Cours
Intelligence Artificielle

M. Ramdani

Intelligence Artificielle

Définition-généralités-Historique

L'IA est la Science dont le but est de faire faire par une machine des tâches que l'homme accomplit en utilisant son intelligence

IA apparue en 1956 (Informatique Heuristique)

L'IA s'attaquent à des problèmes dont on ne connaît pas un algorithme (description des suites d'actions à entreprendre afin d'arriver à une solution en un temps raisonnable)

Définition-généralités-Historique

Ne relève pas de l'IA

- Un programme qui résoud une équation de second degré

Relève de l'IA

- Un programme qui fait de l'intégration symbolique

- Jeu d'échecs
  - On ne connaît pas une méthodes conduisant au meilleur coup à jouer → la situation considérée
  - Nombre de situation possible 10^{120}

Impossibilité tout examen exhaustif ou toute mémorisation

- Jeu de Go

Historique

Le rêve des machines intelligentes dates de l'antiquité

XVII
Descartes introduit l'idée de « l'animal machine », qui aurait certaines activités humaines mais pas toute l'intelligence

1912
Torres y Quevedo réalise un automate pour jouer les finales R+ T contre R par une méthode qui peut gagner contre toute défense

1945
Zuse (père des ordinateurs) programme les règles du jeu d'échecs

1949
Shannon propose une méthode pour jouer aux échecs

1950
Türing, un des premiers informaticiens, l'affine et simule à la main

Historique

1945
Newel,Show et Simon et des psychologues: projet de Programme d'échecs

⇨ Création d'un langage pour manipuler des informations symboliques : IPL1 (1956), père du LISP (Mc Carthy 1960)

1956 Newel, Show et Simon
LOGIC THEORIST est le premier programme de démonstration en logique de propositions. Le nom intelligence artificielle est introduit

1958
Newel et Simon pensent qu'avant 1968, un programme sera champion d'échecs et démontrera un important théorème mathématiques!

Réalisations

1960
Gelertner réalise un programme qui démontre des théorèmes de géométrie
« un triangle qui a deux angles égaux a aussi deux côtés égaux ».

1965
La méthode de résolution est utilisée en démonstration automatique de théorème, en vérification de programme,
En manipulation d'objets:
⇨ Langage PROLOG (Colmerauer 1971)

1967 Greenblatt
Le premier programme d'échecs. Il bat un joueur normal

Réalisations

1970 ⇙

• **Avant** les méthodes sont des améliorations du combinatoire on restreint l'énumération exhaustive à l'aide du bon sens, des fonctions d'évaluations et d'heuristiques

• **Après** les chercheurs sont convaincus que les programmes doivent avoir une connaissance approfondies du domaine étudié;

D'où les problèmes:

- **Quelles connaissances?**
- **Comment les donner?**
- **Comment les représenter?**
- **Comment les utiliser?**

Un programme d'IA doit avoir toute la connaissance nécessaire et ne l'utiliser qu'à bon escient

Réalisations

Systèmes experts

Ils permettent l'utilisation des connaissances

DENDRAL (Buchanan – Surtheland – Feigenbaul 1969)

Travaille en chimie. IL obtient la formule chimique développée d'un corps à partir d'un spectrogramme de masse

MYCIN (Shortliffe 1976)

Fait un diagnostic et propose une thérapeutique en médecine (infections bactériennes du sang)

Prospector DUDA 1979

Aide le géologue à évaluer l'intérêt d'un site en vue de prospection minière

Caractéristique des Systèmes experts

Séparation entre les connaissances nécessaires (bases de connaissances) et le programme qui permet de les utiliser (moteur d'inférence)

Domaines D'IA

⇨Ils manipulent des informations symboliques et non numériques (lettres, mots, signes, dessins, concepts, connaissances).
⇨ Ils impliquent des choix sans certitude entre plusieurs possibilités

- Démonstration Automatique de Théorèmes
  - performance sont > à celles d'un individu moyen
  - les formalisations dans ce domaine sont utiles pour les autres domaines
- Jeux
  - Les performances aux échecs sont > à celles d'un joueur moyen
- Résolution de problèmes
  - Il s'agit de poser, analyser, présenter et résoudre des problèmes dans des situations concrètes.
  - Emplois du temps- colorier une carte- casse tête logique
- Langage naturel
  - Traduction automatique difficile!
  - Compréhension et de la génération du langage naturel (connaissances syntaxiques sémantiques et pragmatiques)

Domaines D'IA

« le professeur a envoyé l'élève chez le censeur parce qu'il bavardait »

voulait le voir »

ne pouvait plus le supporter »

Reconnaissance des formes et perceptions

- Compréhension de la parole
- Lecture d'un manuscrit
- Reconnaissance d'odeur
- Vision artificielle

E I A O

Un système d'enseignement « intelligemment » Assisté par ordinateur doit pouvoir :

- Résoudre des problèmes
- Comprendre le raisonnement (correct ou erroné) de l'élève
- Établir un modèle de l'élève
- Lui proposer des questions en fonction de ce modèle

Résolution de problèmes
Représentations

Qu'est ce qu'un problème?

On a un problème quand:

- On se trouve dans une certaine situation,
- On a un but à atteindre
- On ne voit pas immédiatement la suite d'actions à accomplir pour atteindre ce but (résoudre le problème)

Exemple

Aller au sommet du Toubkal
Démontrer un théorème
Trouver le plus court chemin d'un point à un autre
Faire un diagnostic
Gagner une partie d'échecs

Résolution de problèmes
Représentations

Comment procéder?

POLYA 1957 « How to solve it »

Il faut

- Comprendre le problème
- Concevoir un plan
  - Trouver un rapport entre les données (la situation) et l'inconnue (le but)
  - Considérer éventuellement des problèmes auxiliaires
- Mettre le plan à exécution
- Examiner la solution

Tous est lié aux problèmes de représentations

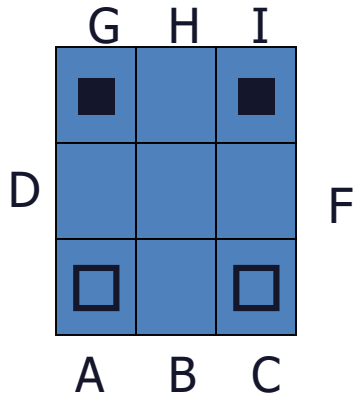
Représentations

De l'énoncé du problème, du problème, Des objets manipulés.

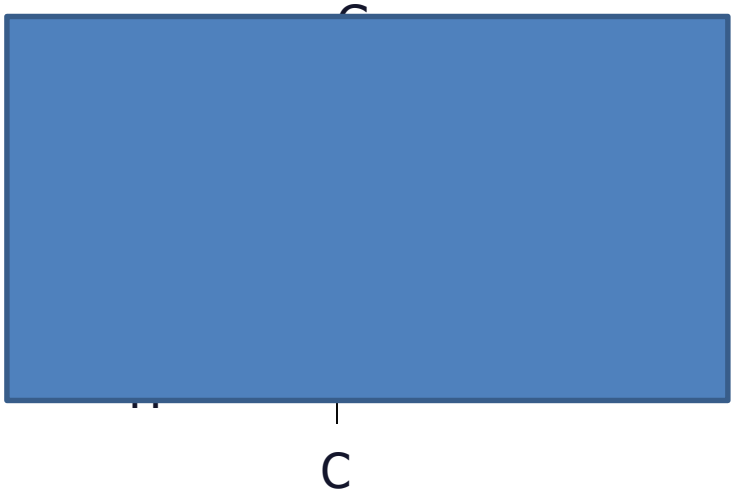
Il faut connaître aussi l'univers de recherche et les opérations permises

Résolution de problèmes
Représentations

Un problème peut devenir facile à résoudre si on utilise une « bonne » représentation:
On considère un échiquier 3x3 et quatre cavaliers disposés ainsi:



Le problème consiste à intervertir les cavaliers blancs et noirs par une succession de déplacements « légaux » des cavaliers

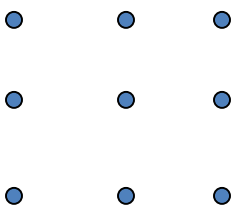


Résolution de problèmes
Représentations

Petits problèmes de casse-tête

• Former quatre triangles équilatéraux avec six allumettes

• passer par chacun des neuf points suivants, en traçant au plus quatre segments de droite sans lever le crayon.



Résolution de problèmes
Représentations

1 Problème du fermier, du renard, de l'oie et du grain

Un fermier veut traverser une rivière, il ne dispose que d'un Petit bateau dans lequel il ne peut transporter qu'une chose à la fois avec lui. Il ne peut laisser le renard et l'oie ni l'oie et le grain.
Comment faire la traversée?
Situation acceptable

2 Pb du crypto arithmétique

SEND + MORE =MONEY
S,E, D,N,M,O,R,Y $\in$ {0,……9}
S ≠ E ≠ D ≠ N ≠ M ≠ O ≠ R ≠ Y

Cours
Intelligence Artificielle

M. Ramdani

Intelligence Artificielle

Définition-généralités-Historique

L'IA est la Science dont le but est de faire faire par une machine des tâches que l'homme accomplit en utilisant son intelligence.

Plus globalement
L'IA est la Science de concevoir et de développer des systèmes capable de simuler l'intelligence

IA apparue en 1956 (Informatique Heuristique)

L'IA s'attaquent à des problèmes dont on ne connaît pas un algorithme (description des suites d'actions à entreprendre afin d'arriver à une solution en un temps raisonnable)

Ne relève pas de l'IA

- Un programme qui résoud une équation de second degré

Relève de l'IA

- Un programme qui fait de l'intégration symbolique

- Jeu d'échecs
  - On ne connaît pas une méthodes conduisant au meilleur coup à jouer → la situation considérée
  - Nombre de situation possible 10^{120}

Impossibilité tout examen exhaustif ou toute mémorisation

- Jeu de Go

Historique

Le rêve des machines intelligentes dates de l'antiquité

XIII
Aljazari (le robot serveur de thé)

XVII
Descartes introduit l'idée de « l'animal machine », qui aurait certaines activités humaines mais pas toute l'intelligence

1912
Torres y Quevedo réalise un automate pour jouer les finales R+ T contre R par une méthode qui peut gagner contre toute défense

1945
Zuse (père des ordinateurs) programme les règles du jeu d'échecs

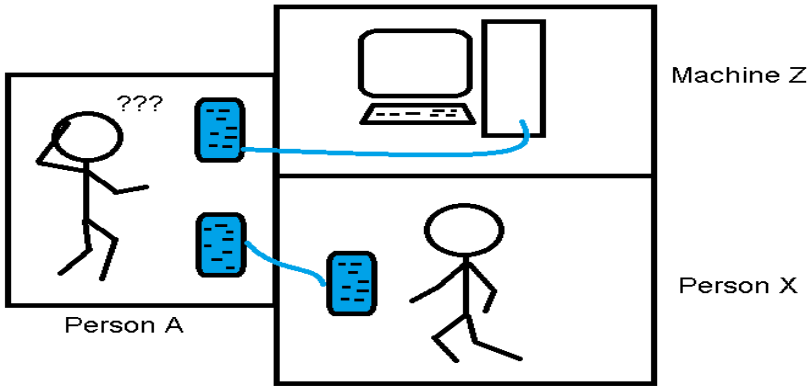
1945
Newel,Show et Simon et des psychologues: projet de Programme d'échecs

1949
Shannon propose une méthode pour jouer aux échecs

1950
Türing, un des premiers informaticiens, l'affine et simule à la main

Historique

Test de Turing (1950)



⇨ Création d'un langage pour manipuler des informations symboliques : IPL1 (1956), père du LISP (Mc Carthy 1960)

1956 Newel, Show et Simon
LOGIC THEORIST est le premier programme de démonstration en logique de propositions. Le nom intelligence artificielle est introduit

1958
Newel et Simon pensent qu'avant 1968, un programme sera champion d'échecs et démontrera un important théorème mathématiques!

Réalisations

1963
Gelertner réalise un programme qui démontre des théorèmes de géométrie (automatique)
« un triangle qui a deux angles égaux a aussi deux côtés égaux ».

1965
La méthode de résolution est utilisée en démonstration automatique de théorème, en vérification de programme, En manipulation d'objets:
⇨ Langage PROLOG (Colmerauer 1971)

1967 Greenblatt
Le premier programme d'échecs. Il bat un joueur normal (DEEP Blue 1996)

1966 Le robot : Shakey 66 (Algorithme Strips)

Réalisations

1970 ⇙

• **Avant** les méthodes sont des améliorations du combinatoire on restreint l'énumération exhaustive à l'aide du bon sens, des fonctions d'évaluations et d'heuristiques

• **Après** les chercheurs sont convaincus que les programmes doivent avoir une connaissance approfondies du domaine étudié;

D'où les problèmes:

- **Quelles connaissances?**
- **Comment les donner?**
- **Comment les représenter?**
- **Comment les utiliser?**

Un programme d'IA doit avoir toute la connaissance nécessaire et ne l'utiliser qu'à bon escient

Réalisations

Systèmes experts

Ils permettent l'utilisation des connaissances

DENDRAL (Buchanan – Surtheland – Feigenbaul 1969)
Travaille en chimie. IL obtient la formule chimique développée d'un corps à partir d'un spectrogramme de masse

MYCIN (Shortliffe 1976)
Fait un diagnostic et propose une thérapeutique en médecine (infections bactériennes du sang)

Prospector DUDA 1979
Aide le géologue à évaluer l'intérêt d'un site en vue de prospection minière

Caractéristique des Systèmes experts
Séparation entre les connaissances nécessaires (bases de connaissances) et le programme qui permet de les utiliser (moteur d'inférence)

Domaines D'IA

⇨Ils manipulent des informations symboliques et non numériques (lettres, mots, signes, dessins, concepts, connaissances).
⇨ Ils impliquent des choix sans certitude entre plusieurs possibilités

- Démonstration Automatique de Théorèmes
  - performance sont > à celles d'un individu moyen
  - les formalisations dans ce domaine sont utiles pour les autres domaines
- Jeux
  - Les performances aux échecs sont > à celles d'un joueur moyen
- Résolution de problèmes
  - Il s'agit de poser, analyser, présenter et résoudre des problèmes dans des situations concrètes.
  - Emplois du temps- colorier une carte- casse tête logique
- Langage naturel
  - Traduction automatique difficile!
  - Compréhension et de la génération du langage naturel (connaissances syntaxiques sémantiques et pragmatiques)

Domaines D'IA

« le professeur a envoyé l'élève chez le censeur parce qu'il bavardait »

voulait le voir »

ne pouvait plus le supporter »

Reconnaissance des formes et perceptions

- Compréhension de la parole
- Lecture d'un manuscrit
- Reconnaissance d'odeur
- Vision artificielle

E I A O

Un système d'enseignement « intelligemment » Assisté par ordinateur doit pouvoir :

- Résoudre des problèmes
- Comprendre le raisonnement (correct ou erroné) de l'élève
- Établir un modèle de l'élève
- Lui proposer des questions en fonction de ce modèle

Résolution de problèmes
Représentations

Qu'est ce qu'un problème?

On a un problème quand:

- On se trouve dans une certaine situation,
- On a un but à atteindre
- On ne voit pas immédiatement la suite d'actions à accomplir pour atteindre ce but (résoudre le problème)

Exemple

Aller au sommet du Toubkal
Démontrer un théorème
Trouver le plus court chemin d'un point à un autre
Faire un diagnostic
Gagner une partie d'échecs

Résolution de problèmes
Représentations

Comment procéder?

POLYA 1957 « How to solve it »

Il faut

- Comprendre le problème
- Concevoir un plan
  - Trouver un rapport entre les données (la situation) et l'inconnue (le but)
  - Considérer éventuellement des problèmes auxiliaires
- Mettre le plan à exécution
- Examiner la solution

Tous est lié aux problèmes de représentations

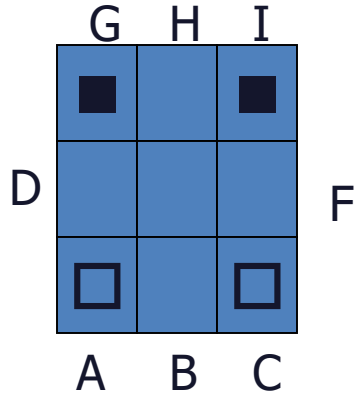
Représentations

De l'énoncé du problème, du problème, Des objets manipulés.

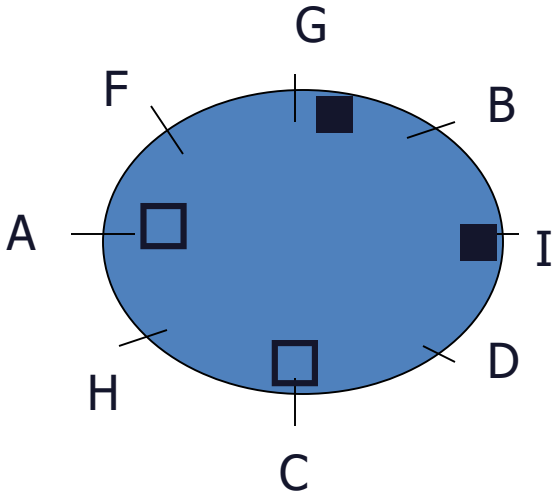
Il faut connaître aussi l'univers de recherche et les opérations permises

Résolution de problèmes
Représentations

Un problème peut devenir facile à résoudre si on utilise une « bonne » représentation:
On considère un échiquier 3x3 et quatre cavaliers disposés ainsi:



Le problème consiste à intervertir les cavaliers blancs et noirs par une succession de déplacements « légaux » des cavaliers

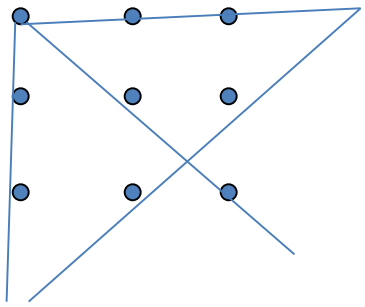


Résolution de problèmes
Représentations

Petits problèmes de casse-tête

• Former quatre triangles équilatéraux avec six allumettes

• passer par chacun des neuf points suivants, en traçant au plus quatre segments de droite sans lever le crayon.



Résolution de problèmes
Représentations

1 Problème du fermier, du renard, de l'oie et du grain

Un fermier veut traverser une rivière, il ne dispose que d'un Petit bateau dans lequel il ne peut transporter qu'une chose à la fois avec lui. Il ne peut laisser le renard et l'oie ni l'oie et le grain.
Comment faire la traversée?
Situation acceptable

2 Pb du crypto arithmétique

SEND + MORE =MONEY
S,E, D,N,M,O,R,Y $\in$ {0,……9}
S ≠ E ≠ D ≠ N ≠ M ≠ O ≠ R ≠ Y

4 Pb des missionnaires et les cannibales

5 pb 2 cruches de 3 et 5 L on veut mesurer 4 L

Résolution de problèmes
Représentations

Conclusion:

On peut attendre d'une bonne représentation les caractéristiques suivantes:

- Elle doit rendre explicite les choses importantes
- Elle doit traduire les contraintes inhérentes au problème
- Elle doit être complète et concise
- Elle doit se prêter à une représentation en mémoire et être calculable

En intelligence artificielle il existe plusieurs techniques « classiques » de représentation:

- Logique des propositions
- Logique des prédicats
- Frames
- Réseaux sémantiques
- Les objets

ENUMERATIONS
RECHERCHES ARBORESCENTES

I ENUMERATIONS

I.1 Enumération explicite

• La méthode la plus simple quand on ne sait pas a priori faire autrement pour résoudre un pb combinatoire

•Un problème est combinatoire, s'il consiste à chercher un élément x dans X, satisfaisant un ensemble de contraintes X est appelé espace de recherche (fini)

Dans la réalité il y a trois possibilité, on cherche :

1. un élément quelconque satisfaisant les contraintes.
2. Tous les éléments
3. Un élément optimal (en un sens à préciser)

Exemple

1. Chercher un mot de passe
2. Cryptarithmétique, problème des huit dames
3. Emploie du temps

Une méthode pour résoudre le problème de type 1:

Prendre tous les éléments l'un après l'autre de l'espace de recherche;
regarder s'il convient; si oui, le problème est résolu, on s'arrête;
Sinon continuer

Pour le problème de type 2

Continuer jusqu'au bout de la recherche.

Pour le problème de type 3

S'arrêter si on peut prouver qu'un élément trouvé est optimal,
Sinon continuer

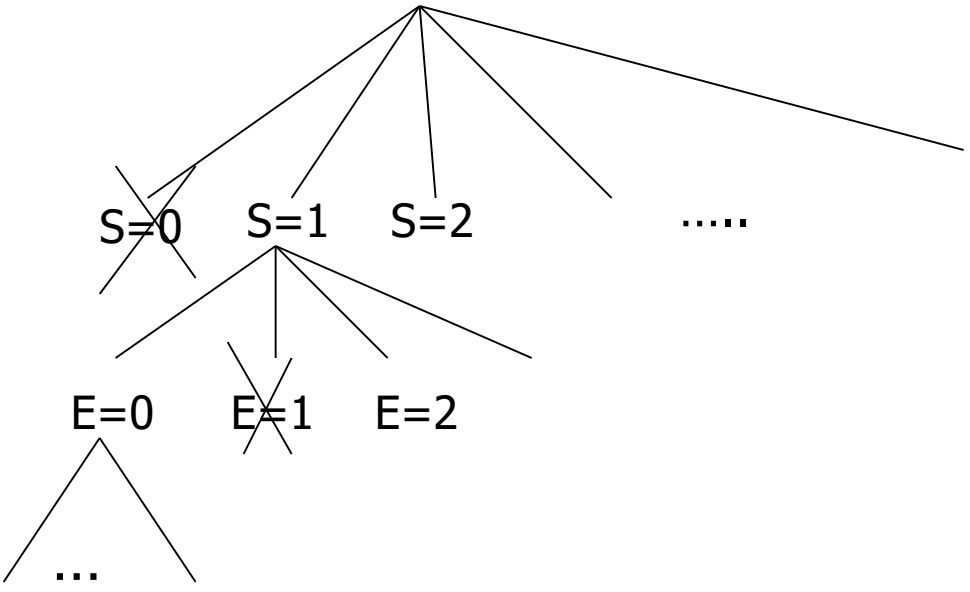
Le problème cryptarithmétique SEND + MORE = MONEY
On peur essayer tous les 8-uplets possibles pour (S,E,N,D,M,O,R,Y) et regarder s'ils vérifient les contraintes.

- Le programme comportera 8 boucles imbriquées
- 10^8 passages

I.2 Enumération implicite

• C'est une amélioration de la méthode explicite lorsqu'on peut construire les x morceau par morceau

Exemple
Le problème cryptarithmétique SEND + MORE = MONEY

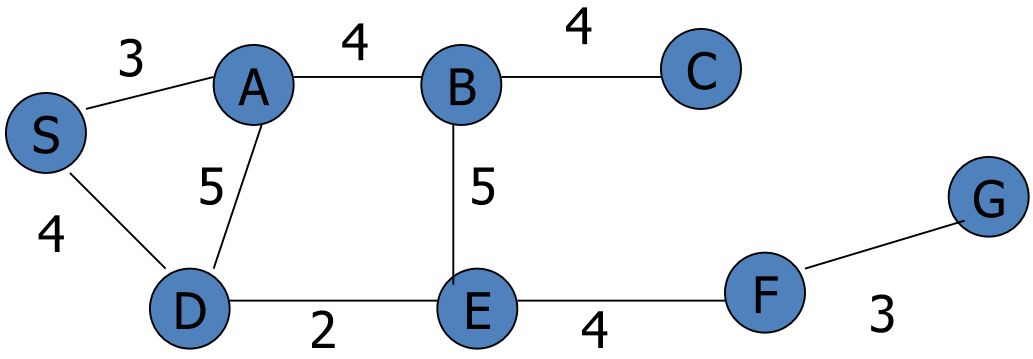


•On choisit les valeurs l'une après l'autre, on s'arrête dés que l'on trouve qu'une contrainte n'est pas satisfaite

2 Recherche arborescente

2.1 recherche d'un chemin d'un point à un autres

On travail sur l'exemple suivant :



Le problème consiste à trouver un (resp. le plus court) chemin de S à G.

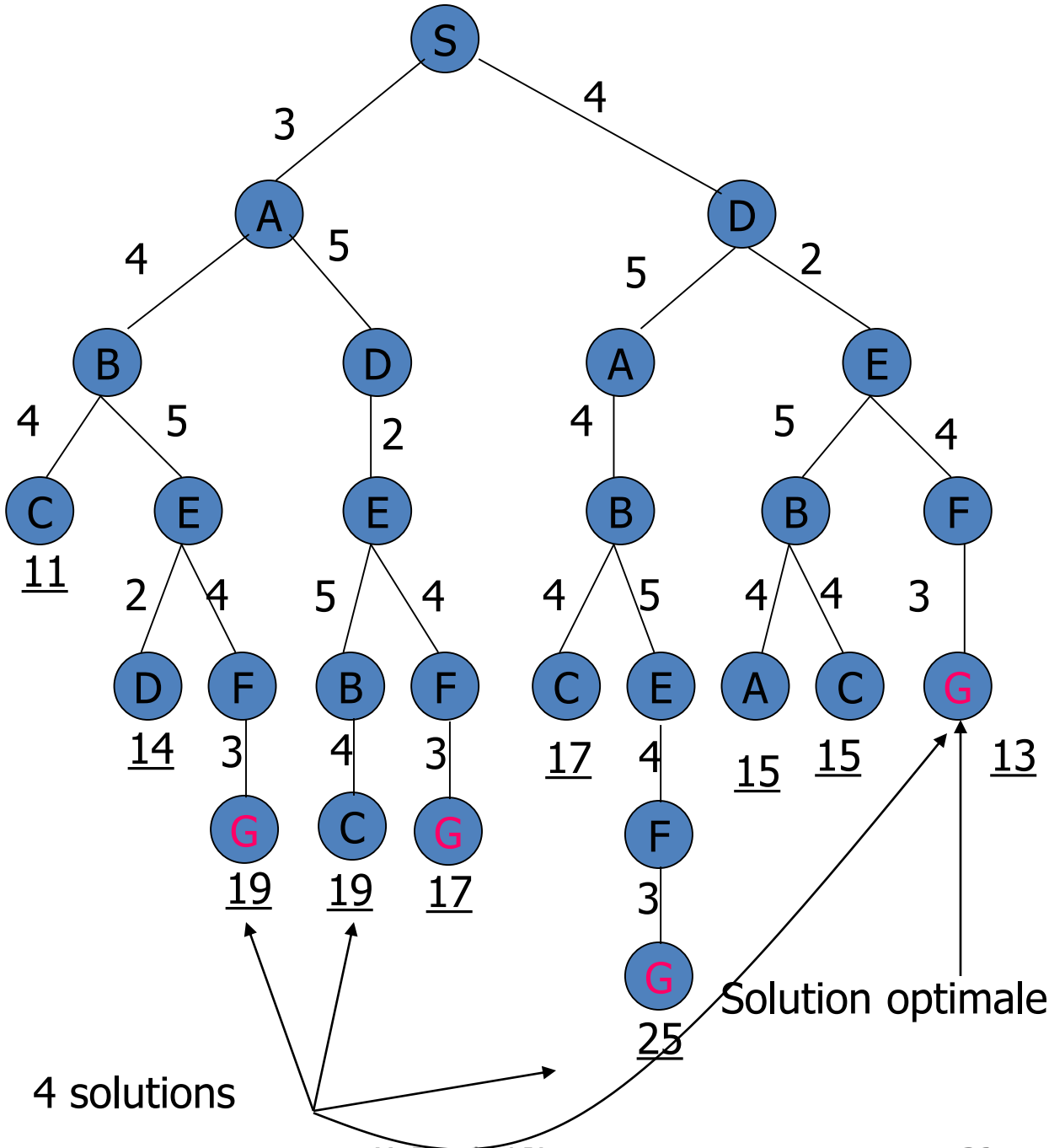
Remarque:

Le problème dit de recherche dans un graphe d'état consiste à trouver un chemin d'un nœud initial vers un nœud final

Les graphes d'états qu'on trouve en IA ont la caractéristique d'être gros (voir infinis):

- Pas de représentation en mémoire

•l'arbre de recherche obtenu à partir du graphe précédent, on ne crée pas les nœud qui est identique à un de ses ancêtres



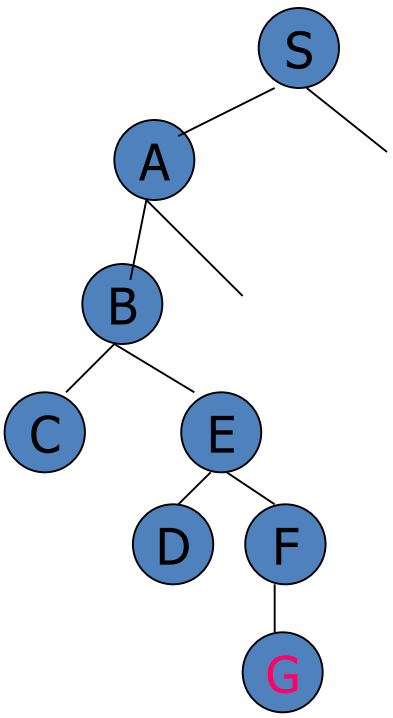
2.2 Recherche d'une solution

1. Recherche en profondeur

On parcourt l'arbre en profondeur d'abord, de gauche à droite:

Former une liste constituée de la seule racine
Tant que le liste n'est pas vide
    Chercher si le premier élément de la liste est le but à atteindre
    Si oui s'arrêter avec succès
    Sinon, enlever cet élément et le remplacer, en début de liste, par ses fils.
Si la liste est vide, échec.

S
A D
BDD
CEDD
EDD
DFDD
GDD



Danger si l'arbre est profond ou a des branches infinies

2. Méthode du gradient (« Hill climbing »)

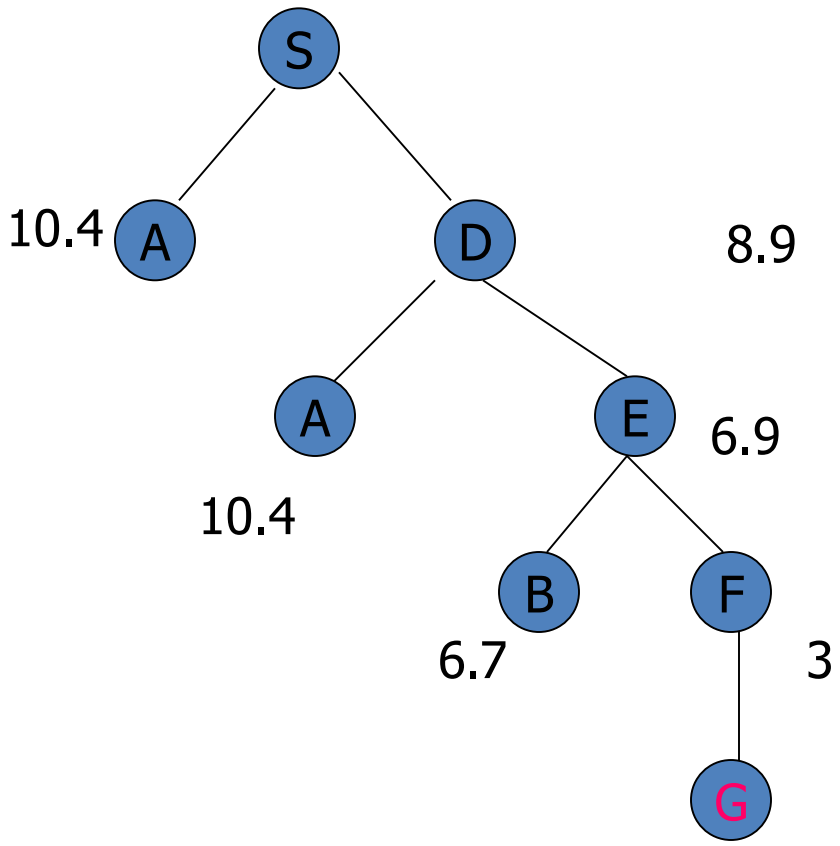
- C'est une recherche en profondeur, on choisit de développer le nœud le plus près au but.
- On devra avoir une fonction d'évaluation de la distance au but

```
Former une liste constituée de la seule racine
Tant que le liste n'est pas vide
    Chercher si le premier élément de la liste est le but à atteindre
    Si oui s'arrêter avec succès
    Sinon                    enlever cet élément,
                             ordonner la liste de ses fils selon la fonction d'évaluation
                             et le remplacer, en début liste, par le
                 meilleur    de ses fils.
Si la liste est vide, échec.
```

L'efficacité de la méthode dépend de la qualité de cette fonction d'évaluation

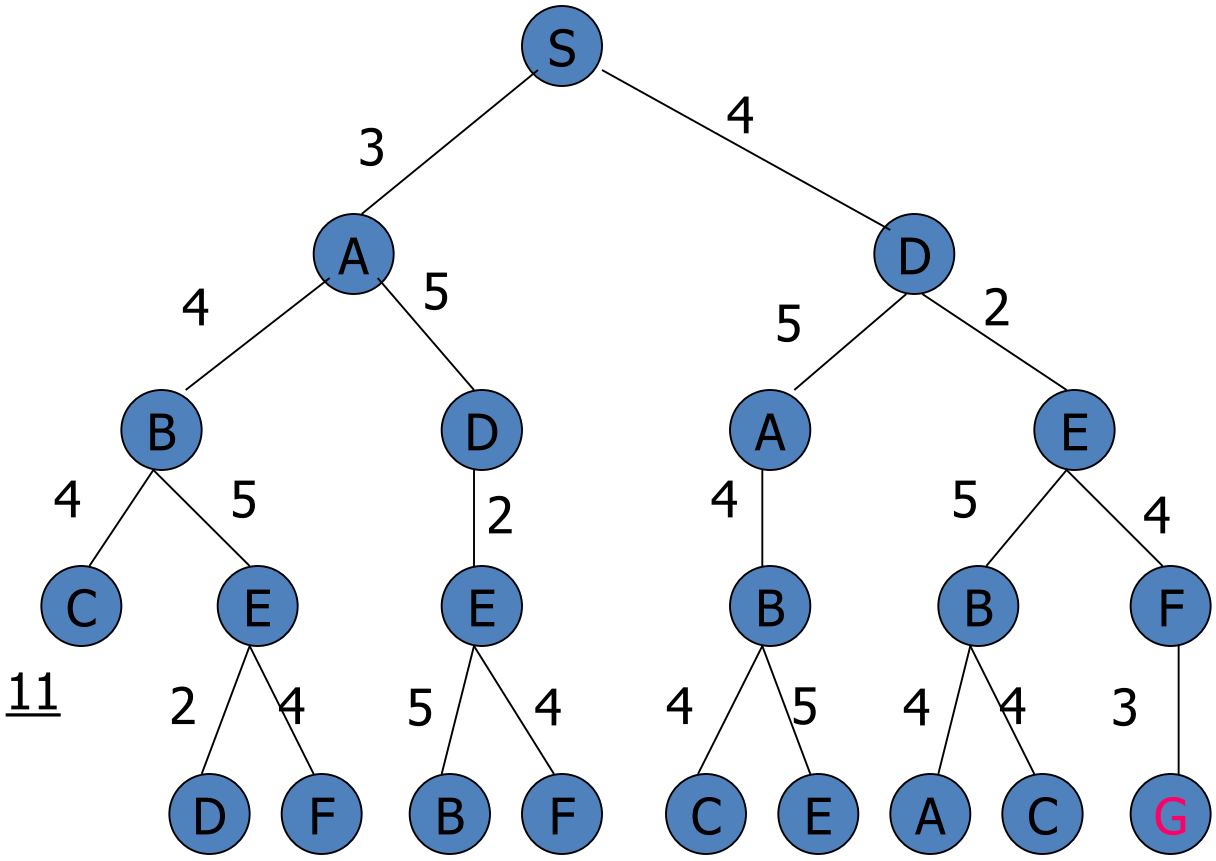
Supposons que les distances à G à vol d'oiseau:

S	A	B	C	D	E	F	G
12	10.4	6.7	4	8.9	6.9	3	0



3. Recherche en largeur

Former une liste constituée de la seule racine
Tant que le liste n'est pas vide
    Chercher si le premier élément de la liste est le but à atteindre
    Si oui s'arrêter avec succès
    Sinon, enlever cet élément et ajouter ses fils en fin de liste.
Si la liste est vide, échec.



Cette méthode est bonne s'il y a des branches très longues

4. Recherche du meilleur premier (best first search)

Former une liste constituée de la seule racine
Tant que le liste n'est pas vide
    Chercher si le premier élément de la liste est le but à atteindre
    Si oui s'arrêter avec succès
    Sinon, enlever cet élément et ajouter ses fils et trier la liste.
Si la liste est vide, échec.

- Cette méthode donne assez souvent, la même solution que la méthode du gradient

2.3 recherche de la meilleur solution

1. Procédure du « British Museum »

- On explore tout, en profondeur ou en largeur.
- S'i l y a *b* branches à chaque nœud et *d* est la longueur des branches, on examinera b^d nœuds

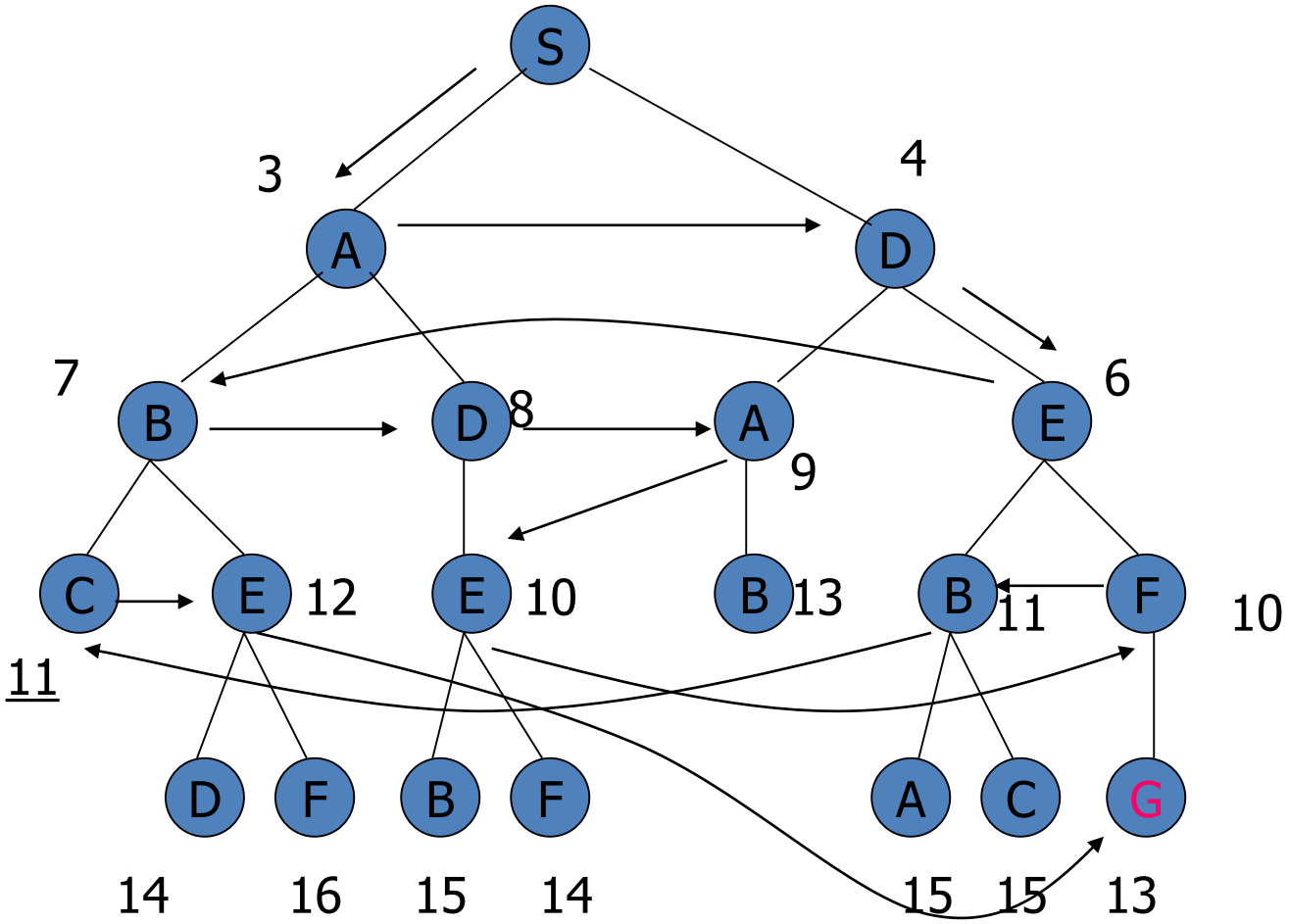
A éviter en général

2. Ramification bornée (branch and bound search)

On évalue à chaque nœud créé, le coût de la racine à ce nœud. On développe le nœud de plus faible coût

```
Former une liste de chemins constituée de la seule racine
Tant que le liste n'est pas vide
    Si le premier chemin atteint le but s'arrêter avec succès
    Sinon    retirer ce chemin de la liste
             former les nouveaux chemins possibles à partir de chemin en rajoutant une étape
             les ajouter à la liste
             trier la liste en fonction des coûts des chemins
Si la liste est vide, échec.
```

2. Ramification bornée (branch and bound search)(suite)



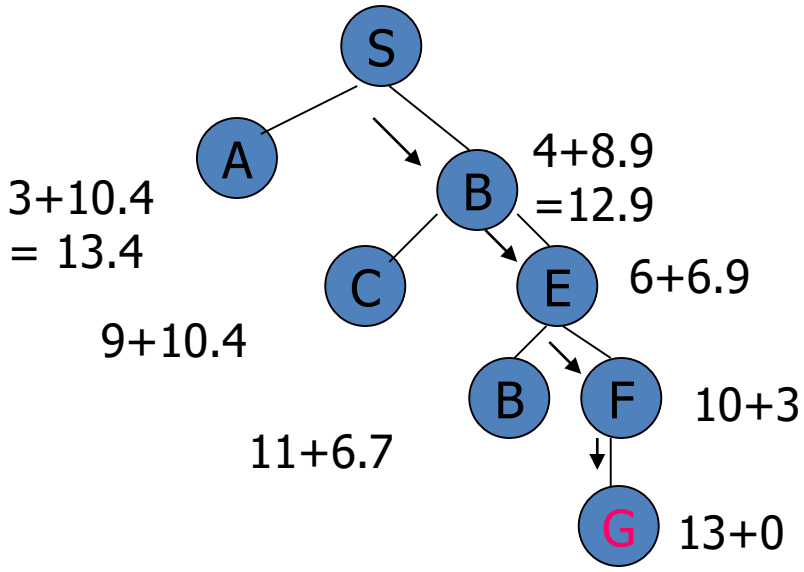
3. Amélioration par des estimation

• On remplace les coûts par une évaluation f définie de la façon suivante:

• Pour chaque nœud n, f(n)= g(n)+h(n)

Où g(n) est le coût réel pour atteindre n
h(n) est une estimation de la distance de n au but (heuristique)

• La procédure est identique à la précédente, sauf que le tri se fait avec cette fonction d'évaluation



•Si h est une sous-estimation, on trouve à coup sûr une solution optimal
•Si h est une sur-estimation, on peut ne pas l'obtenir

4. Autre amélioration:
élimination des chemins redondants (programmation dynamique)

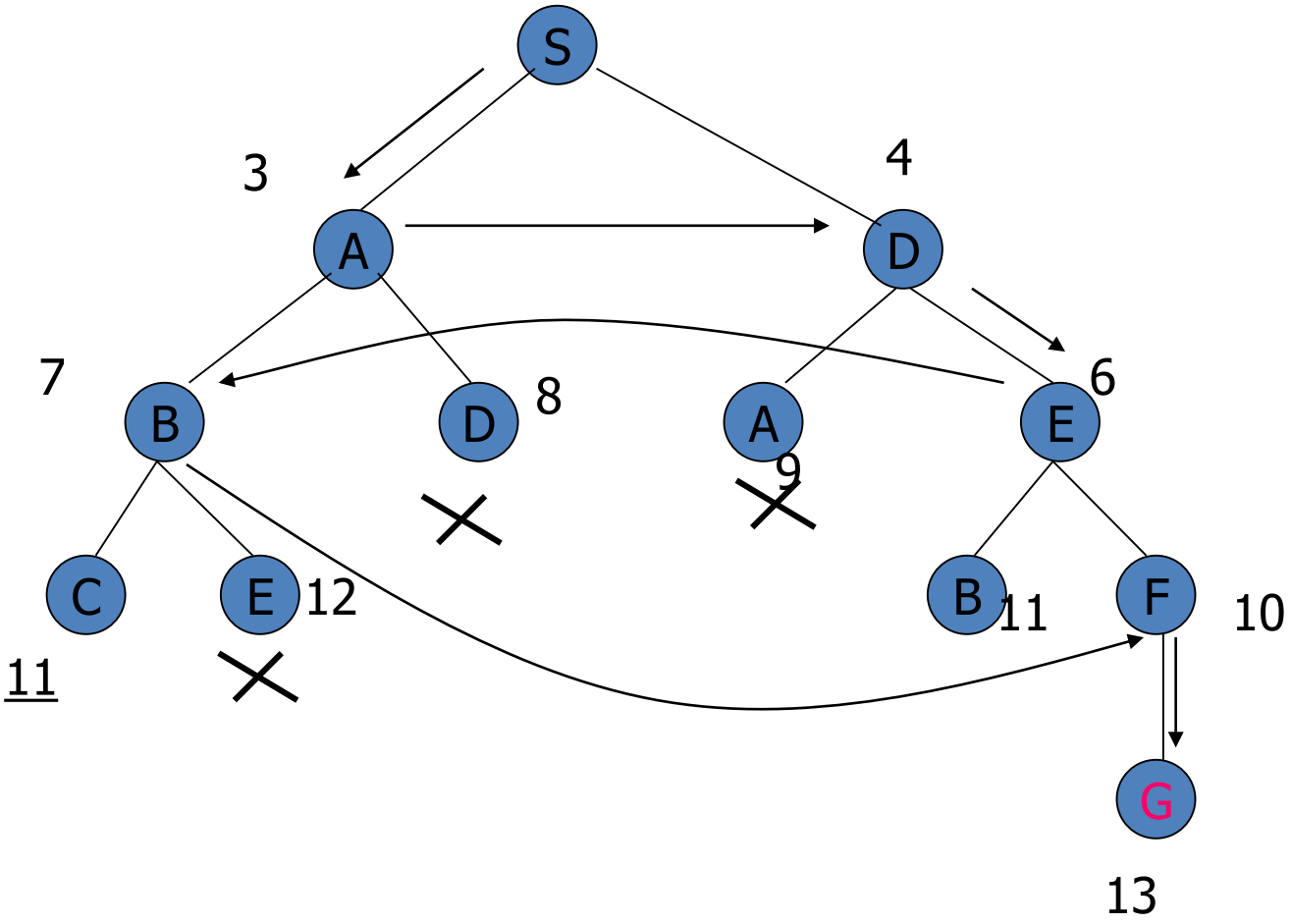
- Sur l'exemple, le chemin SAD ne peut donner une solution optimale puisque son coût réel est 8 et que le coût du chemin SD est 4. Il est inutile de garder SAD.

- Le principe général de la programmation dynamique:
  - Toute sous stratégie d'une stratégie optimale est optimale

    Min (S à G) = min (S à n) + min (n à G)

- Dans la procédure:
  - on regardera dans la liste des chemins si plusieurs atteignent le même nœud; si oui, on ne gardera que le meilleur

élimination des chemins redondants
Dans l'exemple:



L'ALGORITHME A*

• Cette méthode combine les deux améliorations précédente

• Dans la méthode A*, à chaque instant, on a une liste de nœuds en attente, ordonnée selon la fonction d'évaluation f, avec

$f(n) = g(n)+h(n)$

g(n) = coût du chemin allant de s à n

h(n) = estimation du coût de n au but

On désignera de plus par :

g*(n) = le coût optimal pour aller de s à n

h*(n) = le coût optimal pour aller de n au but

$f^*(n) = g^*(n)+h^*(n)$ est le coût optimal d'un chemin passant par n.

Remarque

$g^*(n) \leq g(n)$ et $h(n) \leq h^*(n)$

Si h est une sous-estimation, alors A* est admissible, est donne une solution optimale si elle existe

Exemples de jeux individuels
Arbres de recherches

Exemple1 :

But

1	2	3
8		4
7	6	5

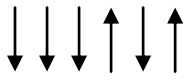
départ

2	8	3
1	6	4
7		5

- Développement en profondeur.
- Développement en largeur.
- Développement avec la méthode du gradient avec l'heuristique :
  - Le nombre de pions mal placés
  - la somme des distances des positions des pions par rapport à leur places dans le but.
- Développement avec la procédure A*

Exemple 2

On considère 6 flèches données en position S0 ; il s'agit de les amener à la position Sb, les seuls coups permis étant les retournements simultanés des deux flèches adjacentes



S0

Sb

Fonctions d'évaluations :

- Nombre de flèches mal placées
- La distance maximale entre deux flèches mal placées

Recherche dans un arbre de jeu:

⇨ Les jeux à deux joueurs
⇨ Problème du joueur quel est le meilleur coup qu'il peut jouer

Si on développe toute l'arbre :

- *L'arbre est trop grande pour être exploré de manière exhaustive*.

- Pour le Tic-tac-toe
  - ✓ Le nœud de départ à 9 successeur qui ont chacun 8 successeur D'où 362880 feuilles

- L'arbre de jeu d'échecs est estimé à 10^{120},
  - ➢ 10^{12}siecles pour explorer toute l'arbre

Algorithme minmax

- Exploration de l'arbre à un nombre n de niveaux
- on dit qu'un joueur d'échecs « il voit 4 ou 5 coups en avance »

- Profondeur limitée ➔jugement sur les feuilles à une profondeur n

- On utilise une fonction heuristique f à valeur dans Z

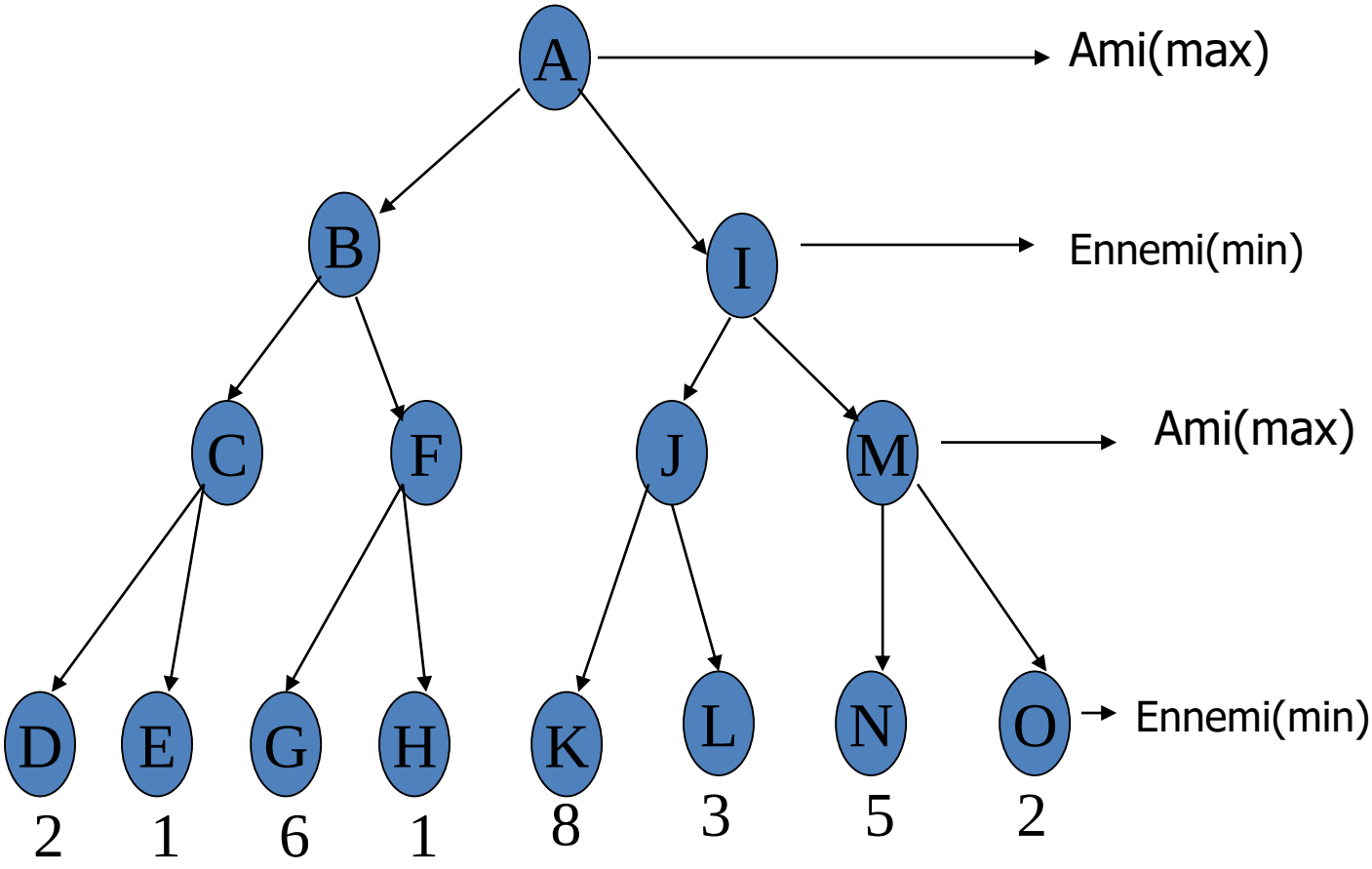
Algorithme minmax

• Une situation est d'autant considéré meilleur pour le joueur 1 que f(s) est grand.

⇨ Il s'agit dons de propager les valeurs calculées aux feuilles jusqu'à la racine.

⇨ Comme à chaque changement de niveau on change de joueur :
• On considère du point de vue du joueur 1 (Max), qui lui jouera toujours le coup de valeur maximum alors que son adversaire jouera toujours le coup de valeur minimum.

Algorithme minmax



Le nœud C est valué à max (val(D), val(E))
Le nœud B est valué à min (val(C) val(F))

Tic-Tac-toe

Supposons que Max marque les croix (X) et Min marque des cercles O, et que Max joue en premier.

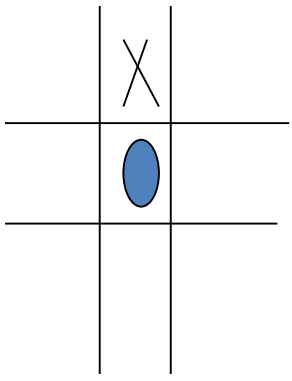
⇨Nous effectuons une recherche en largeur d'abord jusqu'à ce que tous Les nœuds au niveau 2 soient engendrés.

⇨ Applications d'une fonction statique aux positions associées à ces nœuds.

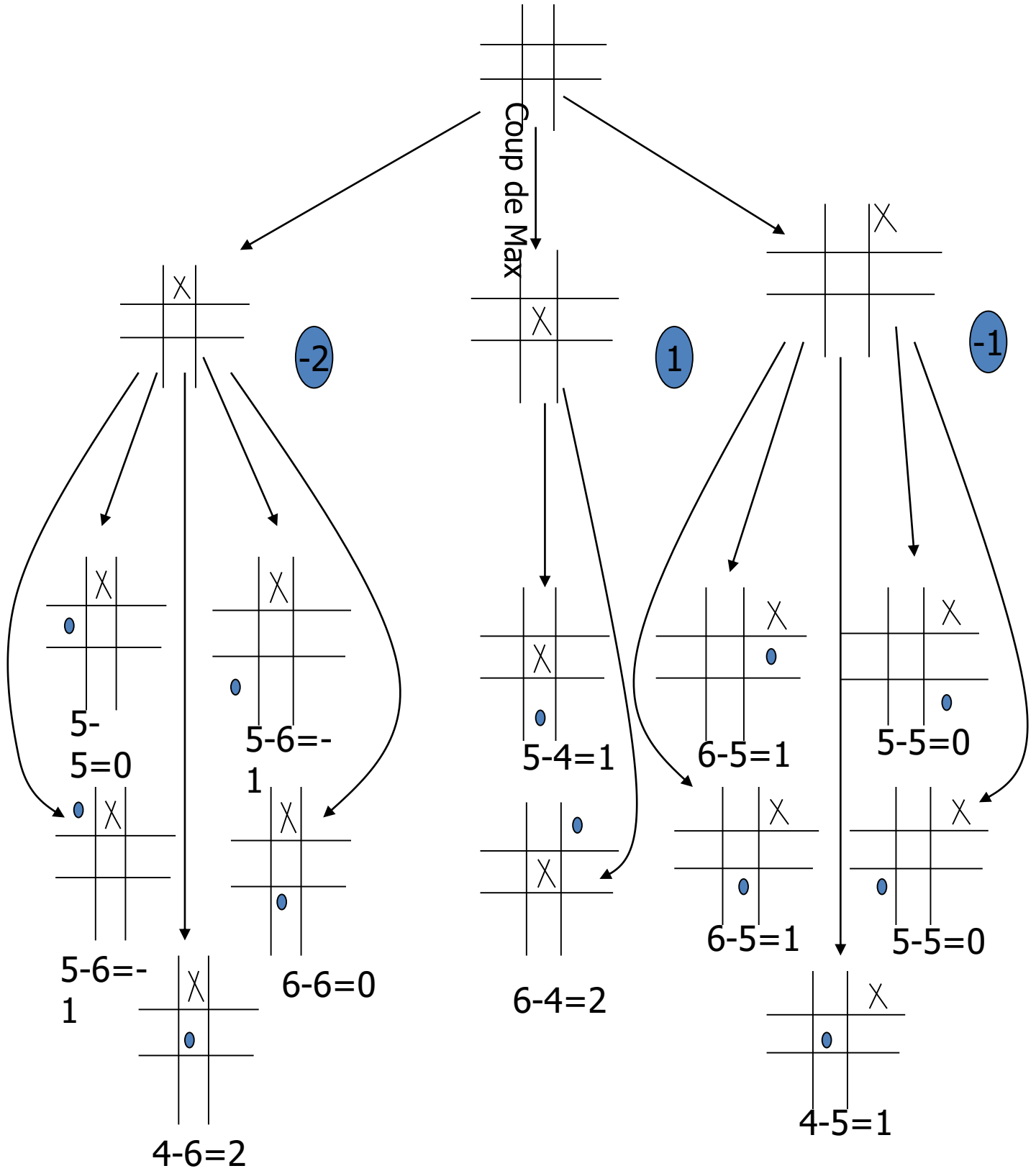
Tic-Tac-toe

f(p) fonction d'évaluation d'une position p est définie par :

- si p n'est pas gagnante ni pour Max ni pour min:
  f(p)= nombre de rangées, colonnes ou diagonales ouvertes pour Max - nombre de rangées colonnes ou diagonales ouvertes pour Min
- si p est une victoire pour Max
  f(p)= +∞
- si p est une victoire pour Min
  f(p)= - ∞



f(p)=4-6=-2

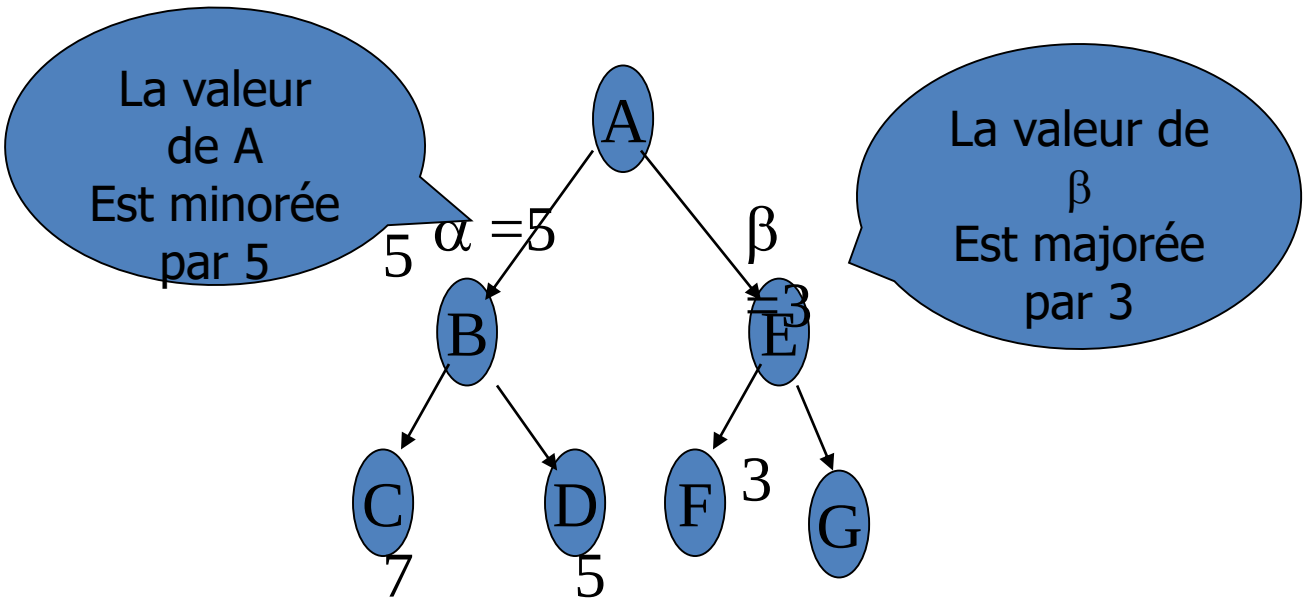

Coup de Max
-2
1
-1
5-5=0
5-6=-1
5-4=1
6-5=1
5-5=0
5-6=-1
6-6=0
6-4=2
6-5=1
5-5=0
4-6=2
4-5=1

Procédure alpha-Bêta

Pour Minmax

C'est après que l'arbre de recherche a été engendré que commence l'évaluation de la position.

⇨Une réduction considérables si on effectue des évaluations aux feuilles et qu'on calcule les valeurs propagées au moment où l'arbre est engendré



-Les valeurs alpha des nœuds Max ne peuvent jamais diminuer

-Les valeurs bêta des nœuds Min ne peuvent jamais augmenter

systèmes experts

systèmes à base de règles

• Un système expert est un programme conçu pour raisonner habilement à propos de tâches dont on pense qu'elles requièrent une expertise humaine considérable

• Les systèmes experts sont des logiciels destinés à remplacer ou assister l'homme dans les domaines où est reconnue une expertise humaine :

- Insuffisamment structuré pour constituer une méthode de travail

- La modélisation de la connaissance dans ces domaines va être effectuée en décomposant le savoir des experts humains en unités de travail appelés règles

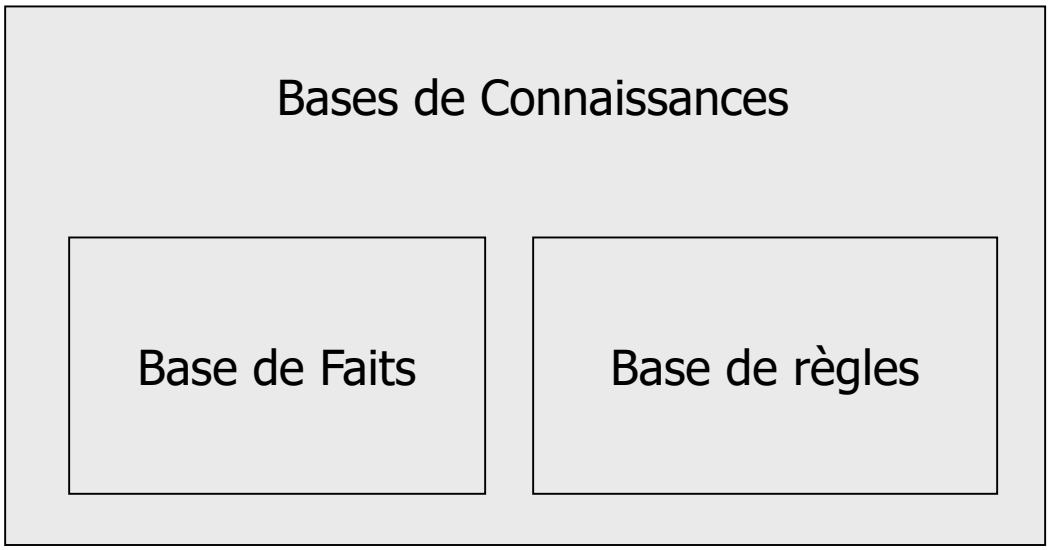
Un système expert est un système informatique capable de mettre en œuvre des règles décrites par des experts dans des domaines difficilement modélisables

Composants d'un système expert

Un système expert est composé de règles et d'un programme informatique permettant La mise en œuvre ces règles

- Langage de descriptions de règles
  - Proche du langage naturel
    si condition alors Action

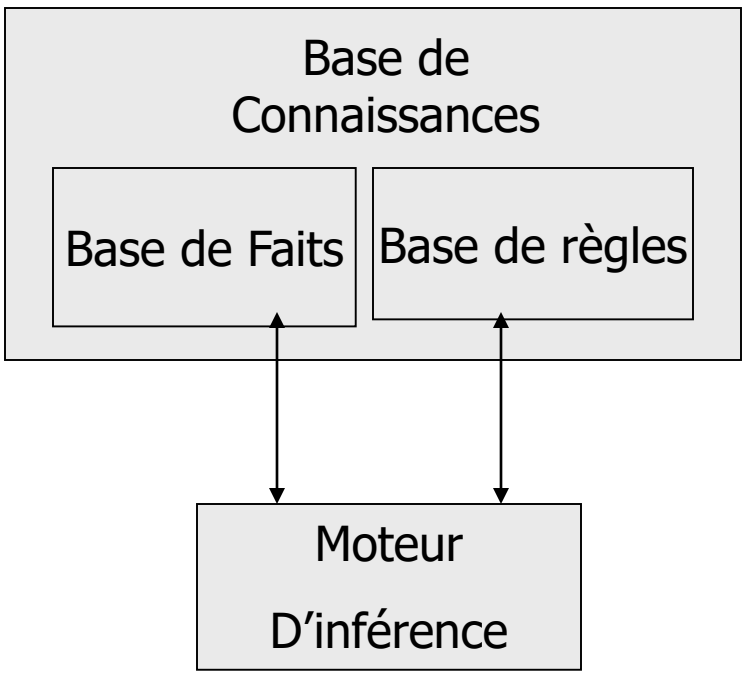
- Les différentes connaissances exprimées vont prendre place dans une base de connaissances.



Composants d'un système expert

Les faits et les règles sont mis en correspondance pour pouvoir effectuer des déductions et arriver éventuellement à une solution du problème initialement posé

Cette mise en correspondance est effectuée par un moteur d'inférence (indépendant du domaine)



Systèmes experts

Générateurs de systèmes experts

- OPS ,RETE, Prolog, Clips

OPS a permis de développer un ensemble de système expert tel que :
ACE : surveillance de câble téléphonique
AiRPLAN : décollage et appontage d'avions

Rédaction associative des règles

- Donner les informations au systèmes sans jamais expliciter leur utilisation:
  - Les données manipulées par les systèmes expert sont déclaratives

Les données manipulées par un programme informatique sont procédurales

Principe de fonctionnement des moteurs d'inférences

Un moteur d'inférence enchaîne des cycles de travail comportant chacun deux Phases :

Phase d'évaluation
Le moteur détermine s'il existe dans la base des règles, des règles à déclencher si oui qu'elles sont ces règles
Phase d'exécution
Le moteur déclenche les règles retenue par l'évaluation
L'arrêt

- **Phase d'évaluation (absence de règles déclenchables)**
- **Phase d'exécution (par la règle)**

Phase d'évaluation

- Sélection
  Déterminer à partir de base de fait (BF) et de Base de règles (BR),
  Un **Fi** $\subset$**BF** et **R1** $\subset$ **BR**

- Filtrage (Pattern Matching)
  Le moteur d'inférence compare la partie déclencheur de chacune des règles de R1 par rapport à l'ensemble de fait Fi:
  R2 $\subset$ **R1**

- Résolution de conflits
  Le moteur détermine les règles, soit R3 sous-ensemble de R2, qui doivent être effectivement « déclenchées »
  - Sélectionner les règles qui ont le moins servi
  - Les règles les moins complexes

Phase d'exécution

Le moteur d'inférence commande la mise en œuvre des actions définis par les règles de R3 Si R3 est non vide

Régime irrévocable

R3=∅ il y a des moteurs qui s'arrêtent

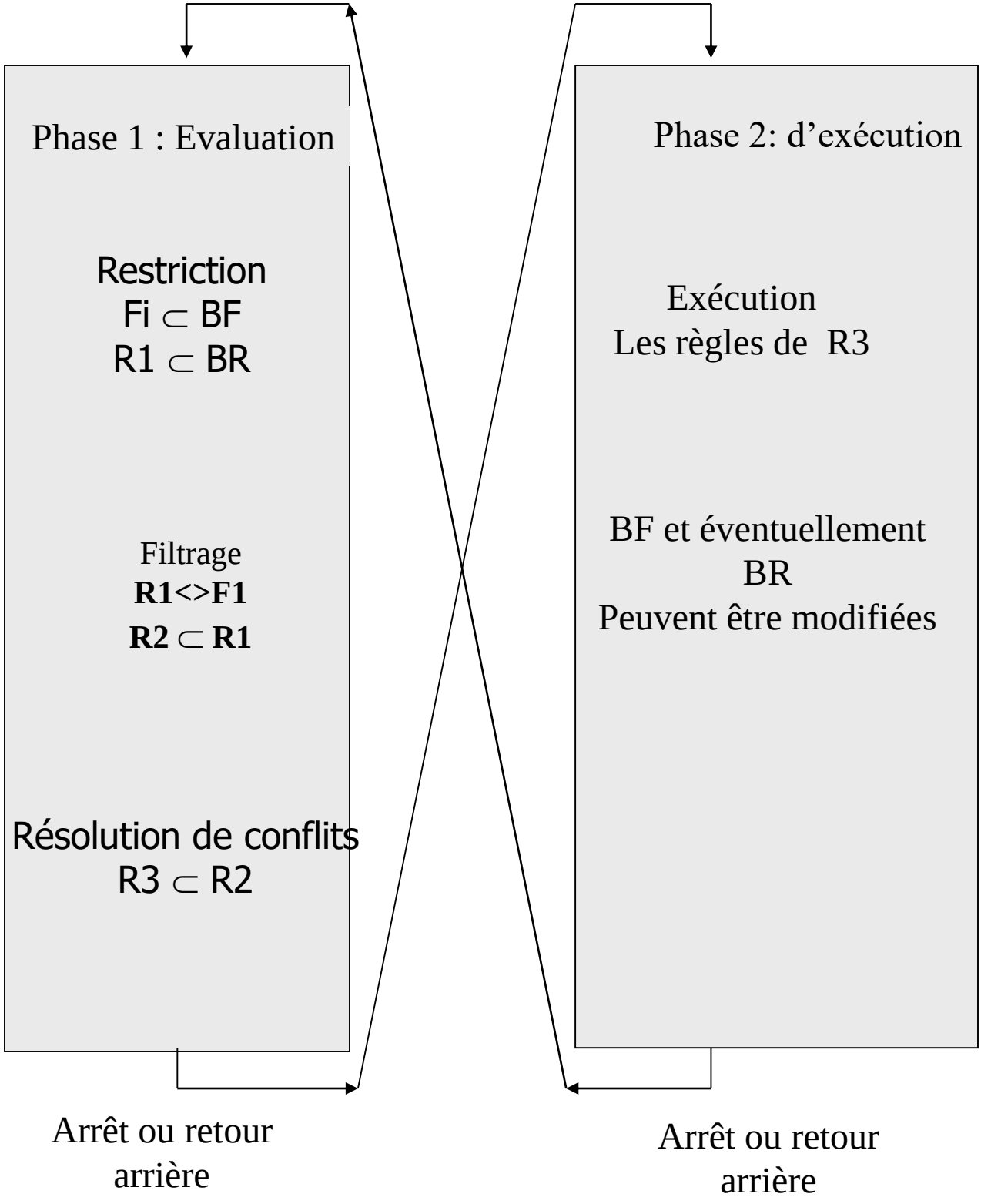
- C'est les moteurs à régime irrévocable

Régime par tentatives

Le moteur revient sur une résolution de conflit antérieur

- Retour arrière (backtrack)

Lorsqu'une impasse est rencontrée au cycle n, un retour arrière est réalisé au cycle n+ 1 par retour au cycle n-1 (retour arrière chronologique)
R3 à l'étape n+1 ≠ R3 à l'étape n-1


Phase 1 : Evaluation
Restriction
Fi ⊂ BF
R1 ⊂ BR
Filtrage
R1<>F1
R2 ⊂ R1
Résolution de conflits
R3 ⊂ R2
Phase 2: d'exécution
Exécution
Les règles de R3
BF et éventuellement
BR
Peuvent être modifiées
Arrêt ou retour
arrière
Arrêt ou retour
arrière

Remarque

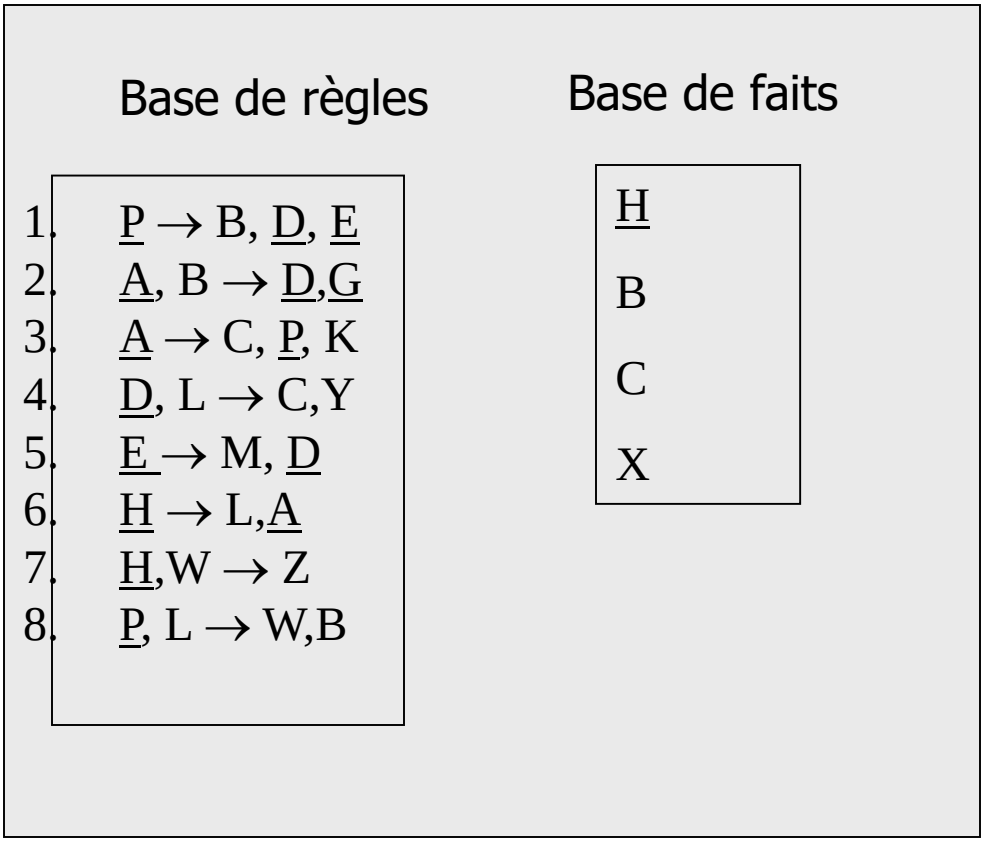
Le filtrage constitue R2 comme un ensemble de couples de la forme (<faits> <règle>):

- À chaque règle on associé un groupe de <faits> de F1 qui déterminent la compatibilité du déclencheur de la règle avec F1

- La résolution de conflits porte sur les couples (<faits>,<règle>) et non seulement sur les règles

Exemple d'un micro-système expert

Base de faits : 4 fait symbolisés par : B, C, X, H



Condition de déclenchement % BF → effet sur BF

H l'hypothèse à établir

1èr cycle

Seule la règle 6 est applicable soit R2={6} d'où
BF= {L, A, B, C, X}
Dés lors que la règle 6 est appliquée, l'ensemble de conflit de ce cycle devient vide.

2ème cycle

Nouvel R2 ordonnée selon l'ordre de filtrage {2, 3}
Priorité à la règle 2
BF = {D, G, L, B, C, X} R2 ={3}

3ème cycle

R2={4}, d'où BF ={Y, G, L, B, C, X}, et R2 devient vide

4ème cycle

R2 = ∅ le cycle ne peut être achevé par la phase d'exécution, retour arrière au cycle suivant

5ème cycle

Retour arrière (backtraking)

On revient à BF = {D, G, L, B, C, X} début de 3ème cycle on reprend R2 produit au 3ème Cycle, il en résulte un retour arrière

6ème cycle

La sélection/restriction restaure l'état de BF existant au début du deuxième cycle soit **BF= { L,A, B, C, X}, R2 = { 3}**
D'où **BF= {P, K, L, B, C, X}** et **R2** devient vide

7ème cycle

R2 = {1, 8} **BF= {D, E, K, L, B, C, X}** **R2 = { 8}**

8ème cycle

R2 = {4, 5} **BF= {Y, E, K, L, B, C, X}** **R2 = { 5}**

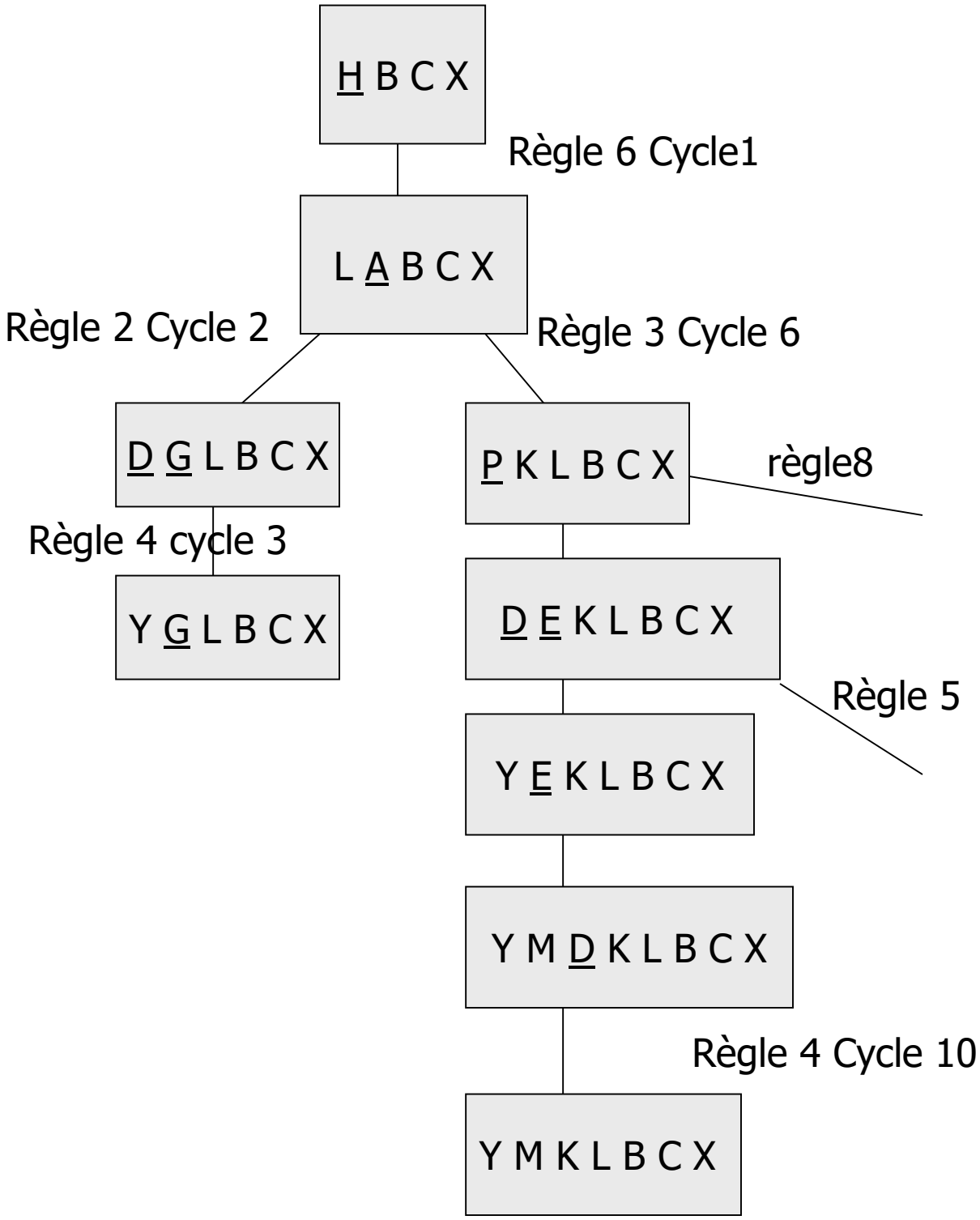
9ème cycle

R2 = {5} **BF= {Y, M, D, K, L, B, C, X}** **R2 = { }**

10ème cycle

R2 = {4} **BF= {Y, M, K, L, B, C, X}** **R2 = { }**

Il n'y a plus d'hypothèse à vérifier : il s'arrête


H B C X
Règle 6 Cycle1
L A B C X
Règle 2 Cycle 2
Règle 3 Cycle 6
D G L B C X
P K L B C X
règle8
Règle 4 cycle 3
Y G L B C X
D E K L B C X
Règle 5
Y E K L B C X
Y M D K L B C X
Règle 4 Cycle 10
Y M K L B C X

Les moteurs d'inférences

Tous les moteurs d'inférences travaille en cherchant une solution à un problème posé.

Cette recherche peut être représenté par une arborescence:

la racine base de fait
les différentes branches l'application d'une règle particulière

La recherche d'une solution est conditionnée par les caractéristiques du moteur:

- Le formalisme
- Le contrôle
- La stratégie d'exploration
- Le type de raisonnement

Les moteurs d'inférences

Le formalisme:

- La logique des propositions
  (moteur d'ordre 0)
- La logique des prédicats
  (moteur d'ordre 1)

Contrôle:

- Le chaînage
  Sens de la recherche
  Chaînage avant (dirigé par les données)(Forward chaining)
  Chaînage arrière (dirigé par les buts) (backward chaining
- Le retour arrière (backtracking)
  Retour arrière
  Régime irrévocable
- Exploration
  Stratégie du moteur
  en largeur, en profondeur, par heuristiques
- Raisonnement
  Moteur monotone (Un fait initial ou établie ne peut plus être nié).

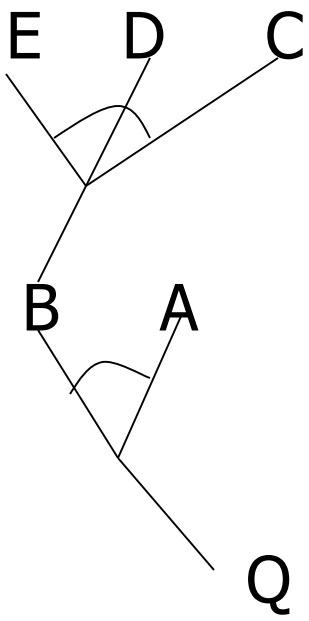
Les moteurs d'inférences

Exemples de moteurs particuliers

Chaînage avant

SCHEMAVANT1

- Contrôle irrévocable
- Recherche en profondeur
- Raisonnement monotone
- Restriction : Chaque règle est déclenchée au plus une fois



Graphe ET-OU établissant Q, lorsque A,C,D,E sont établis

Base de règles

K,L,M $\rightarrow$I
I,L,J $\rightarrow$ Q
C,D,E $\rightarrow$ B
A,B $\rightarrow$ Q
L,N,O,P $\rightarrow$ Q
C,H $\rightarrow$ R
R,J,M $\rightarrow$ S
F,H $\rightarrow$ B
G $\rightarrow$ F

Base de faits

A,C,D,E,G,H,K

Si K et L et M sont établis alors on conclut que I est établi

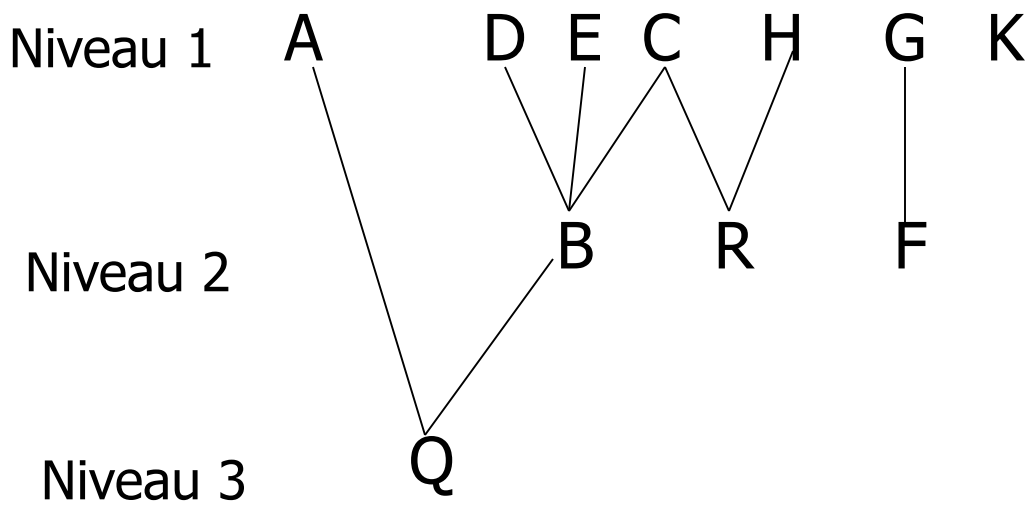
Les moteurs d'inférences

SCHEMAVANT 2

Production et intégration des faits en « largeur d'abord »

Soit un état E de la base de faits:

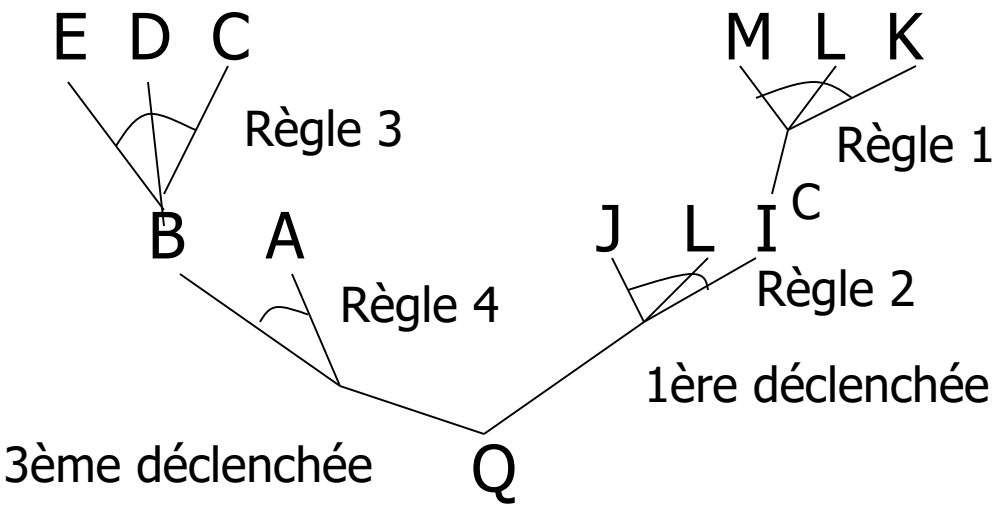
• Le moteur cherche à déclencher l'une après l'autre, **toutes** les règles compatible avec E, **avant** d'adjoindre leurs conclusion à E et de les utiliser pour déclencher de nouvelles règles.



Les moteurs d'inférences

Chaînage arrière

- Production des sous problèmes en profondeur d'abord
- Compare la partie conclusion des règles aux faits à établir du moment



Compilation de la base de connaissance

Problème

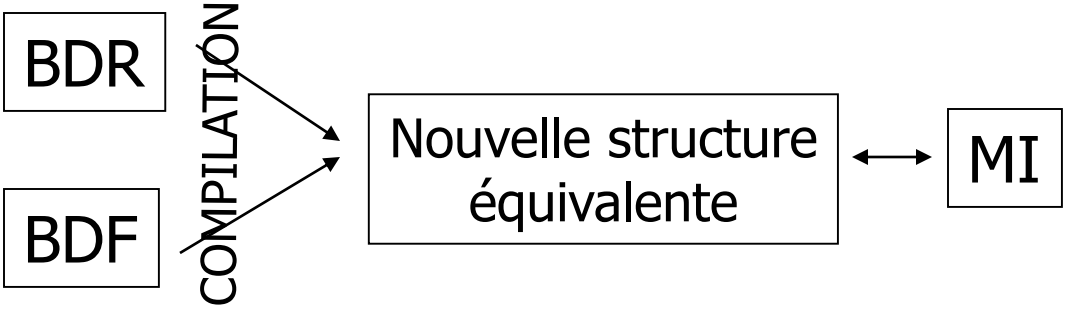
Tous les moteurs d'inférences
- l'accès associatif
- Non déterminisme

Complexité des algorithmes

Solution:

- Compilation

Transformation de la base de connaissance en une structure plus organisée et plus simple



Le langage LISP

1. Introduction

LISP le langage de l'intelligence artificielle

- 95% d'applications IA en lisp
- Outil de développement rapide
- J. Mc Carthy au MIT

Plusieurs dialectes

Common Lisp
Lelisp

2. Eléments de base

2.1. Objet de base

L'atome :

- Un nombre 1.2, 1.3, 20, etc..
- Un symbole a, poids, age,etc..
- Un caractère #\a, #\b, etc…
- une chaîne de caractères " tu apprends "

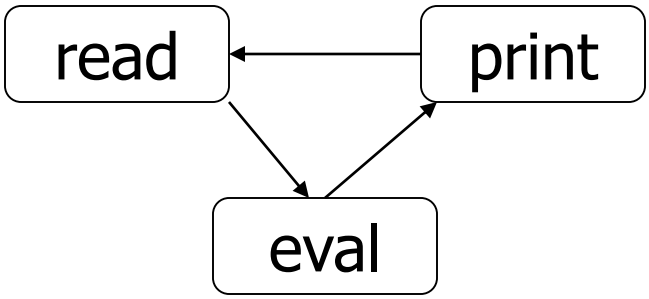
LISP

2.2 Structure de Base

- Lisp provient de LISt Processing (traitement de liste)
- Lisp tourne autour des listes
- Une liste est définie par:
  - une parenthèse ouvrante
  - un ensemble d'éléments séparés par des blancs
  - une parenthèse fermante
  - Exemple
    - (avion bateau auto) ou (1 a 8)
    - (1 (a b c))

2.3. Fonctionnement de l'interpréteur

Lisp est interpréteur qui fonctionne selon la boucle suivante



Toute information entrée sera envoyée à l'évaluateur qui va retourner une valeur si l'information peut être évaluée ou une erreur dans le cas contraire

Les informations envoyées à l'évaluateur sont de trois types (S-expression):

- Constante
  - Retourne cette constante
- Symbole
  - Retourne la valeur du symbole
- Liste
  - Son premier élément une fonction
  - Les autres éléments des arguments de la fonction

Lisp

Remarques

- Il existe une fonction qui permet d'éviter l'évaluation : quote
  (quote(a b c)) ou ` (a b c)
  Retourne (a b c)

- Si une fonction f1 a pour argument une autre fonction f2, le résultat de l'évaluation de f1 est directement envoyé à f2

- Deux symboles prédéfinis
  - le t « true »: il sert à retourner une valeur non nulle
  - Le nil ou liste ()

Lisp

3 Fonctions de base

3.1 Fonctions arithmétiques

L'addition:

```
? (+ 3 6)
? 9
```

La multiplication

```
? (* 3 6)
?18
```

La soustraction – et la division /

```
? (+ (* 2 5) (- 7 6)))
? 11
```

3.2 Les fonctions des comparaison

```
? (= 7 6)
? Nil
? (> 7 6)
? t
```

Lisp

3.3 Traitement de listes

- La fonction de construction le **cons**
  - E lle permet d'ajouter un élément à la tête de liste

    (cons 1 ()) retourne (1)

    (cons 8 '(a b c)) retourne (8 a b c)

    (cons '(a b) '(c d)) retourne ((a b) c d)
- La fonction **list**

  (list ' a ' b 'c) retourne (a b c)

  (list '(a b c) d 15) retourne ((a b c) d 15)

Les fonction de décomposition

- La fonction car

  Permet d'accéder au premier élément de la liste

  (car '(a b c)) retourne a
- La fonction cdr

  Retourne tous les éléments d'une liste sauf le premier

  (cdr '(a b c)) retourne (b c)

car et cdr n'affecte pas la liste initiale

lisp

3.4 l'affectation

La fonction setq affecte une valeur à un symbole:
(setq a 18) affecte la valeur 18 à a
(setq animal '(carnivore herbivore))

3.5 Contrôle

La fonction if comporte trois arguments:

- Le premier est une condition
- Le deuxième est une expression qui est évalué si la condition ne retourne pas nil
- La troisième dans le cas contraire

```
(if (= x 0) 'erreur (/ 1 x))
```

La fonction cond possède plusieurs arguments

```
(cond ((= x 0) 0)
      ((< x 0) 1)
      (t  –1))
```

3.6 Prédicats prédéfinis

- De comparaison equal
  (equal l ())
- (null l)
- Symbolp (l) vrai si l est un symbole
- Atom (l) vrai si l est un atome
- Listp(l) vrai si l est une liste
- Numberp vrai si l est un nombre

4 définition des fonctions

La programmation en lisp consiste à ajouter des fonctions à celles déjà existantes il se fait par la fonction defun :

```
(defun signe (x)
          (cond    ((= x 0) 0)
                   ((< x 0) 1)
                   (t  –1)))
(defun surfdisque(r)
                   (* 3.14 (* r r)))
```

BIBLIOGRAPHIE

• **J.L. LAURIERRE**, Intelligence artificielle, résolution de problèmes par l'Homme et la machine, Eyrolles 1986

• **H.Farreny, M.Ghallab**, Elément d'intelligence Artificielle, Hermés 1987

• **P.H. Winston**, Artificial Intelligence, Addison – Wesley 2 ème édition 1984