Module Master M2

Raisonnement et Décisions
Chapitre VIII : Révision des cours de systèmes complexes et raisonnement et décisions

Présenté par : Prof. Kholladi Mohamed-Khireddine
Département d'Informatique
Facultés des Sciences Exactes
Université Echahid Hamma Lakhdar d'El Oued
Tél. 0770314924
Email. kholladi@univ-eloued.dz et kholladi@yahoo.fr
Site Web. www.univ-eloued.dz
http://kholladi.doomby.com/ et http://kholladi.e-monsite.com/

VIII – Cours de révision des cours de systèmes complexes et de raisonnement et décision

VIII.1 - La connaissance

Quelle est la différence entre les trois termes : donnée, information et connaissance?

1. La donnée transporte l'information : ce sont des signaux non interprétés ;
2. L'information est une interprétation de la donnée ;
3. La connaissance utilise l'information dans le cadre d'actions, dans un but précis. Les actions peuvent être la prise de décisions, la création de nouvelles informations, etc.

Questions fondamentales

1. Acquisition de la connaissance du domaine
2. Représentation de la connaissance (un formalisme syntaxique + sémantique)
3. Contrôle du raisonnement (complétude et validité)

La connaissance, c'est la capacité à mobiliser des informations pour agir. Le passage de l'information à la connaissance est lié à l'expérience de l'action, cela veut dire veut qu'il n'a pas de frontière parfaitement définie.

Définition

La connaissance, c'est l'information (donnée) qui influence un processus. La connaissance, c'est l'ensemble des notions et des principes qu'une personne acquiert par l'étude, l'observation ou l'expérience et qu'elle peut intégrer à des habiletés. Le synonyme de

génie cognitif : discipline étudiant l'extraction et la formalisation de connaissances provenant d'un expert humain en vue d'automatiser le raisonnement. En informatique, l'ingénierie des connaissances évoquerait les techniques pour manipuler des connaissances sur ordinateur.

VIII.2 - Formalismes de représentation de connaissances

Il existe plusieurs formalismes. Le choix d'un formalisme à utiliser dépend à la fois du domaine d'application, des opérations à mettre en œuvre sur ces connaissances (type de problème à résoudre) et surtout de la culture du concepteur du modèle de représentation comme sur la figure VIII.1.

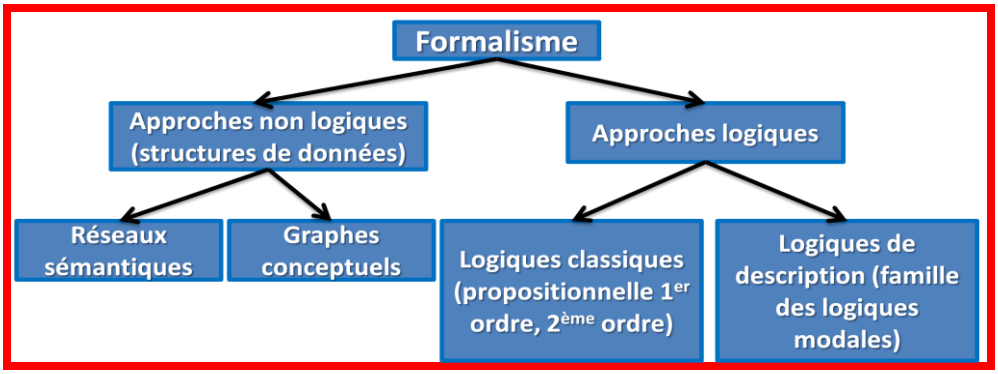

Figure VIII.1 - Formalismes de représentation de connaissances

VIII.3 - Typologie des problèmes

On distingue quatre type de typologie des problèmes :

1. Les problèmes de tournées étroitement apparentés aux problèmes de détermination de trajets. Ils présentent une différence notable. Cela consiste à visiter au moins toutes les villes en commençant et en terminant par la ville de départ.
2. Le problème du voyageur de commerce ou TSP (Traveling Salesperson Problem) est une variante du problème précédent dans lequel chaque ville doit être visitée exactement une fois. Le but de départ est de chercher l'itinéraire le plus court et au moindre coût.
3. Le problème d'agencement des circuits VLSI ou celui de placement-routage nécessite de positionner des millions de composants et de connexions sur une puce de façon à minimiser l'encombrement, les temps morts et les capacités parasites.
4. Le problème de déplacement d'un robot est une généralisation du problème de recherche d'itinéraire. Au lieu de suivre des chemins discrets, un robot peut se mouvoir dans un espace continu, caractérisé par un ensemble infini d'actions et d'états possibles.

VIII.4 - Résolution de Problème par l'exploration

VIII.4.1 - Résolution de problème

La figure VIII.2 illustre les deux démarches de résolution d'un problème. La démarche traite le problème pour le résoudre et la démarche de résolution traite le domaine pour le modéliser à des de représentation des connaissances.

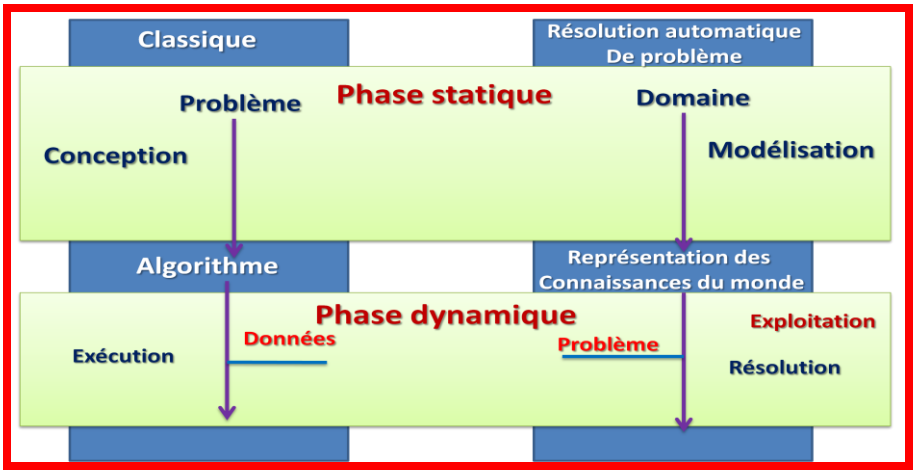

Figure VIII.2 – Résolution de problème

Dans cette partie du cours, on va voir deux types d'approche de résolution pour deux classes de problèmes différents. La première classe de problèmes sont les systèmes experts (type de problème : classification, diagnostic, réponse à des questions. Exemple : est ce que la "baleine bleue" est un mammifère ?) et le deuxième classe de problèmes : la planification (définir la suite d'actions pour atteindre un état donné, exemple : comment aller à la fac ?).

VIII.4.2 - Introduction

La résolution d'un problème revient souvent à chercher un état but dans l'espace des états. Un agent (un objet ou système) orienté-but qui utilise un mécanisme d'exploration est dit : agent (objet ou système) solveur de problème. La recherche de solution nécessite :

1. La formulation du problème :
    a. Espace des états ou ensemble des états,
    b. État initial,
    c. État but,
    d. Actions,
    e. Modèle de transition,
    f. Coût de chemin.

2. La stratégie d'exploration : comment parcourir l'arbre d'exploration pour arriver à une solution?

VIII.4.3 - Spécification de l'environnement d'une tâche

Les agents (objets ou systèmes) interagissent avec leurs environnements au moyen de capteurs et d'effecteurs (actionneurs) comme sur la figure VIII.3.

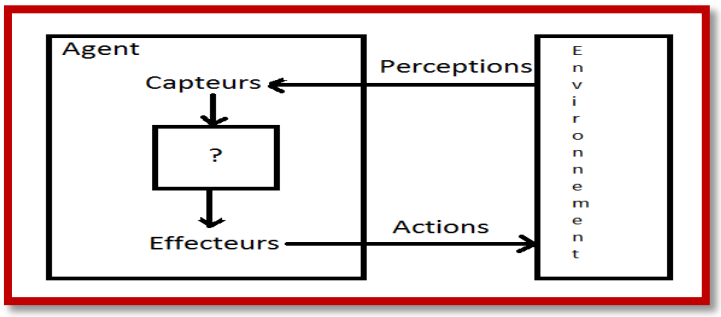

Figure VIII.3 – Agent et son environnement

Le tableau suivant, nous donne des exemples d'environnement de tâche et leurs caractéristiques.

Environnement de la tâche	Observable	Agents	Déterministe	Episodique	Statique	Discret
Problème des mots croisés	Entièrement	Mono	Déterministe	Séquentiel	Statique	Discret
Partie d'échecs chronométrée	Entièrement	Multi	Déterministe	Séquentiel	Semi	Discret
Pocker	Partiellement	Multi	Stochastique	Séquentiel	Statique	Discret
Backgammon	Entièrement	Multi	Stochastique	Séquentiel	Statique	Discret
Conduite de taxi	Partiellement	Multi	Stochastique	Séquentiel	Dynamique	Continu
Diagnostic médical	Partiellement	Mono	Stochastique	Séquentiel	Dynamique	Continu
Analyse d'image	Entièrement	Mono	Déterministe	Episodique	Semi	Continu
Robot détecteur de défauts	Partiellement	Mono	Stochastique	Episodique	Dynamique	Continu
Contrôleur de raffinerie	Partiellement	Mono	Stochastique	Séquentiel	Dynamique	Continu
Répétiteur s'anglais interactif	Partiellement	Multi	Stochastique	Séquentiel	Dynamique	Discret

VIII.4.4 - Formalisation d'un problème

a - Qu'est-ce que l'exploitation de l'exploration?

C'est le processus de construction des séquences d'actions par un agent (un objet ou un système) pour atteindre ses buts. Pour qu'un agent puisse commencer à explorer pour trouver des solutions, il faut d'abord identifier le but et formuler précisément le problème. Un

problème se compose de six parties : l'espace des états, l'état initial, un ensemble d'actions, un modèle de transition décrivant les résultats de ces actions, une fonction de test de but et une fonction de coût de chemin. L'environnement du problème est représenté par un espace des états. Un chemin dans l'espace des états allant de l'état initial à un état but est une solution. Les algorithmes d'exploration traitant des états et les actions comme étant atomiques : ils ne tiennent pas compte de leur éventuelle structure interne. Un problème peut être décrit ou articulé selon six composantes comme sur le tableau suivant :

Composantes du problème	**Significations et explications**
Espace d'états	L'ensemble des états du problème
Etat initial	Point de départ de l'exploration
Actions	Toutes les actions possibles qui permettent le passage entre états. Action(s) : actions disponibles à l'état s.
Modèle de transition	Il décrit l'effet des actions sur les états, par la fonction Résultat. Résultat(s, a) = s'. s': état résultant de l'exécution de l'action a à l'état s.
Test du but	Condition ou état de succès (arrêt).
Coût de chemin	Une fonction qui attribue des coûts aux chemins de recherche. c(s, a, s') : coût d'exécution de l'action a à l'état s (une étape). Coût chemin = somme (coûts étapes)

b - Agent (objet ou système) de résolution de problèmes

Pour qu'un agent (objet ou système) procède à la résolution des problèmes, il passe par les quatre étapes suivantes :

1. Formulation d'un but : un état à atteindre ;
2. Formulation du problème : les états et les actions à considérer ;
3. Exploration de solution : examiner les différentes séquences d'actions menant à un état but et choisir la meilleure ;
4. Exécution : accomplir la séquence d'actions sélectionnées.

c - Exemple de problème

Un voyageur qui est dans la ville Arad cherche un chemin vers Bucarest (Roumanie). Quel est le plus court chemin entre deux villes données (voir Arad et Bucarest) ? (voir la figure VIII.4 représentant les villes de la Roumanie)

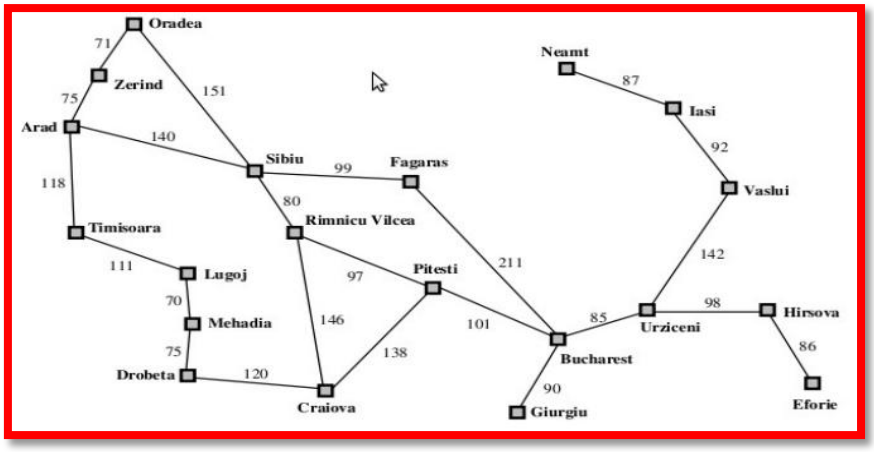

Figure VIII.4 – Carte des villes de la Roumanie

d - Exemple de formulation du problème

Le tableau suivant illustre la formulation de notre problème de déplacement entre les villes de la Roumanie surtout de trouver le chemin pour aller d'une ville à une autre.

Composantes du problème	**Significations et explications**
Espace des états	Ensemble des n villes de la Roumanie
Etat initial	État initial : Dans(Arad)
Actions	Actions: Aller(Zerind), Aller(Sibiu), etc.
Modèle de transition	Result(Dans(Arad),Aller(Zerind)) = Dans(Zerind), et.
Test du but	Dans(Bucarest)? ou {Dans(Bucarest)}
Coût de chemin	Coût(Dans(Arad), Aller(Zerind), Dans(Zerind)) = 75 Path_cost(<Arad, Sibiu,Fagaras,Bucarest>) = 450 meilleur chemin?

VIII.5 - Résolution de Problème par les stratégies d'exploration

VIII.5.1 - Stratégie d'exploration

Comment explorer l'espace des états d'un problème pour trouver une solution (chemin vers un but)? Une stratégie d'exploration est systématique, et devra permettre d'atteindre un état but à partir de l'état initial. Une solution est le chemin entre ces deux états, son coût est la longueur de ce chemin. Une solution est optimale si elle a le plus petit coût. Une stratégie est optimale si elle présente le meilleur temps d'exploration. Elle est complète, si elle garantie de trouver une solution lorsque celle-ci existe. Un algorithme général d'exploration dans un arbre tel "Exploration-Arbre" considère tous les chemins possibles pour trouver une solution, tandis qu'un algorithme d'exploration dans un graphe tel que "Exploration-Graphe" ignore les chemins redondants. Les algorithmes d'exploration sont évalués selon leur complétude, leur optimalité, leur complexité en temps et leur complexité en espace. La complexité dépend de b, le facteur de branchement dans l'espace des états et de d, la profondeur de la solution la moins profonde. Il existe deux catégories :

1. Exploration non-informée ou aveugle (Blind search) : la définition du problème est la seule connaissance. Exemples : en profondeur d'abord (breadth-first search), en largeur d'abord (depth-first search), etc.
2. Exploration informée (Heuristique) : une exploration guidée; connaissance de la désirabilité (l'attractivité) d'une action. Exemple : recherche du meilleur d'abord (best-first search).

Les stratégies d'exploration (y compris la non-informée) divergent dans l'ordre de développement des nœuds à explorer. Ils déterminent l'ordre de développement des nœuds.

- Explorations non informées :
  - Aucune information additionnelle.
  - Elles ne peuvent pas dire si un nœud est meilleur qu'un autre.
  - Elles peuvent seulement dire si l'état est un but ou non.
- Explorations informées (heuristiques) :
  - Elles peuvent estimer si un nœud est plus prometteur qu'un autre.

VIII.5.2 - Évaluation des stratégies d'exploration

Il existe quatre critères d'évaluation des stratégies d'exploration, qui sont :

1. Complétude : Est-ce que l'algorithme garantit de trouver une solution s'il y en a une?

2. Optimalité : Est-ce que la stratégie trouve la solution optimale?
3. Complexité en temps : Combien de temps pour trouver une solution?
4. Complexité en espace : Quelle quantité de mémoire a-t-on besoin?

a – Complexité

Elle est exprimée en utilisant les quantités suivantes :

- B (ou b) : le facteur de branchement
  - C'est-à-dire le nombre maximum de successeurs à un nœud.
- D (ou d) : la profondeur du nœud but le moins éloigné.
- M (ou m) : la longueur maximale d'un chemin dans l'espace d'états.

La complexité en temps est le nombre de nœuds générés pendant la recherche et la complexité en espace et le nombre maximum de nœuds en mémoire.

VIII.5.3 - Les méthodes d'exploration non informée

Les méthodes d'exploration non informée ont accès seulement à la définition de problème : les algorithmes de base sont les suivants sur le tableau :

Algorithmes d'exploration	Développe	Caractéristiques
Exploration en largeur	Les nœuds les moins profonds d'abord	Elle est complète et optimale pour des coûts d'étape unitaires, mais de complexité en espace exponentielle.
Exploration à coût uniforme	Les nœuds ayant le plus bas coût de chemin g(n)	Elle est optimale pour des coûts d'étape quelconques.
Exploration en profondeur d'abord	Le nœud non développé le plus profond d'abord	Elle n'est ni complète, ni optimale, mais possède une complexité en espace linéaire.
Exploration limitée en profondeur	Le nœud non développé le plus profond jusqu'à la limite	Elle ajoute une limite de profondeur pour optimiser l'exploration en profondeur.
Exploration en profondeur itérative	Elle appelle l'exploration en profondeur d'abord et augmente les limites de profondeur jusqu'à ce qu'un but soit trouvé.	Elle est complète, optimale pour des coûts d'étape unitaires, possède une complexité en temps comparable à celle de l'exploration en largeur et possède une complexité en espace linéaire.
Exploration Bidirectionnelle	Elle consiste à exécuter deux explorations simultanées : l'une vers l'aval depuis l'état initial et l'autre vers l'amont à partir du but, en espérant qu'elles se rencontrent au milieu.	Elle peut énormément réduire la complexité en temps mais elle n'est pas toujours applicable et peut exiger trop d'espace.

VIII.5.4 - Les méthodes d'exploration informée

Les méthodes d'exploration informée peuvent avoir accès à une fonction heuristique h(n) qui estime le coût d'une solution à partir de n. Les performances de ces méthodes dépendent de la qualité de la fonction heuristique. Les algorithmes de base sont les suivants sur le tableau :

Algorithmes d'exploration	Développe	Caractéristiques
Algorithme générique d'exploration par le meilleur d'abord	Choisir un nœud à développer selon une fonction d'évaluation.	
Exploration gloutonne par le meilleur d'abord	Les nœuds ayant le h(n) minimal.	Elle n'est pas optimal mais souvent efficace.
L'exploration A*	Les nœuds ayant la valeur f(n)=g(n)+h(n) minimal.	A* est complet et optimal, à condition que h soit admissible (pour "Exploration-Arbre") ou consistance (pour "Exploration-Graphe"). La complexité en espace de A* est encore prohibitive.
RBFS (exploration récursive pour le meilleur d'abord)	Il utilise des quantités limites de mémoire s'il dispose assez de temps. Il peut résoudre les problèmes que A* ne peut pas résoudre parce qu'il manque de mémoire.	Algorithme d'exploration robuste et optimal
SMA* (Exploration A* sous contrainte simplifié)	Il utilise des quantités limites de mémoire s'il dispose assez de temps. Il peut résoudre les problèmes que A* ne peut pas résoudre parce qu'il manque de mémoire.	Algorithme d'exploration robuste et optimal

VIII.5.5 - Stratégie d'exploration dans l'arbre d'un problème

La formulation de problème est l'espace ou l'ensemble des états, état initial, état but, fonction de transition (entre états), fonction de coût de chemin. La solution est le chemin de moindre coût entre l´état initial et l'état but. Le principe d'exploration ou de recherche, c'est de développer le meilleur nœud d'abord, ce qui nécessite d'information supplémentaire sur le problème (dite, heuristique), pour guider l'exploration ou la recherche.

Avantage :

1. Efficace en termes de calcul ; parcours partiel (minimal) de l'espace d'exploration.
2. Garantie de l'optimalité de la solution, sous certaines conditions sur l'heuristique utilisée.

Inconvénient : complexité spatiale prohibitive.

Des variantes de A∗ contournent ce problème. La fonction d'évaluation f donne la désirabilité d'un nœud.

f (e) ↓ ⇒ désirabilité (ou l'attractivité) du nœud e ↓

f (e) = g(e) + h(e), ∀e ∈ espace des états.

g(e) : donne le coût de chemin allant de l'état initial e_0 jusqu'au nœud e.

h(e) : estime le plus petit coût entre l'état courant e et l'état but e_b.

a - Problème du chemin le plus court avec A*

Problème : le plus court chemin de Arad vers Bucarest ?

Formulation : espace des états = villes, e_0 = Arad, e_b = Bucarest.

Fonction de transition = voisinage des villes, fonction de coût = distance entre villes voisines.

Fonction heuristique : h(e) : distance à vol d'oiseau entre les villes e et e_b.

Distance réelle entre e et e_b ne peut être meilleure que h(e).

Fonction d'évaluation :

f (e) = g(e) + h(e), ∀e ∈ E (E étant l'ensemble des états).

g(e) : distance réellement parcourue depuis e_0 pour arriver à e.

h(e) : estimation optimiste de coût entre e et e_b.

Le tableau suivant donne la distance entre les villes.

Villes	Distance	Ville	Distance
Arad	366	Mehadia	241
Bucarest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vasli	199
Lugoj	244	Zerind	374

b - Illustration de A*

La figure VIII.5 illustre les états de transitions entre un nœud et un autre en indiquant la manière de calculer les valeurs. Il s'agit de recherche du chemin le plus court de la ville Arad à Bucarest ? Les nœuds sont étiquetés avec **f = g + h**.

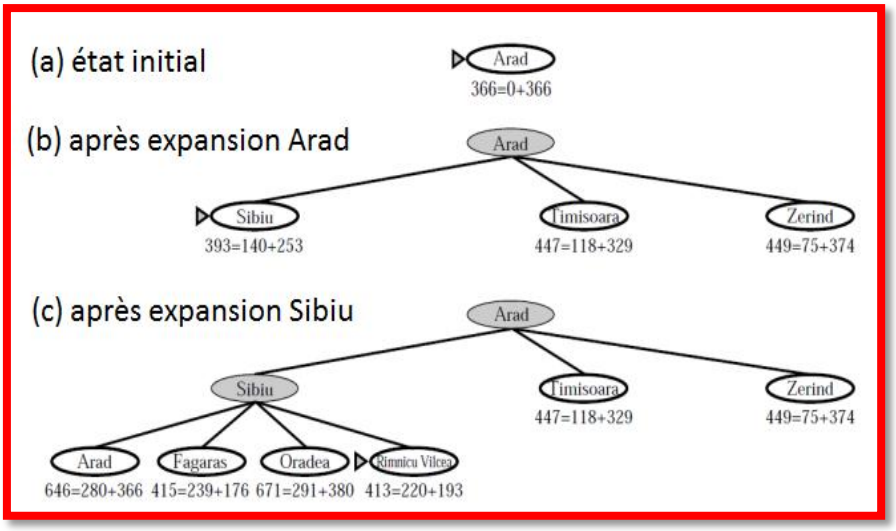

Figure VIII.5 – Les états de transitions

A chaque fois, on cherche le meilleur nœud, s'il n'est pas un état but, on le développe, c'est-à-dire, on met ses successeurs dans la liste des nœuds à visiter. Les nœuds en gris sont les nœuds visités, ceux en blanc sont à visiter et le petit triangle gris pointe vers le meilleur nœud courant. La figure VIII.6 indique les transitions jusqu'à Rimnicu Vilcea.

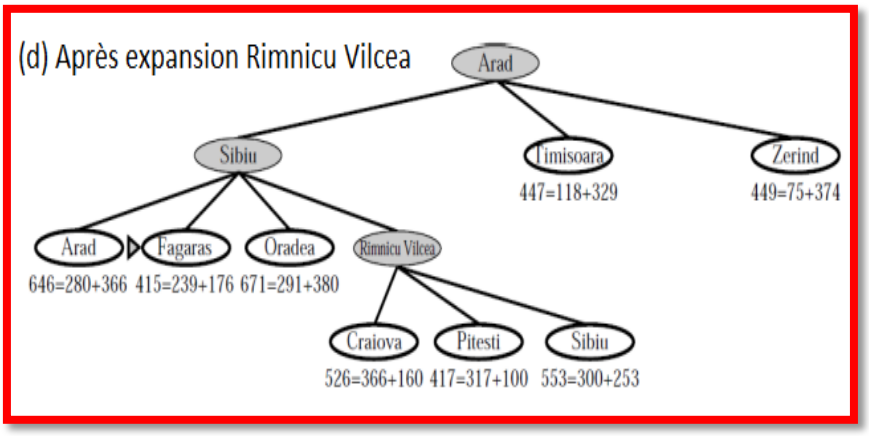

Figure VIII.6 – Transitions jusqu'à Rimnicu Vilcea

Le nœud Sibiu est doublant, on garde le premier vue qu'il a une meilleure évaluation. Le nœud but Bucarest est rencontré, néanmoins on continue la recherche au cas où un chemin de coût meilleur existe comme sur la figure VIII.7.

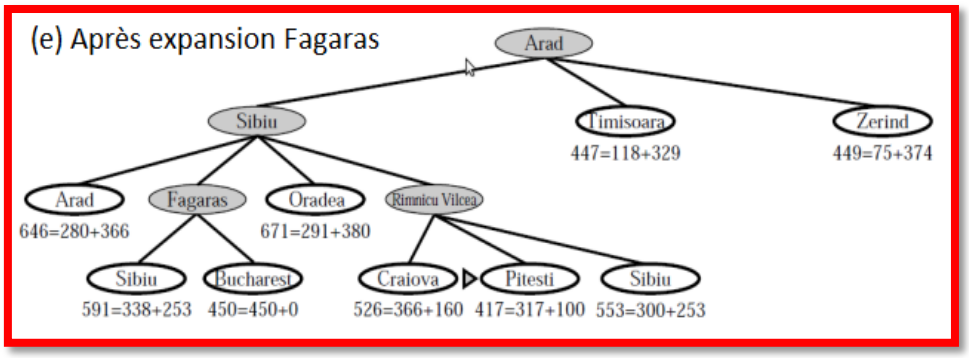

Figure VIII.7 – Transitions jusqu'à Pitesti

L'arrêt de l'exploration, vu qu'on est sure de ne pas pouvoir trouver mieux. Le coût de la recherche est f (e_b) = 418 = longueur du chemin le plus court. La séquence des nœuds parcourus de l'état initial jusqu'au but est la solution optimale désirée : Arad → Sibiu → RimnicuVilcea → Pitesti → Bucarest comme sur la figure VIII.8.

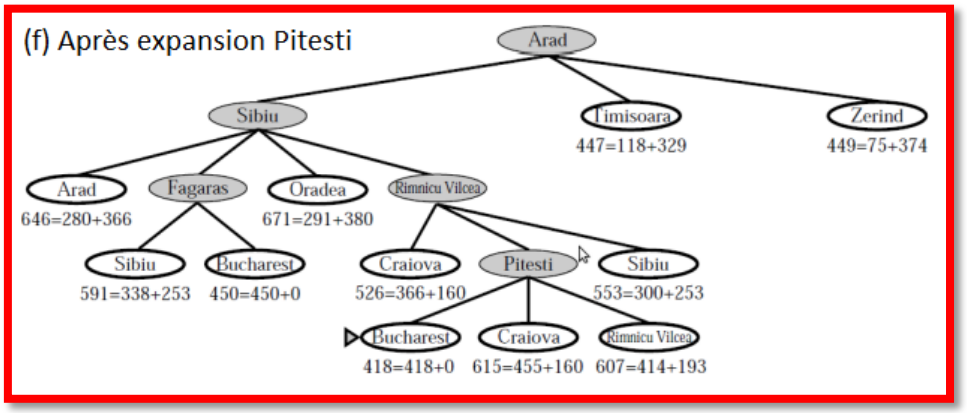

Figure VIII.8 – Transitions jusqu'à Bucharest

VIII.6 - Domaine d'application

Les systèmes experts ont été conçus pour résoudre certains types de problèmes comme en médecine, en droit, en chimie, en éducation, etc. Les catégories de problèmes abordés par les systèmes experts sont :

- L'interprétation ou la construction d'une description abstraite à partir de données.
- La prédiction des conséquences à partir de situations données.
- Le diagnostic d'une défaillance à partir d'un ensemble d'observations.
- La conception d'une configuration de composants à partir d'un ensemble de contraintes.
- La planification d'une séquence d'actions pour l'accomplissement d'un ensemble de buts à partir de certaines conditions de départ et en présence de certaines contraintes.
- La réparation d'un dysfonctionnement.

- Le contrôle du comportement d'un environnement complexe.

11.7 - Structure et fonctionnement d'un système expert

L'architecture d'un système expert typique est constituée de plusieurs modules qui s'interagissent :

- L'interface utilisateur sert à simplifier la communication, elle peut utiliser la forme question-réponse, le menu, le langage naturel etc.
- La base de connaissances contient les connaissances concernant la résolution du problème.
- Le moteur d'inférence applique une stratégie de résolution en utilisant les connaissances et ceci pour en dériver une nouvelle information.
- La base de faits contient les données spécifiques liées à l'application traitée. Elle peut contenir aussi les solutions intermédiaires ou les conclusions partielles trouvées lors de l'inférence.
- Le module d'explication permet au système expert d'expliquer son raisonnement.
- L'éditeur permet l'édition des connaissances dans la base.

VIII.7.1 - Structure d'un système expert

La figure VIII.9 illustre la structure d'un système expert.

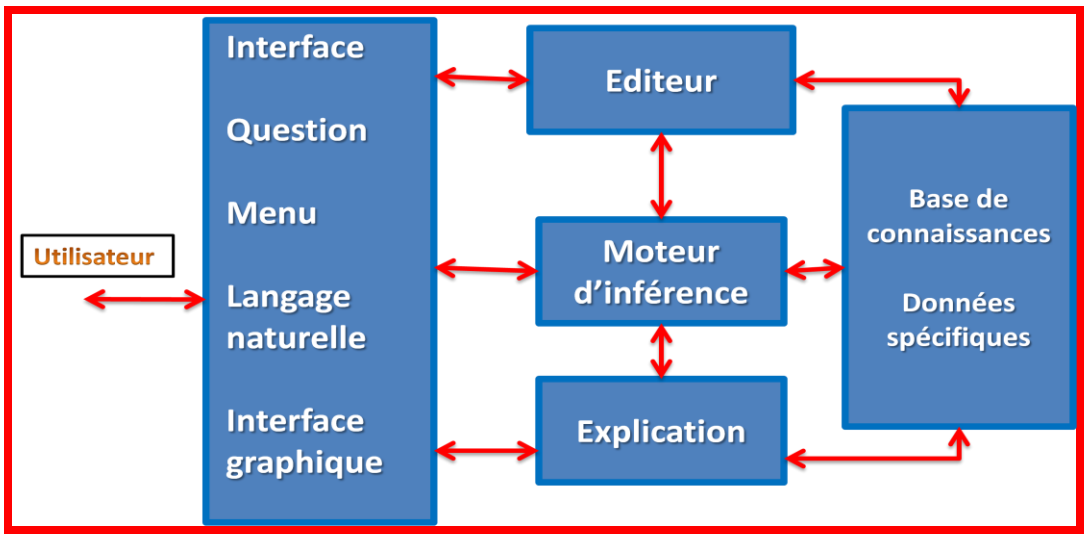

Figure VIII.9 – Structure d'un système expert

La figure VIII.10 illustre les interactions de l'expert et de l'utilisateur avec le système expert.

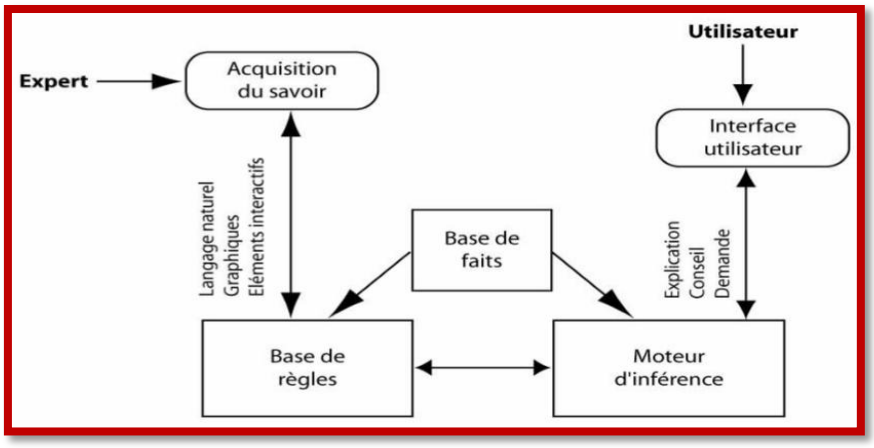

Figure VIII.10 – Interactions d'expert et d'utilisateur avec le système expert

VIII.7.2 – Architecture

La figure VIII.11 illustre l'architecture d'un système expert.

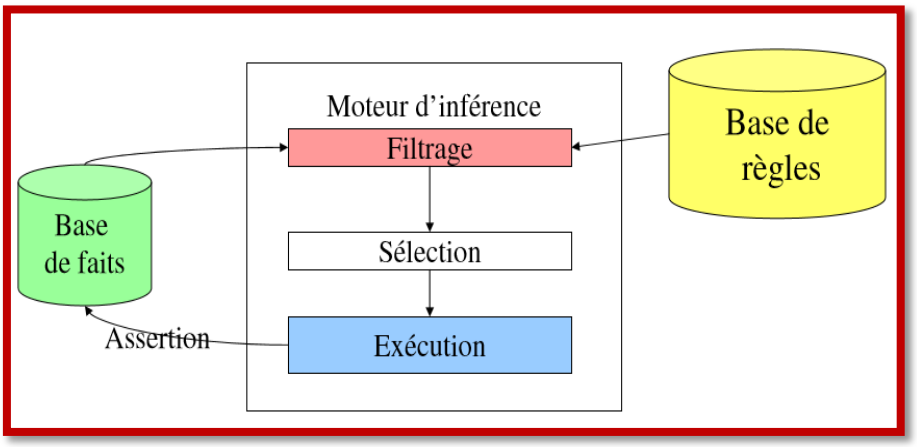

Figure VIII.11 – Architecture d'un système expert

VIII.7.3 - Base de faits

- Variable :
  - Contient les faits exprimés par l'expert.
  - S'enrichit des nouveaux faits déduits.
- Type de faits :
  - Booléens : vrai, faux
- Exemple : activé
  - Symboliques : appartenant à un domaine fini de symboles
- Exemple : auteur
  - Réels : représentation des faits continus

- Exemple : température

VIII.7.4 – Formules

Les types de systèmes experts sont :

- Ordre 0 : utilisation de faits/formules booléens uniquement : activité, ¬activité
- Ordre 0+ : faits pouvant être représenté sous la forme : activité $\wedge$ (température < 100 v liquide $\neq$ eau)
- Ordre 1 : faits pouvant utiliser la logique du premier ordre : $\exists$x liquide(x) $\wedge$ x $\neq$ eau $\wedge$ température(x) < 100.

La condition est une formule déclenchant une règle.

VIII.7.5 - Base de règles

Elle définit la connaissance sous forme de savoir-faire. Elle n'évolue pas au cours des sessions.

Règle = SI <X> ALORS <Y> :

X = prémisse = conjonction de conditions

Condition = 'température > 100' $\wedge$ 'liquide = eau'

Y = conclusion qui peut être un fait ou une valeur

Exemple : 'interrupteur=off'

On a la possibilité de coder par exemple : Si A ou B alors C (écriture sur deux lignes) ou Si A lors B et C (écriture sur deux lignes)

On a l'impossibilité de coder par exemple : Si A alors B ou C.

VIII.7.6 - Moteurs d'inférences

Le but est d'inférer de nouvelles connaissances à partir de la base de connaissance. Le moteur d'inférences a trois temps suivants : la sélection d'un ensemble de règles pouvant s'appliquer à la base de faits actuelle, La résolution de conflit par le choix de la meilleure règle à appliquer et le déclenchement de l'exécution de la règle et des opérations de mise à jour. Il a aussi trois méthodes : le chaînage avant pour déduire un fait. Le chaînage arrière pour donner la liste de faits (et des règles) menant à un fait particulier et le chaînage mixte pour combine les chaînages avant et arrière.

VIII.8 - Mode de raisonnements

Il existe de nombreux types de moteurs, capables de traiter différentes formes de règles logiques pour déduire de nouveaux faits à partir de la base de connaissance. On distingue souvent trois catégories, basées sur la manière dont les problèmes sont résolus :

1. Les moteurs dit à chaînage avant : qui partent des faits et règles de la base de connaissance, et tentent de s'approcher des faits recherchés par le problème.
2. Les moteurs dits à chaînage arrière : qui partent des faits recherchés par le problème, et tentent par l'intermédiaire des règles, de remonter à des faits connus,
3. Et les moteurs dits à chaînage mixte : qui utilisent une combinaison de ces deux approches chaînage avant et chaînage arrière.

Le concept de connaissance est donc lié à l'interprétation des données et exige de ce fait un interprète, qui devra générer de nouveaux faits à partir de données (inférence). On distingue trois techniques d'inférences :

1. Déduction : $(p \wedge (p \rightarrow q)) \Rightarrow (q)$
2. Abduction : $(q \wedge (p \rightarrow q)) \Rightarrow$ peut-être(p)
3. Induction : $p(A) \rightarrow q$, $p(B) \rightarrow q$, $\Rightarrow \forall x\ (p(x) \rightarrow q)$

VIII.8.1 - Chaînage avant

Le raisonnement guidé par les données est pour détecter les règles dont les prémisses sont vérifiées (filtrage), sélectionner la règle à appliquer, appliquer la règle et recommencer jusqu'à ce qu'il n'y ait plus de règle applicable.

Exercice

Soit la base de règles suivantes :

1. R1 : $(\neg A \rightarrow B) \rightarrow P$
2. R2 : (P et Q) $\rightarrow$ F
3. R3 : $(C \rightarrow A) \rightarrow Q$
4. R4 : $F \rightarrow (D \rightarrow K)$
5. R5 : $K \rightarrow$ (M et L)

La base initiale de faits est : (A, D). Prouvez le fait M par chaînage avant.

VIII.8.2 - Chaînage arrière

Le but initial placé au sommet d'une pile va permettre de détection des règles qui concluent à ce but, la résolution de conflits va permettre d'application de la règles, voir, les éléments des prémisses deviennent de nouveau sous-buts à atteindre. Le critère d'arrêt, c'est lorsque la pile est vide ou aucune règle applicable.

a - Arbre et/ou

La figure VIII.12 illustre un exemple d'arbre et/ou.

Soit BF = {B, C},

Fait = H et BR composée des règles :

1. B ∧ D et E → F
2. G ∧ D → A
3. C ∧ F → A
4. B → X
5. D → E
6. X ∧ A → H
7. C → D
8. X ∧ C → A
9. X ∧ B → D.

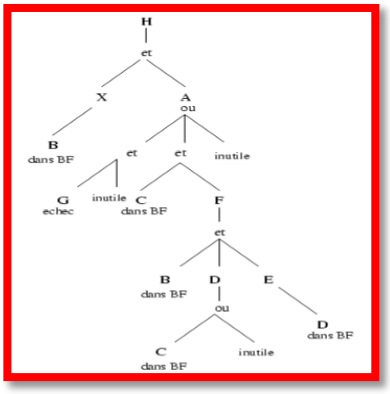

Figure VIII.12 – Illustration d'un exemple d'arbre et/ou

b - Chaînage arrière avec fait demandable

La figure VIII.13 illustre un exemple de chaînage arrière avec demandable.

1. B ∧ C → A
2. D ∧ E → A
3. F ∧ G → A
4. I ∧ J → G
5. J → ¬ E

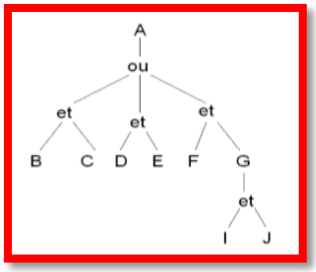

Figure VIII.13 – Illustration d'un exemple de chaînage arrière avec fait demandable

On suppose que les faits B, D, F et I sont les seuls faits demandables. La mémoire de travail est initialisée avec l'information J est vrai. La question posée au système est : A est-il vrai ? Quelles sont les questions pertinentes à poser à l'utilisateur ?

- B est-il vrai n'est pas une question pertinente. Aucune règle ne conclut sur le fait C qui n'est pas non plus demandable. Le fait B ne peut être utilisé que conjointement à C, la valeur de vérité de B n'apportera aucune information sur celle de A.
- D est-il vrai n'est pas non plus pertinente. En effet, comme on sait que J est vrai, que cela implique que E est faux et que D n'est utilisé que conjointement à E.
- F est-il vrai est pertinente. En effet, le fait G est encore déductible. Mais si la réponse à cette question est NON,
- I est-il vrai n'est plus pertinente car la valeur de G ne sert plus à rien.

Exercice 1

Un expert a construit la base de règles suivantes :

1. R1 : A et B $\rightarrow$C
2. R2 : D $\rightarrow$A
3. R3 : E $\rightarrow$ F
4. R4 : G $\rightarrow$ H
5. R5 : I $\rightarrow$ F
6. R6 : H et F et J $\rightarrow$ B
7. R7 : H et K $\rightarrow$J
8. R8 : G et F $\rightarrow$ K

La base initiale de faits est : (D, G, I). Prouvez le fait C par chaînage avant et chaînage arrière.

Exercice 2

Soit la base de règles suivantes :

1. R1 : (A et E) $\rightarrow$ G
2. R2 : (E $\rightarrow$ K) $\rightarrow$ D
3. R3 : (A et B) $\rightarrow$ (C et D)
4. R4 : (B $\rightarrow$A) $\rightarrow$ ((K et ¬F) ou G)
5. R5 : (A et G) $\rightarrow$ B
6. R6 : (K $\rightarrow$ E) $\rightarrow$C
7. R7 : (F et E) $\rightarrow$ B

La base initiale de faits est : (A, F). Prouvez le fait C par chaînage avant.

Exercice 3

Chaînage mixte

1. Ali, Fatima, Omar ne seront pas tous présent
2. Si Farid vient, Omar vient aussi
3. Si Fatima vient, la réunion commencera en retard
4. Si la réunion commence en retard, Omar n'y sera pas
5. Si Ali et Farid assistent à la réunion, elle ne sera pas calme

6. Si Ali, Fatima et Salima assiste à la réunion, et si Omar ne vient pas, la réunion sera fructueuse

Que peut-on déduire du fait : la réunion ne sera pas calme ? Que peut-on déduire du fait : la réunion sera fructueuse ?

Exercice 4 : prédicats (formalisation)

1. Ali est chez-lui ou chez Fatima.
2. Si Ali n'est pas chez-lui, il est chez Fatima.
3. Vous pouvez déduire vos frais médicaux si votre revenu est inférieur à 25 000 DA et que vous avez plus de 65 ans.
4. Vous ne pouvez pas déduire vos frais médicaux si vous n'avez pas plus de 65 ans ou que votre revenu est inférieur à 25 000 DA.
5. Ali réussira son examen ou il n'est pas fort en logique.
6. Si Ali n'est pas fort en logique, Dalila n'est pas forte non plus en logique et elle ne réussira pas son examen.
7. Ali et Dalila réussiront leur examen s'ils sont forts en logique.
8. Neige en hivers, Aid El Fitre après le Ramadan.

VIII.9 - Problème de l'agent (ou objet) aspirateur

Soit un monde de l'agent (ou objet) aspirateur limité à deux emplacements comme sur la figure VIII.14.

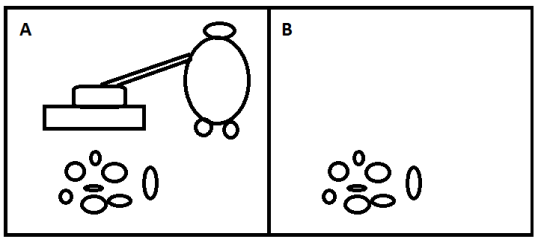

Figure VIII.14 – Robot aspirateur

Le monde de l'aspirateur est tellement simple qu'on peut décrire tout ce qui s'y passe : comme il est également créé de toutes pièces. On peut en inventer de nombreuses variantes. Ce monde particulier ne contient que deux lieux : A et B. L'agent (ou objet) aspirateur perçoit la case dans laquelle il se trouve et si celle-ci contient ou non de la poussière. Il peut choisir de se déplacer vers la gauche, ou vers la droite, d'aspirer la poussière ou de ne rien faire. Voici une fonction agent très simple : si la case courante est sale, alors aspirer, sinon se déplacer vers l'autre case. La tabulation partielle d'une fonction agent simple pour le monde de l'agent (ou objet) aspirateur comme sur le tableau suivant :

Les séquences de perceptions	Actions
[A, Propre]	Droite
[A, Sale]	Aspirer
[B, Propre]	Gauche
[B, Sale]	Aspirer
[A, Propre], [A, Propre]	Droite
[A, Propre], [A, Sale]	Aspirer
…	…
[A, Propre], [A, Propre], [A, Propre]	Droite
[A, Propre], [A, Propre], [A, Sale]	Aspirer
…	…

Etablir le tableau de résolution du problème comme suit : espace des états, état initial, actions, modèles de transition, tests de but, coût du chemin).

Composantes du problème	Significations ou Explications
Espace des états	L'état est déterminé par l'emplacement de l'agent et les emplacements de la poussière. L'agent (ou objet) est dans un des deux emplacements, qui peuvent contenir de la saleté ou non. Ainsi, il y a $2*2^2=8$ états possibles du monde. Un plus grand environnement doté de n emplacements on aurait $n*2^n$ états.
Etat initial	N'importe quel état peut être pris comme état initial.
Actions	Dans cet environnement élémentaire, chaque état a juste trois actions possibles : Gauche, Droite et Aspirer. Un plus grand environnement pourrait aussi disposer des actions Haut et Bas.
Modèle de transition	Les actions ont leurs effets prévus, sauf que les actions Gauche dans le carré le plus gauche, Droit dans le carré le plus à l'extrême droite, et Aspirer dans un endroit propre n'ont aucun effet. On peut l'illustrer dans l'espace des états complets.
Tests de but	Il vérifie si toutes cases sont propres.
Coût de chemin	Chaque étape coûte 1 et le coût de chemin est donc le nombre d'étapes dans le chemin.

L'espace des états pour le monde de l'aspirateur les liens dénotent des actions : G = Gauche, D = Droite, A = Aspirer selon la figure VIII.15.

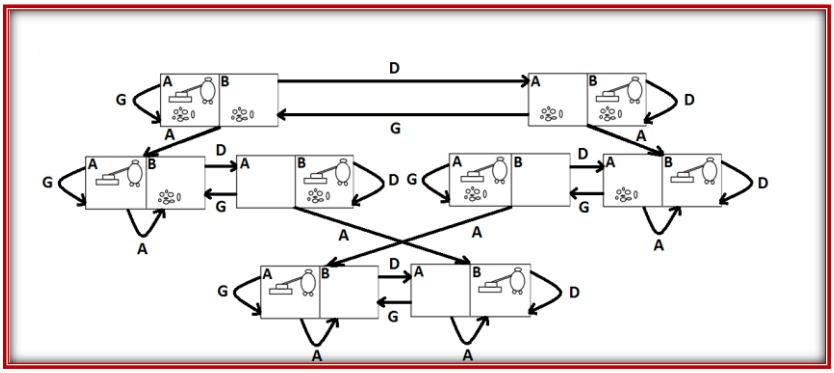

Figure VIII.15 – Espace des états pour le monde de l'aspirateur

VIII.10 - Raisonnement

Il est important d'avoir un langage rigoureux. Les langues humaines (françaises, arabes, anglaises et autres) sont souvent ambigües. Prenons l'exemple de la conjonction "ou" en français; au restaurant fromage ou dessert signifie l'un ou l'autre mais pas les deux. Par contre si dans un jeu de carte on cherche les as ou les cœurs alors il ne faut pas exclure l'as de cœur. Autre exemple : que répondre à la question As-tu 100 DA en poche? si l'on dispose de 200 DA? Il y a des notions difficiles à expliquer avec des mots, par exemple la continuité d'une fonction est souvent expliquée par :

- On trace le graphe sans lever le crayon. Il est clair que c'est une définition peu satisfaisante.
- $f : I \rightarrow R$ est continue en un point $x_0 \in I$ avec $\forall \varepsilon > 0\ \exists \delta > 0\ \forall x \in I\ (|x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| < \varepsilon)$.

  La démarche d'explication, c'est le raisonnement.

Par exemple, "Est-ce qu'une augmentation de 20%, puis de 30% est plus intéressante qu'une augmentation de 50% ?". On peut penser oui ou non, mais pour en être sûr il faut suivre une démarche logique qui mène à la conclusion. Cette démarche doit être convaincante pour nous mais aussi pour les autres. On parle de raisonnement. Les mathématiques sont un langage pour s'exprimer rigoureusement, adapté aux phénomènes complexes, qui rend les calculs exacts et vérifiables. Le raisonnement est le moyen de valider ou d'infirmer une hypothèse et de l'expliquer à autrui.

En logique, une assertion est une phrase ou une proposition. C'est un énoncé qui peut être soit vrai, soit faux, pas les deux en même temps.

- **Exemples** :

1. Il pleut.
2. Je suis plus grand que toi.
3. 2+2=4
4. 2×3=7
5. Pour tout $x \in R$, on a $x^2 \geq 0$.

Un prédicat est une assertion qui dépend d'un ou plusieurs paramètres. Une tautologie est un prédicat toujours vrai. Une assertion P peut dépendre d'un paramètre x. L'assertion : $\forall x \in E$ P(x) est une assertion vraie lorsque les assertions P(x) sont vraies pour tous les éléments x de l'ensemble E. L'assertion : $\exists x \in E$ P(x) est une assertion vraie lorsque l'on trouve au moins un x de E pour lequel P(x) est vraie. La négation de $\forall x \in E$ P(x) et $\exists x \in E$ non P(x). La négation de $\exists x \in E$ P(x) et $\forall x \in E$ non P(x). Le tableau suivant résume la signification des quantificateurs.

Quantificateurs	Signification
$\exists$	Il existe
$\forall$	Pour tout
$\exists!$	Il existe un unique

En logique, la signification des connecteurs est la suivante sur le tableau suivant :

Connecteurs	Signification logique	Signification humaine
$\wedge$	Et logique	Conjonction
ν	Ou logique	Disjonction
$\Rightarrow$	Implication	Si alors
$\Leftrightarrow$	Équivalence	Si et seulement Si
$\neg$	Non logique	Négation
$\oplus$	Ou exclusif	Ou exclusif

Exemples de relations logiques :

$$\neg(p \wedge q) \equiv \neg p \vee \neg q$$

$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$

$$p \Rightarrow q \equiv \neg p \vee q$$

$$\neg(p \Rightarrow q) \equiv p \wedge \neg q$$

$$(p \Leftrightarrow q) \equiv (p \Rightarrow q) \wedge (q \Rightarrow p)$$

$$p \oplus q \equiv (p \wedge \neg q) \vee (\neg p \wedge q)$$

Une grammaire BNF (Backus_Naur Form) d'énoncés en logique propositionnelle, avec la priorité des connecteurs par ordre décroissant. La priorité des connecteurs est :$\neg$, $\wedge$, $\vee$, $\Rightarrow$ et $\Leftrightarrow$.

Énoncé → Énoncé Atomique|ÉnoncéComplexe

Énoncé Atomique → Vrai|Faux|P|Q|R|…

ÉnoncéComplexe → (Énoncé)|[Énoncé]

| ¬Énoncé

|Énoncé ∧ Énoncé

|Énoncé V Énoncé

|Énoncé ⇒ Énoncé

|Énoncé ⇔ Énoncé

Exercice

Effectuer la négation de la phrase suivante : "Tous les habitants de la ville d'El Oued qui ont des yeux noisettes gagneront au loto et prendront leur retraite avant 50 ans".

Solution

On procède d'abord à la formulation de la phrase :

H = tous les habitants de la ville d'El Oued
Y = des yeux noisettes
G = gagneront au loto
R = retraite avant 50 ans

$$\forall h \in H \wedge Y \mid h \in G \text{ et } h \in R$$

La négation de l'expression sera donc :

$$\exists h \in H \wedge Y \mid h \notin G \text{ ou } h \notin R$$

La négation de la phase devient : "Il existe un habitant de la ville d'El Oued qui a des yeux noisettes ne gagnera pas au loto ou ne prendra pas sa retraite avant 50 ans".

Exemple

Soient les assertions suivantes :

1. Les étudiants Master M2 apprendront si et seulement s'ils assistent à mon cours de raisonnement et décision.
2. Si on assiste à mon cours de raisonnement et décision, on peut s'y intéresser.
3. On s'intéresse à mon cours de raisonnement et décision ou on s'absente.
4. Si on ne s'absente pas, les étudiants Master M2 apprendront et ils n'assisteront pas à mon cours de raisonnement et décision.

Vérifier la consistance de ces quatre propositions.

Solution

1. Les étudiants Master M2 apprendront si et seulement s'ils assistent à mon cours de raisonnement et décision.
   a. p = Les étudiants Master M2 apprendront et q = ils assistent à mon cours de raisonnement et décision.
   b. La proposition 1 devient $p \Leftrightarrow q$.
2. Si on assiste à mon cours de raisonnement et décision, on peut s'y intéresser.
   a. q = On assiste à mon cours de raisonnement et décision et r = on peut s'y intéresser
   b. La proposition 2 devient $q \Rightarrow r$
3. On s'intéresse à mon cours de raisonnement et décision ou on s'absente.
   a. r = On s'intéresse à mon cours de raisonnement et décision et s = on s'absente
   b. La proposition 3 devient r v s
4. Si on ne s'absente pas, les étudiants Master M2 apprendront et ils n'assisteront pas à mon cours de raisonnement et décision.
   a. $\neg s$ = Si on ne s'absente pas, p = les étudiants Master M2 apprendront et $\neg q$ = ils n'assisteront pas à mon cours de raisonnement et décision.
   b. La proposition 4 devient $\neg s \Rightarrow (p \wedge \neg q)$

Pour établir la consistance de ces quatre propositions, il suffit d'établir leur table de vérité comme sur le tableau suivant :

p	q	r	s	$p \Leftrightarrow q$	$q \Rightarrow r$	r v s	$\neg s \Rightarrow (p \wedge \neg q)$	$p \wedge \neg q$
0	0	0	0	1	1	0	0	0
0	0	0	1	1	1	1	1	0
0	0	1	0	1	1	1	0	0
0	0	1	1	1	1	1	1	0
0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	1	1	0
0	1	1	0	0	1	1	0	0
0	1	1	1	0	1	1	1	0
1	0	0	0	0	1	0	1	1
1	0	0	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1
1	0	1	1	0	1	1	1	1
1	1	0	0	1	0	0	0	0
1	1	0	1	1	0	1	1	0
1	1	1	0	1	1	1	0	0
1	1	1	1	1	1	1	1	0

$p \Leftrightarrow q$ Il faut que p et q soient identique pour qu'elle vraie.
$q \Rightarrow r$ c'est faux uniquement dans le cas où q = 1 et r = 0.
r v s est vraie pour cela il suffit que r ou s soit vraie
$p \wedge \neg q$ est vraie si les deux sont vraies
$\neg s \Rightarrow (p \wedge \neg q)$ c'est faux uniquement dans le cas où $\neg s = 1$ et $p \wedge \neg q = 0$
Pour que les quatre propositions soient consistante, il faut de prendre les lignes à 1 des quatre propositions.
On voit que seulement dans trois cas, les propositions sont consistantes.