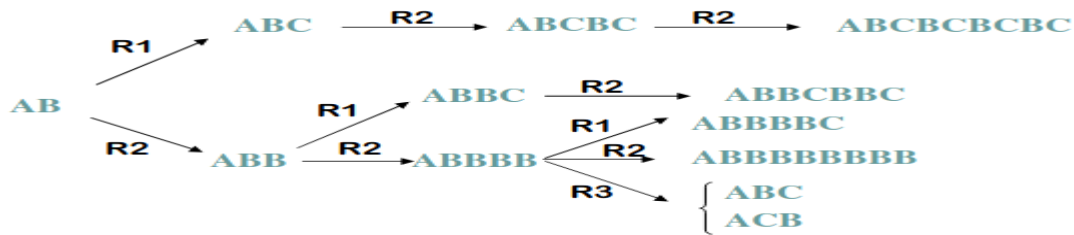
#### ♦ Moteur d'inférence

Le moteur d'inférence permet au SBC de **raisonner logiquement** à partir des faits et des règles disponibles.

Il applique des règles logiques (comme le **modus ponens**) pour **déduire de nouvelles connaissances**, qui peuvent ensuite être **ajoutées à la base de faits**.

### 2/Graphe d'états :

- C'est un **modèle abstrait** utilisé pour représenter un **problème à résoudre**.
- **Chaque nœud** = un **état possible** du système.
- **Chaque arc** = une **action** qui transforme un état en un autre.



### **\*/Espace de recherche :**

C'est l'ensemble de tous les états possibles qu'on peut atteindre à partir de l'état initial. Il est souvent représenté par un **graphe d'états**.

### **\*/Graphe de recherche :**

C'est l'ensemble des nœuds (états) parcourus depuis l'état initial (EI) jusqu'à l'état final (EF) en appliquant les règles de transition.

Il est **construit dynamiquement** par un algorithme de recherche (comme DFS, BFS, etc.) pendant la résolution du problème.

### **\*/Solution :**

C'est un chemin dans le graphe de recherche allant de l'état initial à un état final (objectif).

### **3/Heuristique :**

méthode approximative pour **orienter la recherche** d'une solution.

- ♦ Utilisée pour **gagner du temps**, surtout quand l'espace de recherche est grand.
- ♦ Ne garantit **pas toujours la solution optimale**.
- ♦ Exemple : en IA, la fonction  **$h(n)$**  estime le coût pour atteindre l'objectif depuis un état  **$n$** .
- ♦ Utilisée dans les algos comme  **$A^*$**  pour **éviter d'explorer tous les chemins**.

### **Heuristique de recherche**

Si l'objectif à atteindre est de la forme  **$xy$**  et que  **$y$**  appartient à la base de faits (BF), alors le nouveau but devient  **$X0$**  et  **$y$**  est supprimé de la base de faits.

### **Exemple :**

- Objectif : "Faire un gâteau au chocolat" ( **$xy$**  = gâteau au chocolat,  **$y$**  = chocolat).
- Base de faits initiale : BF = {farine, sucre, œufs, beurre, levure, chocolat}.

- L'objectif est d'atteindre "gâteau au chocolat". Ici, **y** = chocolat est dans la base de faits.
- Après traitement, **chocolat** est supprimé de la base de faits et le nouveau but devient **X0** = "Faire un gâteau sans chocolat".

#### 4/Représentation en Espace d'États vs Espace de Problèmes :

##### Représentation en Espace d'États

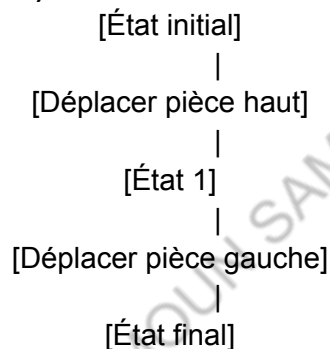
-La représentation en espace d'états consiste à chercher la solution en décrivant la situation (état) de l'univers considéré, et l'ensemble des opérateurs qui permettent de transformer cette situation.

##### Exemple : Problème du taquin (puzzle 3x3)

Chaque état représente une configuration du puzzle.

Les opérateurs : déplacer une pièce vers la case vide.

##### Graphe (représentation simplifiée)



□ Ici, on explore différents **états** jusqu'à atteindre un état **objectif**.

##### Représentation en Espace de Problèmes

-La représentation en espace de problèmes consiste à décrire le problème et à utiliser des opérateurs de transformations du problème.

Ces opérateurs décomposent le problème en plusieurs sous-problèmes plus simples (réduction de problème).

##### Exemple : Résolution d'une équation complexe

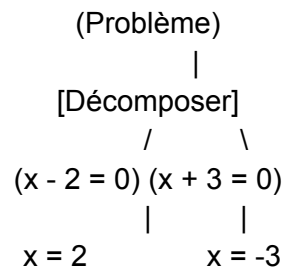
Problème : Résoudre  $(x - 2)(x + 3) = 0$

Opérateur : factorisation ou isolement

Sous-problèmes :

- Résoudre  $x - 2 = 0$
- Résoudre  $x + 3 = 0$

## Décomposition (représentation simplifiée)



- On décompose le problème principal en **sous-problèmes plus simples**.

## 5/Planification Hiérarchique :

La **planification hiérarchique** consiste à résoudre un problème en le **décomposant** en **sous-tâches** plus simples, jusqu'à atteindre des **actions primitives** exécutables.

Elle permet de **réduire la complexité** du plan global en se concentrant d'abord sur les tâches principales, puis sur les détails.

### ⚙️ STRIPS (Stanford Research Institute Problem Solver)

#### ♦ But :

Formaliser des actions à l'aide de :

- **Préconditions** : conditions nécessaires pour appliquer l'action
- **Effets (Ajouts / Retraits)** : conséquences de l'action sur l'état

#### ♦ Représentation d'un opérateur :

**Action** : Aller(x, y)

**Préconditions** :  $\hat{A}(x)$ , Route(x, y)

**Ajouts** :  $\hat{A}(y)$

**Suppressions** :  $\hat{A}(x)$

### ⚙️ ABSTRIPS (Abstract STRIPS)

#### ♦ Principe :

Extension de STRIPS utilisée pour la **planification hiérarchique**.

- On définit les actions à **différents niveaux d'abstraction**.
- Permet de planifier globalement, puis affiner localement.

## Exemple simplifié

**Objectif** : Le robot doit aller de A à C

**Faits initiaux** :

- $\hat{A}(\text{Robot}, A)$
- $\text{Route}(A, B), \text{Route}(B, C)$

**Opérateurs (STRIPS)** :

Action :  $\text{Aller}(A, B)$

Préconditions :  $\hat{A}(\text{Robot}, A), \text{Route}(A, B)$

Ajouts :  $\hat{A}(\text{Robot}, B)$

Suppressions :  $\hat{A}(\text{Robot}, A)$

Action :  $\text{Aller}(B, C)$


Préconditions :  $\hat{A}(\text{Robot}, B), \text{Route}(B, C)$

Ajouts :  $\hat{A}(\text{Robot}, C)$

Suppressions :  $\hat{A}(\text{Robot}, B)$

- ♦ Plan possible :

$\text{Aller}(A, B) \rightarrow \text{Aller}(B, C)$

**Final** : Robot à C 

## 6/Facteurs déterminant le choix entre chaînage avant et arrière :

- **Chaînage avant (recherche dirigée par les données)** :

Raisonnement déductif, partir des faits initiaux et inférer de nouveaux faits pour atteindre un but.

La recherche dirigée par les données est utilisée quand : La majorité des données sont dans l'état initial. Il y a beaucoup de faits initiaux. Il y a peu de règles par rapport aux buts possibles. Le but est difficile à formuler ou inconnu.

- **Chaînage arrière (recherche dirigée par les buts)** :

Raisonnement inductif, partir du but à atteindre, le décomposer en sous-buts eux mêmes subdivisés à leur tour jusqu'à arriver à des faits initiaux.

La recherche dirigée par les buts est utilisée quand : Le but est facile à formuler. Beaucoup de règles sont applicables aux faits initiaux. Certaines données doivent être demandées à l'utilisateur.

## \*/Trace du raisonnement :

### a) Chaînage avant :

1. **Faits initiaux** : mange-viande, rapide, tacheté.
2. Applique la règle **R2** : "Si mange-viande, alors carnivore".
  - Conclusion : carnivore.
3. Applique la règle **R3** : "Si carnivore et rapide, alors félin".
  - Conclusion : félin.
4. Applique la règle **R4** : "Si félin et tacheté, alors guépard".
  - Conclusion : guépard.

Le raisonnement en chaînage avant suit cette logique, passant des faits initiaux vers la conclusion "guépard" à travers les règles.

### b) Chaînage arrière :

1. Le but à atteindre est **guépard**.
2. Applique la règle **R4** : "Si félin et tacheté, alors guépard".
  - Sous-but : félin et tacheté.
3. Pour obtenir **félin**, on applique la règle **R3** : "Si carnivore et rapide, alors félin".
  - Sous-but : carnivore et rapide.
4. Pour obtenir **carnivore**, on applique la règle **R2** : "Si mange-viande, alors carnivore".
  - Sous-but : mange-viande (déjà donné dans les faits initiaux).

Ainsi, on arrive aux faits initiaux ("mange-viande", "rapide", "tacheté") en décomposant le but "guépard".

### Résumé du raisonnement :

- **Chaînage avant** : On commence avec les faits initiaux et applique les règles pour arriver à la conclusion.
- **Chaînage arrière** : On part du but, on le décompose en sous-objectifs, et on cherche à atteindre ces objectifs avec les règles disponibles.

## II/Systèmes de raisonnement logique par résolution :

**Logique** : Définit un raisonnement correct, indépendamment du domaine.

**Raisonnement** : Utilise des propositions pour en déduire une conclusion.

**Proposition déclarative** : Énoncé pouvant être **vrai (1)** ou **faux (0)**.

### 1/Logique propositionnelle (logique d'ordre 0) :

- Ne contient **pas** de quantificateurs ("il existe", "pour tout").
- Utilise uniquement les **connecteurs logiques** :
  - $\neg$  : NON,  $\wedge$  : ET (Conjonction),  $\vee$  : OU (Disjonction),  $\rightarrow$  : IMPLIQUE,  $\leftrightarrow$  : ÉQUIVALENT

## Tables de vérité

La table de vérité est un tableau qui montre les résultats logiques d'une expression pour toutes les combinaisons possibles de valeurs.

a)  $A \rightarrow B$  (implication)

A	B	$A \rightarrow B$
1	1	1
1	0	0
0	1	1
0	0	1

📌 L'implication est **fausse uniquement** quand  $A = 1$  et  $B = 0$ .

b)  $A \leftrightarrow B$  (équivalence)

A	B	$A \leftrightarrow B$
1	1	1
1	0	0
0	1	0
0	0	1

📌 L'équivalence est **vraie quand A et B ont la même valeur**.

## Exemple de raisonnement

- Si **il fait beau** (beau = 1) **et ce n'est pas samedi** ( $\neg$ samedi = 1), alors **je fais du vélo** (vélo = 1).
- Si **je fais du vélo** (vélo = 1), alors **il y a du vent** (vent = 1).  
⇒ Donc, **si beau = 1 et samedi = 0, alors vent = 1**.

Formellement :

- $(\text{beau} \wedge \neg \text{samedi}) \rightarrow \text{vélo}$
- $\text{vélo} \rightarrow \text{vent}$
- $\Rightarrow (\text{beau} \wedge \neg \text{samedi}) \rightarrow \text{vent}$

## 2/Propriétés logiques et transformations :

### Propriétés des formules logiques

- **Formule valide** : toujours vraie, quelle que soit l'interprétation.

Exemple :  $A \vee \neg A$

Explication : Cette formule est toujours vraie, peu importe que  $A$  soit vrai ou faux.

- **Formule consistante** : vraie dans au moins une interprétation (sinon, inconsistante).

Exemple :  $A \wedge B$

Explication : Cette formule est consistante si  $A$  et  $B$  peuvent être vraies dans une certaine interprétation.

- **Modèle** : interprétation qui rend une formule vraie. Un ensemble est **satisfiable** s'il a au moins un modèle.

Exemple :  $A \rightarrow B$

Interprétation : Si  $A$  est vrai et  $B$  est vrai, alors cette formule est vraie dans ce modèle

- **Équivalence logique** : deux formules sont équivalentes si elles ont la même table de vérité (on note  $f \equiv f'$ ).

Exemple :  $A \rightarrow B \equiv \neg A \vee B$

Explication : Ces deux formules sont logiquement équivalentes, elles donneront toujours le même résultat de vérité.

- **Satisfaction** : un ensemble de formules  $\{f_1, \dots, f_n\}$  satisfait une formule  $f$  si tout modèle de  $\{f_1, \dots, f_n\}$  est aussi modèle de  $f$  (notation :  $\{f_1, \dots, f_n\} \models f$ ).

Exemple :

Ensemble de formules :  $A \rightarrow B$  et  $B \rightarrow C$

Formule :  $A \rightarrow C$

Explication : Si les deux premières formules sont vraies, alors  $A \rightarrow C$  sera également vraie dans cette interprétation.



## Règles de transformation utiles

- **Loi du tiers exclu** :  $A \vee \neg A$

Une proposition est soit vraie, soit fausse. Il n'y a pas de 3e possibilité.

*Exemple* : "Il pleut" ou "il ne pleut pas".

- **Modus ponens** :  $((A \rightarrow B) \wedge A) \rightarrow B$

Si A implique B et que A est vrai, alors B est vrai.

*Exemple* : Si je révise (A), alors je réussis (B).

J'ai révisé  $\rightarrow$  donc je réussis.

- **Modus tollens** :  $((A \rightarrow B) \wedge \neg B) \rightarrow \neg A$

Si A implique B et que B est faux, alors A est faux.

*Exemple* : Si je mange (A), alors j'ai de l'énergie (B). Je n'ai pas d'énergie  $\rightarrow$  donc je n'ai pas mangé.

- **Contraposition** :  $(A \rightarrow B) \equiv (\neg B \rightarrow \neg A)$

Une implication est équivalente à sa forme contraposée.

*Exemple* : Si je suis malade (A), alors je reste au lit (B). C'est pareil que : si je ne reste pas au lit  $\rightarrow$  je ne suis pas malade.

- **Double négation** :  $\neg \neg A \equiv A$

"Pas pas A", c'est la même chose que A.

*Exemple* : Il n'est pas faux que tu as raison  $\rightarrow$  Donc tu as raison.

- **Implication** :  $A \rightarrow B \equiv \neg A \vee B$

Une implication peut se réécrire avec "ou".

*Exemple* : Si tu cours (A), alors tu transpires (B)  
 $\leftrightarrow$  soit tu ne cours pas, soit tu transpires.

- **Équivalence** :  $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

A est équivalent à B si A implique B **et** B implique A.

*Exemple* : Tu es majeur (A) **si et seulement si** tu as 18 ans ou plus (B).

- **Idempotence** :  $A \vee A \equiv A, A \wedge A \equiv A$

Répéter une même proposition avec OU ou ET ne change rien.

*Exemple* : "Je suis content OU je suis content"  $\rightarrow$  juste "je suis content".

- **Lois de De Morgan** :

- $\neg(A \vee B) \equiv \neg A \wedge \neg B$

- $\neg(A \wedge B) \equiv \neg A \vee \neg B$

*Exemples* :

1. Ce n'est pas (je dors **ou** je mange)  
 $\rightarrow$  donc je **ne dors pas** ET je **ne mange pas**.

2. Ce n'est pas (je dors **et** je mange)  
 $\rightarrow$  donc je **ne dors pas** OU je **ne mange pas**.

- **Commutativité** :

$$A \vee B \equiv B \vee A, A \wedge B \equiv B \wedge A$$

On peut échanger l'ordre des propositions.

- **Associativité** :

$$(A \vee B) \vee C \equiv A \vee (B \vee C)$$

$$(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$$

L'ordre des regroupements ne change rien.

- **Distributivité** :

$$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$$

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$

Comme en mathématiques, on peut distribuer les opérateurs.

### 3/Logique des prédicats (logique du premier ordre) :

#### Remarque :

Lorsque l'exercice mentionne explicitement la *logique propositionnelle*, on travaille généralement avec des **clauses concrètes**, c'est-à-dire des formules en forme normale disjonctive (FND) ou conjonctive (FNC), utilisées notamment pour les arbres de résolution. En revanche, lorsqu'on est dans le cadre de la *logique des prédicats*, on manipule des **clauses générales** contenant des quantificateurs ( $\forall$ ,  $\exists$ ) et des variables, ce qui permet de représenter des faits plus abstraits et généraux.

#### ♦ Définition générale

- Extension de la logique propositionnelle.
- Elle introduit **des variables** et **des quantificateurs** ( $\forall$ ,  $\exists$ ).
- Ex :  $\forall x \text{ (homme}(x) \rightarrow \text{mortel}(x))$   
« Tous les x, si x est un homme alors x est mortel »

#### ♦ Éléments de syntaxe

- **Constantes** :  $V$ ,  $F$ ,  $\text{Malika}$ ,  $\text{Socrate}$ , etc.
- **Connecteurs logiques** :  $\wedge$ ,  $\vee$ ,  $\neg$ ,  $\rightarrow$ ,  $\leftrightarrow$
- **Variables** :  $x$ ,  $y$ ,  $z$ ...
- **Fonctions** :  $f(\dots)$ ,  $g(\dots)$ ,  $\text{mere}(\text{Malika})$ ...
- **Prédicats** :  $p(\dots)$ ,  $q(\dots)$ ,  $\text{homme}(x)$ ,  $\text{mortel}(x)$
- **Quantificateurs** :  $\forall$  (pour tout),  $\exists$  (il existe)

#### ♦ Définitions clés

- **Terme** :
  - Variable ( $x$ )
  - Constante ( $\text{Malika}$ )
  - Fonction sur des termes ( $\text{mere}(\text{Malika})$ )

- **Atome :**

- Prédicat appliqué à des termes :  $\text{humain}(\text{Socrate})$

- **Formule :**

- Atome ( $p(x)$ )
- Connecteurs ( $\neg F, F1 \wedge F2, F1 \rightarrow F2$ , etc.)
- Quantificateurs ( $\forall x F, \exists x F$ )

- ◆ **Exemples de formules valides**

- $\forall x \neg A \leftrightarrow \neg \exists x A$
- $\forall x A \leftrightarrow \neg \exists x \neg A$

- ◆ **Littéraux**

- **Positif** : un atome ( $\text{Rouge}(x)$ )
- **Négatif** : la négation d'un atome ( $\neg \text{Rouge}(x)$ )

- ◆ **Forme clausale**

- Forme normalisée des formules :  
 $\forall x_1 \dots \forall x_n (C1 \wedge C2 \wedge \dots \wedge Cn)$   
où chaque  $C_i$  est une **clause**, c'est-à-dire une **disjonction de littéraux**.

- ◆ **Types de clauses**

- **Clause** : disjonction de littéraux ( $\neg p \vee q$ )
- **Clause concrète** : sans variables
- **Clause de Horn** : au plus un littéral positif

- ◆ **Règle de résolution (inférence logique)**

- Ex :  
 $p \rightarrow q, q \rightarrow r \Rightarrow p \rightarrow r$   
(Transitivité)

- En **forme clausale** :

$$\{\neg p \vee q, \neg q \vee r\} \models \{\neg p \vee r\}$$

- Si deux clauses contiennent des littéraux **complémentaires** ( $\phi$  et  $\neg\phi$ ), on peut les **résoudre** et **déduire une nouvelle clause** sans ces littéraux.

## Principe de Résolution et de l'Unification :

La résolution est une méthode logique où deux clauses avec des littéraux complémentaires sont combinées pour créer une nouvelle clause. Si une clause vide apparaît, cela prouve l'inconsistance de l'ensemble. La réfutation consiste à prendre la négation de la formule à prouver et à démontrer son incohérence. L'unification permet de rendre deux termes identiques en remplaçant des variables. Un unificateur le plus général est la substitution la plus simple. La mise sous forme clausale transforme une formule quantifiée en une forme sans quantificateurs, composée de clauses disjonctives.

### mise sous forme clausale :

#### Cas 1 : Clauses concrètes

Les clauses concrètes sont celles où les variables sont déjà instanciées à des termes concrets (pas de quantificateurs).

#### 1. Éliminer les connecteurs $\rightarrow$ et $\leftrightarrow$ :

- $A \rightarrow B$  devient  $\neg A \vee B$
- $A \leftrightarrow B$  devient  $(\neg A \vee B) \wedge (\neg B \vee A)$

#### 2. Distribuer les négations ( $\neg$ ) :

- $\neg(\neg A)$  devient  $A$
- $\neg(A \wedge B)$  devient  $\neg A \vee \neg B$
- $\neg(A \vee B)$  devient  $\neg A \wedge \neg B$

#### 3. Mettre la formule sous forme conjonctive :

- $A \vee (B \wedge C)$  devient  $(A \vee B) \wedge (A \vee C)$
- $(A \wedge B) \vee C$  devient  $(A \vee C) \wedge (B \vee C)$

#### 4. Transformer chaque facteur en clause distincte :

- $(A \vee B) \wedge C$  devient  $\{A \vee B, C\}$

Chaque conjonction devient une disjonction de littéraux (clauses).

## Cas 2 : Clauses non concrètes (génériques)

Les clauses non concrètes contiennent des quantificateurs et des variables libres. On doit les transformer avant d'appliquer les mêmes règles.

### 1. Éliminer les connecteurs $\rightarrow$ et $\leftrightarrow$ :

- $A \rightarrow B$  devient  $\neg A \vee B$
- $A \leftrightarrow B$  devient  $(\neg A \vee B) \wedge (\neg B \vee A)$

### 2. Distribuer les négations ( $\neg$ ) :

- $\neg(\neg A)$  devient  $A$
- $\neg(A \wedge B)$  devient  $\neg A \vee \neg B$
- $\neg(A \vee B)$  devient  $\neg A \wedge \neg B$

### 3. Renommer les variables liées :

- Si une formule contient plusieurs quantificateurs avec les mêmes variables, on les renomme pour éviter les conflits. Exemple :

$$\blacksquare \quad \forall X p(X) \vee \exists X q(X) \text{ devient } \forall X p(X) \vee \exists Y q(Y)$$

### 4. Préfixer les quantificateurs :

- Les quantificateurs universels ( $\forall$ ) sont placés en tête de la formule, suivis de la matrice sans quantificateurs. Exemple :

$$\blacksquare \quad \forall X p(X) \vee \forall Y q(X,Y) \text{ devient } (\forall X \forall Y) p(X) \vee q(X,Y)$$

### 5. Éliminer les quantificateurs existentiels (Skolemisation) :

- Remplacer les variables existentielles par des fonctions ou constantes de Skolem. Exemple :

$$\blacksquare \quad \forall X \exists Y p(X,Y) \text{ devient } \forall X p(X, f(X)), \text{ où } f(X) \text{ est une fonction de Skolem.}$$

$$\blacksquare \quad \exists Y p(Y) \text{ devient } p(a), \text{ où } a \text{ est une constante de Skolem.}$$

### 6. Éliminer les quantificateurs universels :

- Les quantificateurs universels restants sont implicites une fois la formule transformée en forme normale, ils peuvent être ignorés.

## 7. Mettre la formule sous forme conjonctive :

- Distribuer les opérateurs pour obtenir une forme conjonctive.

## 8. Transformer chaque facteur en clause distincte :

- Chaque conjonction devient une disjonction de littéraux.

## 9. Renommer les variables d'une clause à l'autre :

- Pour éviter les conflits de variables, on renomme les variables dans chaque clause. Exemple :

$$\blacksquare \{ p(X) \vee q(X), r(Y) \}$$

### En résumé :

- **Cas concret** : Transformation directe en clauses via distribution, élimination des connecteurs, et conjonction.
- **Cas générique** : Ajout d'étapes de gestion des quantificateurs (renommage, skolemisation) avant de passer à la mise sous forme clausale.

## STRUCTURE GÉNÉRALE DE PROLOG

Élément	Syntaxe	Exemple
Fait	<code>predicate(terme1, terme2).</code>	<code>parent(ali, amine).</code>
Règle	<code>conclusion :- condition1, condition2.</code>	<code>pere(X, Y) :- homme(X), parent(X, Y).</code>
Variable	Majuscule	<code>X, Y</code>
Interdiction (contradiction)	<code>:- condition.</code>	<code>:- mere(X, said), age(X, Y).</code>

### Remarque générale :

En Prolog, **les clauses contenant des négations** (comme  $\neg P(x)$  ou **non**  $P(x)$ ) **se placent toujours dans la partie droite de la règle**, c'est-à-dire **après le  $:-$** .

Autrement dit, **la négation n'apparaît jamais dans la tête d'une clause**, mais uniquement dans le **corps**, pour exprimer une condition ou une contrainte logique.

MAGUEMOUN SAMY