



Fondement de l'IA

Cours: Intélligence Artificielle

2ème année informatique de gestion

Enseignante: Dr. Saoussen Mathlouthi

Saoussen.mathlouthi@ensi-uma.tn

2021-2022

Séance 1: 30/09/2021

Objectifs du cours

Fondement de l'IA

2021-2022



1. Présenter les fondements de l'intelligence artificielle et de la robotique
2. Apprendre les techniques classique d'intelligence artificielle
3. Poser une démarche de résolution de problème
4. Implémenter les algorithmes classiques de l'intelligence Artificielle
5. Intégrer d'une façon pratique des techniques d'IA

Plan du cours



Fondement de l'IA

2021-2022

Volume horaire : 15h de cours et 7,5h de TD/TP

Chapitre 1: Introduction à l'IA

(3h cours)

Chapitre 2: Représentation des connaissances

(3h cours / 1,5h TD)

Chapitre 3: Recherche dans un espace d'état

(3h cours / 1,5h TD / 1,5h TP)

Chapitre 4: Jeux à deux adversaires

(3h cours / 1,5h TP)

Chapitre 5: Satisfaction de contraintes

(3h cours / 1,5h TD)



Chapitre 1: Introduction à l'IA

Plan



Qu'est-ce que l'intelligence artificielle ?

Objectifs de l'IA ?

Exemples d'application de l'IA

L'IA forte et l'IA faible

Qu'est-ce que l'intelligence artificielle ?

Fondement de l'IA

2021-2022



L'intelligence artificielle (terme créé par John McCarthy), souvent abrégée avec le sigle **IA**

L'IA est définie par l'un de ses créateurs, Marvin Lee Minsky, comme « *la construction de programmes informatiques qui s'adonnent à des tâches qui sont, pour l'instant, accomplies de façon plus satisfaisante par des êtres humains car elles demandent des processus mentaux de haut niveau tels que : l'apprentissage perceptuel, l'organisation de la mémoire et le raisonnement critique* ».

Qu'est-ce que l'intelligence artificielle ?

Fondement de l'IA

2021-2022



- **L'intelligence**: la capacité d'acquérir et d'appliquer des connaissances et des compétences
 - **Artificielle**: fabriqué ou produit par des êtres humains plutôt que de se produire naturellement
-
- L'intelligence artificielle (IA) est un processus d'imitation de l'intelligence humaine qui repose sur la création et l'application d'algorithmes exécutés dans un environnement informatique dynamique. Son but est de permettre à des ordinateurs de **penser** et **d'agir** comme des êtres humains.

Qu'est-ce que l'IA?



L'intelligence artificielle est une discipline qui systématisé et automatise les tâches intellectuelles pour:

- **Penser comme un humain**

[The automation of] activities that we associate with human thinking, activities such as decision making, problem solving, learning – Bellman 1978

- **Agir comme un humain**

The art of creating machines that perform functions that require intelligence when performed by people – Kurzweil 1990

- **Penser raisonnable (de manière rationnelle)**

The study of the computations that make it possible to perceive, reason, and act – Winston 1992

- **Agir raisonnablement (de manière rationnelle)**

AI... is concerned with intelligent behaviour in artifacts – Nilsson 1998

Penser comme un être humain

Fondement de l'IA

2021-2022



- Un programme pense-t-il comme un humain?
- Déterminer comment pense les êtres humains?
 - L'introspection: Essayer de se saisir de ces propres pensées.
 - Les expériences psychologique: Observer une autre personne.
 - Observer le fonctionnement du cerveau.
- L'approche cognitive: la description, l'explication, et le cas échéant la simulation, des mécanismes de la pensée humaine, animale ou artificielle.

Agir comme un être humain



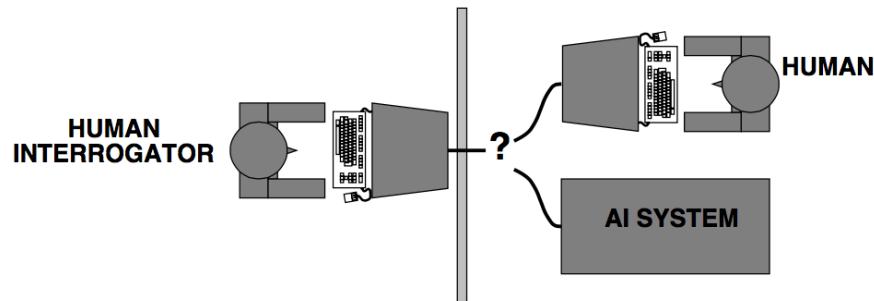
Fondement de l'IA

2021-2022

- Alan Turing (1950) "Computing machinery and intelligence" :
 - "Les machines peuvent-elles penser?"
 - "Les machines peuvent-elles se comporter intelligemment ?"
- Principe du Test de Turing: Le test de Turing a pour vocation de **mesurer la capacité de l'Intelligence Artificielle à être confondue avec l'intelligence humaine.**

Pour répondre à la question "une machine peut-elle penser ?", ce test se déroule de la manière suivante : un humain va dialoguer avec deux interlocuteurs tout en sachant que l'un des deux est une machine. Après 5 minutes de conversation, l'homme doit réussir à déterminer lequel de ses deux interlocuteurs est l'IA. Dans ce jeu des imitations, l'objectif de la machine n'est pas de répondre correctement aux questions, mais de tromper l'interlocuteur humain en répondant de la manière la plus humaine possible.

Test opérationnel pour établir un comportement intelligent : **l'imitation**



Penser rationnellement



Fondement de l'IA

2021-2022

- ❑ Quels sont les arguments et les processus de pensée corrects?
- ❑ Aristote a essayé de codifier le « *bien penser* ».
 - Le **syllogisme**: raisonnement logique augmentatif à deux propositions conduisant à une conclusion vraie
 - Ex: Tous les hommes sont mortels + Socrate est un homme donc Socrate est mortel

Agir rationnellement



Fondement de l'IA

2021-2022

- Comportement rationnel : faire la chose adéquates
 - **La chose adéquate** : celle qui, étant données les informations disponibles, doit permettre d'atteindre au mieux l'objectif
 - Aristote : Tout art et toute investigation, et pareillement toute action et sa suite, est pensé pour **viser** quelque chose de bon

Objectifs de l'IA ?



- Construire des machines pour une large variété d'applications augmentant les capacités de résolution de problèmes « mal posés »
- Formaliser ce qu'est la **connaissance**
- Mécaniser **l'intelligence**
- Utiliser des modèles computationnels pour **comprendre** des **comportements complexes**
- Rendre **l'interaction** avec les systèmes computationnels aussi simple qu'avec les humains

Exemples d'application de l'IA

2021-2022



Fondement de l'IA

Exemple: jeux



Exemple: Robotique



→ *Prise de décision*

Si un ordinateur peut battre le champion du monde d'échecs. Il n'a pas conscience de jouer aux échecs. Du chemin reste à faire sur la compréhension du raisonnement humain.

- Perception
- Prise de décision
- Action
- Communication
- Apprentissage



- **Intelligence artificielle faible**

La notion d'intelligence artificielle faible constitue une approche pragmatique d'ingénieur : chercher à **construire des systèmes de plus en plus autonomes**, des algorithmes capables de résoudre des problèmes **en simulant l'intelligence**

- **Intelligence artificielle forte**

Le concept d'intelligence artificielle forte fait référence à une machine capable non seulement de **produire un comportement intelligent**, mais d'éprouver une impression d'une **réelle conscience de soi**, de « **vrais sentiments** », et une **compréhension de ses propres raisonnements**

Merci pour votre attention





Fondement de l'IA

Cours: Intélligence Artificielle

2ème année informatique de gestion

Enseignante: Dr. Saoussen Mathlouthi

2021-2022

Séance 2: 07/10/2021

Plan



Comparaison machines vs. humains

Les domaines de l'IA

Agent intelligent

Modèle PEAS

Caractéristiques d'environnement

Types des agents

Comparaison machines vs. humains

Fondement de l'IA

2021-2022



- **Ce que les machines font mieux que la plupart des humains**
 - Jouer aux échecs, aux dames, à Othello, au Backgammon
 - Faire des mots-croisés
 - Prouver des théorèmes
 - Trier le courrier
 - Déetecter des fraudes
 - Planifier, organiser les vols par avion
- **Ce que les machines ne savent pas (encore) faire**
 - Reconnaître une voix, un visage
 - Comprendre une langue
 - Composer de la musique, de l'art
 - Naviguer de façon complètement autonome
 - Avoir du bon sens : « Combien de jambes a un poisson ? »

Que sait-on faire avec l'I.A. ?



Fondement de l'IA

2021-2022

Classez les problèmes ci-dessous selon qu'ils sont :

A : résolus

B : bientôt résolus

C : pas prêt de l'être

Conduire une voiture dans un virage simple

Conduire une voiture dans Paris, vers 18h30

Piloter un char d'assaut

Découvrir et démontrer des théorèmes

Composer de la musique

Faire la vaisselle

Discuter avec une personne pendant une heure

Traduire de l'anglais parlé en Norvégien en temps réel

Écrire une histoire drôle

Jouer à la bourse comme des traders professionnels 112

Les domaines de l'IA



Fondement de l'IA

2021-2022

- Jeux (Echecs, Quake, Wow, Stratcraft ...)
- Planification
- Système à base de connaissances
- Traduction automatique
- Diagnostique médical
- Navigation autonome (avions, drones, robots, voitures...)
- Fouille de données
- Reconnaissance de formes
- Identification vocale ou visuelle
- Optimisation de processus industriels
- Interfaces intelligentes
- ...

Quelques succès de l'IA



Fondement de l'IA

2021-2022

- Remote Agent (NASA) : programme de planification autonome embarqué (2000)
- Deep Blue : ordinateur d'IBM qui a battu le champion du monde d'échecs Kasparov (1997)
- ALVINN : système de vision informatique à basé sur un réseau de neurones entraîné pour piloter une voiture (Dean Pomerleau, 1989-1992)
- HipNav : système d'assistance chirurgicale pour la pose de prothèses de hanche (DiGioia et al., 1996)
- PROVERB : programme qui résout des problèmes de mots croisés mieux que la majorité des humains (Littman et al., 1999)

Autres succès de l'IA

Fondement de l'IA

2021-2022



- ViaVoice [Reconnaissance vocale]
- Codage postal [Reconnaissance d'écriture]
- Waston [Système expert]
- Aibi et Asimo [Robots de compagnie]
- Robots assistants [Robots médicaux]
- Bigdog [Robots militaires]
- Google Driveless Car [Véhicule autonome]

Intelligence du raisonnement

Fondement de l'IA

2021-2022



« Tout problème pour lequel il n'existe pas d'algorithme connu, ou de coût raisonnable, relève de l'I.A. »

L'I.A. doit permettre de proposer des solutions logicielles permettant aux programmes de **raisonner logiquement**.

Définition basée sur des formalismes logiques

- Démonstration automatique de théorèmes
- Utilisation de **règles précises** d'inférence
- Qualités issues des **mathématiques** (preuves, explications, ...)

Agent intelligent

2021-2022



- Agent rationnel:

- Opérer sous le contrôle d'une instance autonome
- Percevoir l'environnement
- Persister pendant une période prolongée
- S'adapter au changement
- Être capable de partager les objectifs d'un autre agent

Un agent est une entité qui perçoit et qui agit.

- De manière abstraite, un agent est une fonction qui fait correspondre à un historique de **perceptions** un ensemble **d'actions**.
- Le **processus agent** f prend en entrée une séquence **d'observations** (percepts) et retourne une **action**

$$f: P^* \rightarrow A$$

- Pour une classe donnée d'environnements et de tâches, nous cherchons l'agent (ou classe d'agents) avec la meilleure performance

Agent intelligent

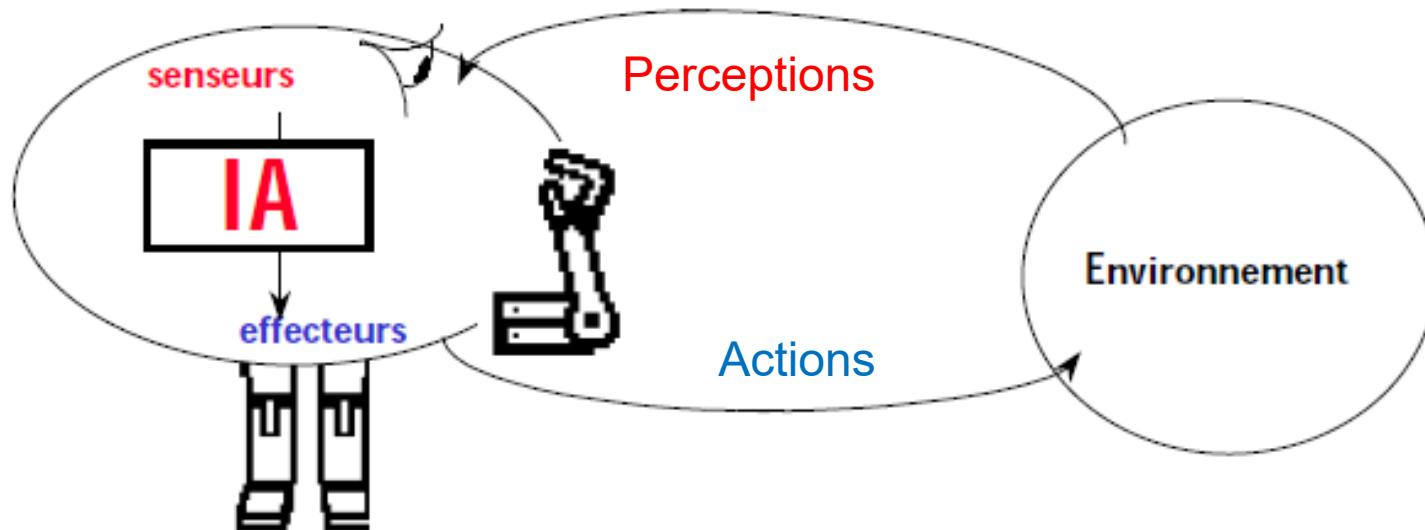


Fondement de l'IA

2021-2022

Capacités fondamentales :

- perception
- représentation des connaissances (modélisation)
- apprentissage
- raisonnement
- prise de décisions



Architecture d'un agent intelligent



Définition:

Un agent est n'importe quelle entité qui perçoit son environnement par des **capteurs (sensors)** et qui agit sur cet environnement par des **actionneurs (actuators)**

- Un agent humain a :
 - des yeux, des oreilles ...etc.
 - des mains, des jambes, une bouche ...etc
- Un agent robot a :
 - des caméras, des capteurs infra rouges et autres capteurs
 - des roues, des jambes, des bras-articulés, et d'autres actionneurs
- Un agent logiciel a :
 - un clavier, un accès lecture à un disque dur ...etc.
 - un écran, un accès écriture à un disque dur ...etc.

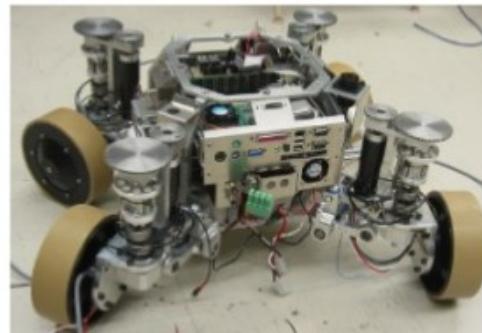
Agent intelligent



Fondement de l'IA

2021-2022

Exemples d'agents intelligents:



- Systèmes d'aide à la décision
- Robots
- IA dans les jeux

Agent intelligent

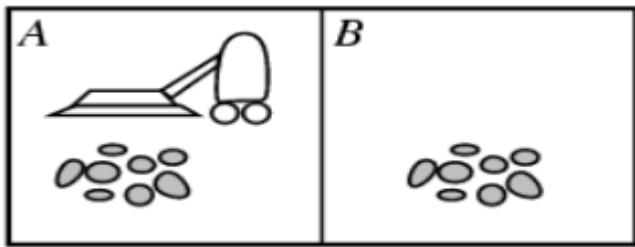


Fondement de l'IA

2021-2022

Exemples d'agents intelligents:

Aspirateur robotisé



- Observations (données sensorielles) : position et état des lieux.

Par exemple : [A; Propre]; [A; Sale]; [B; Propre]; [B; Sale]

- Actions : Droite; Gauche; Aspirer; NoOp

- *f* :

[A; Propre] ! Droite
[A; Sale] ! Aspirer



Pseudo-code de l'agent intelligent:

```
function AGENT(percept) returns action  
static : memory  
    memory  $\leftarrow$  UPDATE-MEMORY(memory,percept)  
    action  $\leftarrow$  CHOOSE-BEST-ACTION(memory)  
    memory  $\leftarrow$  UPDATE-MEMORY(memory,action)  
return action
```

Agent intelligent

2021-2022



- Un agent rationnel doit agir «correctement» en fonction de ce qu'il perçoit et de ses capacités d'action:
 - L'**action correcte** est celle permettant à l'agent de réussir le mieux
 - Besoin d'une **mesure de performance**
- **Mesure de performance**: une fonction objective mesurant la qualité d'un comportement de l'agent
- Étant donné une séquence d'observations (données sensorielles) et des connaissances propres, un agent rationnel devrait **choisir une action qui maximise la mesure de performance**



- **Un agent rationnel ne veut pas dire un agent «qui sait tout»**
(connaître tous les effets de ses actions), **ne veut pas dire un agent «parfait»**
 - La rationalité **maximise la performance escomptée** et non pas la performance réelle
 - Souvent on ne peut pas connaître la performance réelle avant l'action
- Un agent peut effectuer des **actions d'observation** pour cueillir des informations nécessaires à sa tâche.
- Un agent est **autonome** s'il est capable d'adapter son comportement aux changements dans **l'environnement** (capable d'apprendre, de planifier, de raisonner).



- PEAS : un modèle de conception des agents par la spécification des composantes majeures suivantes :
 - mesure de performance (**Performance**)
 - éléments de l'environnement (**Environnement**)
 - les actions que l'agent peut effectuer (**Actionneurs ou Actuators**)
 - la séquence des observations ou percepts de l'agent (**Capteurs ou Sensors**)
- PEAS : **Performance, Environment, Actuators, Sensors**

Modèle PEAS

Fondement de l'IA

2021-2022



Modèle PEAS pour un robot taxi

Agent :

Mesure de performance :

Environnement :

Actionneurs :

Senseurs :



Modèle PEAS pour un robot taxi

Agent : robot taxi

Mesure de performance : sécurité, vitesse, respect du code routier, voyage confortable, maximisation des profits

Environnement : route, trafic, piétons, clients

Actionneurs : volant, changement de vitesse, accélérateur, frein, clignotants, klaxon

Senseurs : Caméras, sonar, compteur de vitesse, GPS, témoins de moteurs, etc.

Modèle PEAS

Fondement de l'IA

2021-2022



Modèle PEAS pour un diagnostic médical automatisé

Agent :

Mesure de performance :

Environnement :

Actionneurs :

Senseurs :



Modèle PEAS pour un diagnostic médical automatisé

Agent : système de diagnostic médical

Mesure de performance : santé des patients, minimisation des coûts, satisfaction des patients

Environnement : patients, hôpital, personnel soignant

Actionneurs : moniteur pour afficher des questions, les résultats de tests ou de diagnostic, le traitement, etc.

Senseurs : clavier et souris pour saisir les symptômes, les réponses aux questions, etc.

Modèle PEAS



Fondement de l'IA

2021-2022

Modèle PEAS pour Pacman

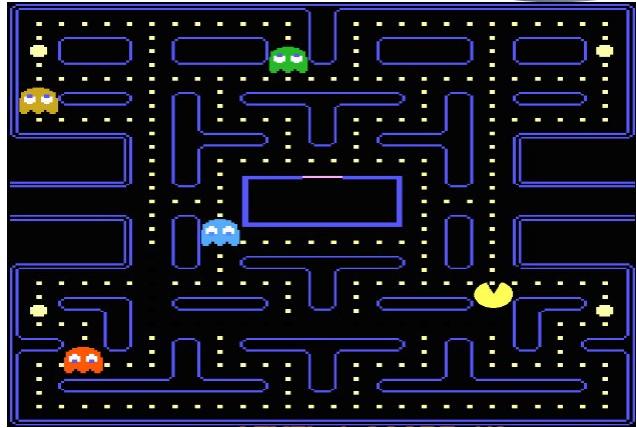
Agent :

Mesure de performance :

Environnement :

Actionneurs :

Senseurs :





Modèle PEAS pour Pacman

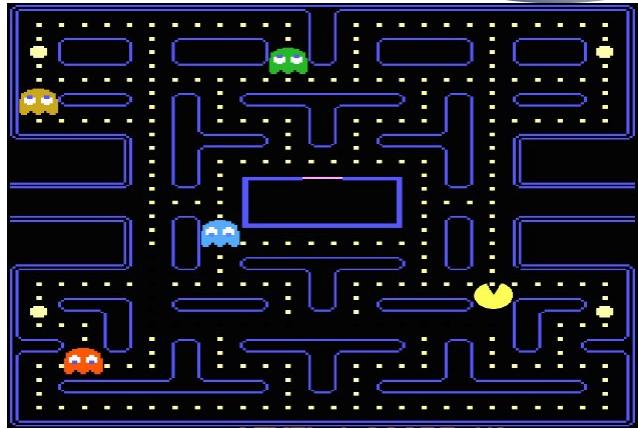
Agent: Pacman

Mesure de performance: score

Environnement: le labyrinthe, les pommes, les fantômes

Actionneurs: se déplacer, manger, crier

Senseurs: senseur de fantômes, senseur de pommes, senseur pour la position



Caractéristiques d'environnement



- Chaque problème aura un environnement avec des caractéristiques différentes
- **Caractéristiques des environnements:**
 - **Complètement observable** (vs. partiellement observable)
 - **Déterministe**(vs. stochastique)
 - **Épisodique**(vs. séquentiel)
 - **Statique**(vs. dynamique)
 - **Discret**(vs. continu)
 - **Agent unique** (vs. multi-agent)
- On identifie souvent les caractéristiques d'environnement en réfléchissant à comment on programmerait/simulerait cet environnement.



Complètement observable (vs. partiellement observable):

grâce à ses capteurs, l'agent a accès à l'état complet ou partiel de l'environnement à chaque instant

- Le jeu des échecs est complètement observable: on voit la position de toutes les pièces
- Le jeu du poker est partiellement observable: on ne connaît pas les cartes dans les mains de l'adversaire



Déterministe (vs. stochastique):

l'état suivant de l'environnement est entièrement déterminé par l'état courant et l'action effectuée par l'agent

- Le jeu des échecs est déterministe: déplacer une pièce donne toujours le même résultat
- Le jeu du poker est stochastique: la distribution des cartes est aléatoire

N.B.: on considère comme stochastique les phénomènes qui ne peuvent pas être prédits parfaitement.



Épisodique (vs. séquentiel):

- ❖ Les opérations/comportements de l'agent sont divisés en épisodes.
- ❖ Chaque épisode consiste à observer l'environnement et effectuer une seule action
- ❖ Cette action n'a pas d'influence sur l'environnement dans l'épisode suivant
- La reconnaissance de caractères est épisodique: la prédiction du système n'influence pas le prochain caractère à reconnaître
- Le jeu du poker est séquentiel: décider si je mise ou pas a un impact sur l'état suivant de la partie



Statique(vs. dynamique):

l'environnement ne change pas lorsque le ou les agents n'agissent pas

- Le jeu des échecs est statique: l'état du jeu ne change pas si personne ne joue
- Le jeu de stratégie en temps réel (comme StarCraft) est dynamique: les unités ont une certaine autonomie; elles peuvent évoluer même si aucun joueur ne fait une action.



Discret(vs. continu):

un nombre limité et clairement distincts de **données sensorielles et d'actions**

- Le jeu des échecs est dans un environnement discret: toutes les actions et état du jeu peuvent être énumérées
- La conduite automatique d'une voiture est dans un environnement continu: l'angle du volet est un nombre réel



Agent unique (vs. multi-agent):

un agent opérant seul dans un environnement

- Résoudre un Sudoku est à agent unique: aucun adversaire
- Le jeu des échecs est multi-agent: il y a toujours un adversaire

Types des agents



4 types d'agents:

- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents

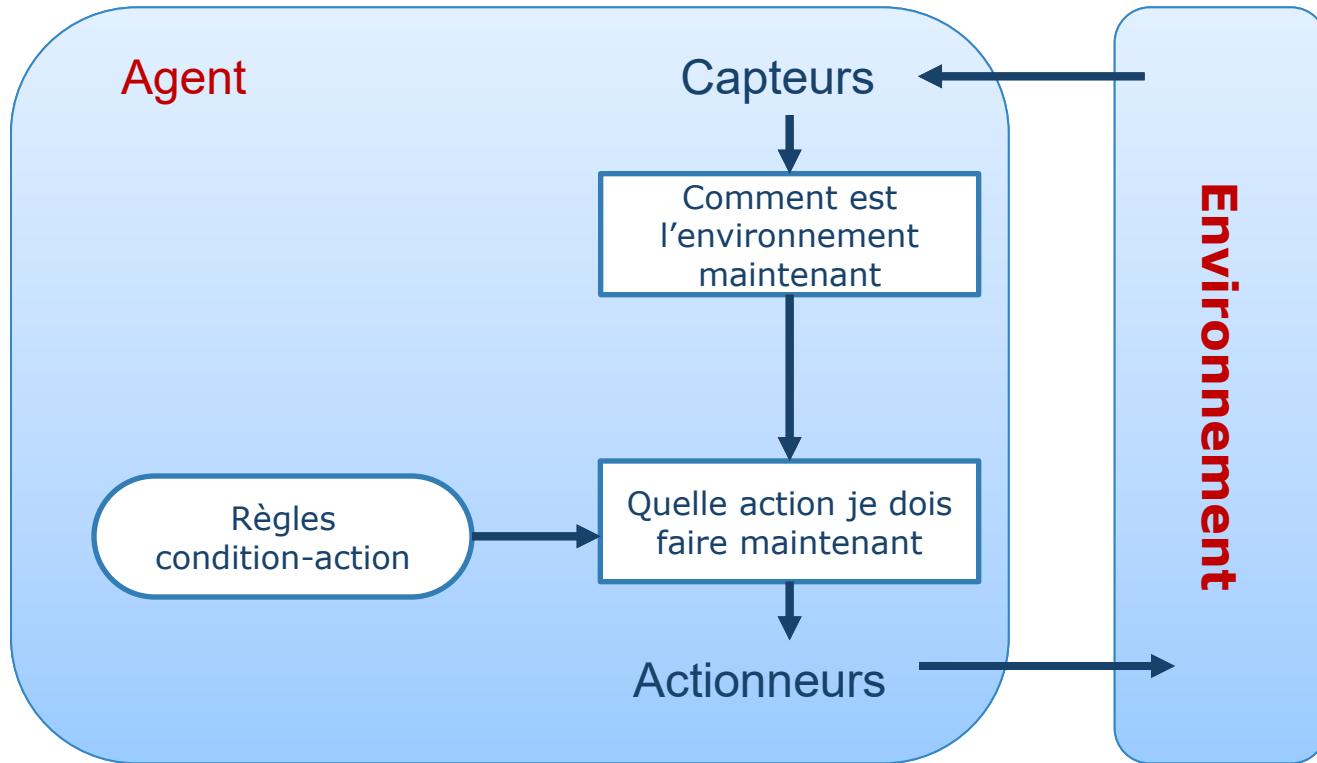
Les types d'agents varient:

- dans la façon de prendre leur décision
- à partir de quelles connaissances prendre ces décisions ?
- Dans la façons de faire l'apprentissage

Types des agents



□ Simple reflex agents

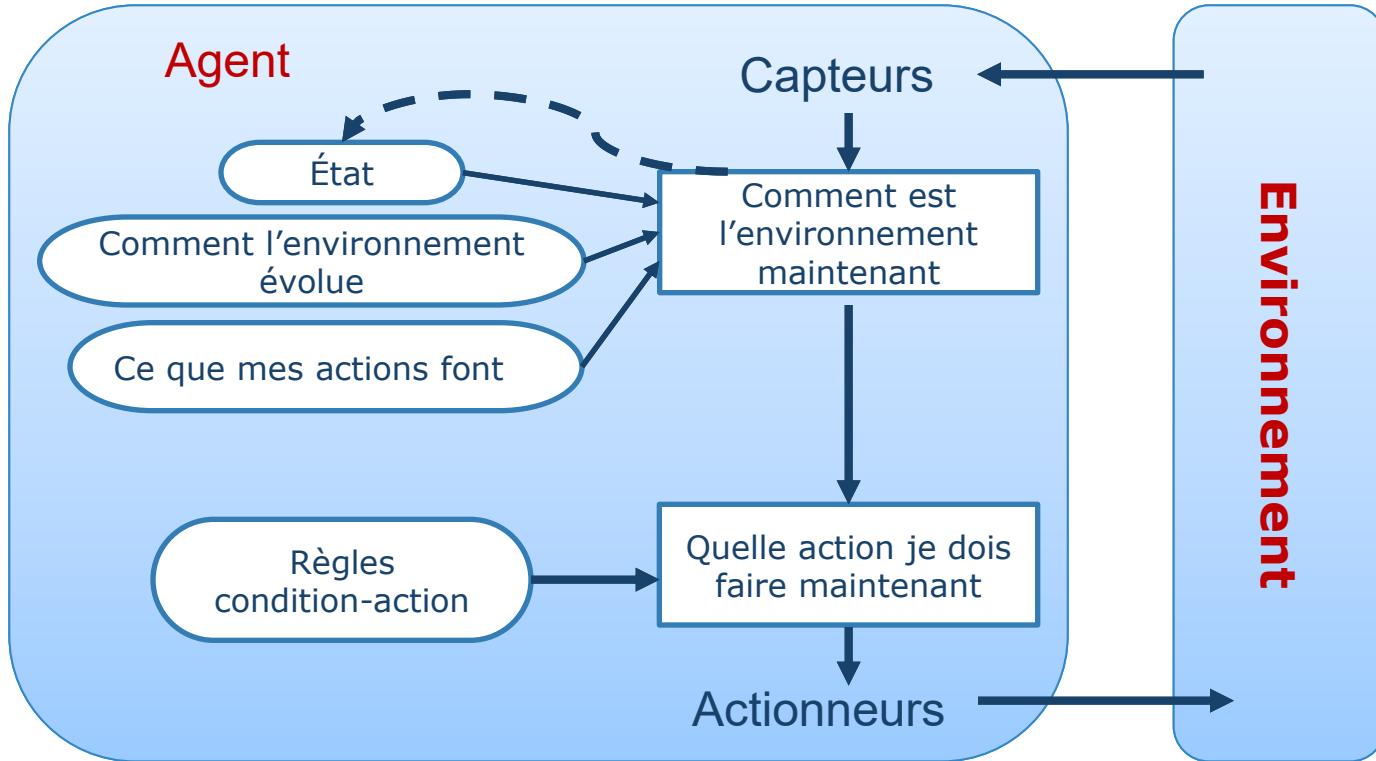


Agit seulement à partir de la perception actuelle en ignorant l'historique

Types des agents



□ Model-based reflex agents



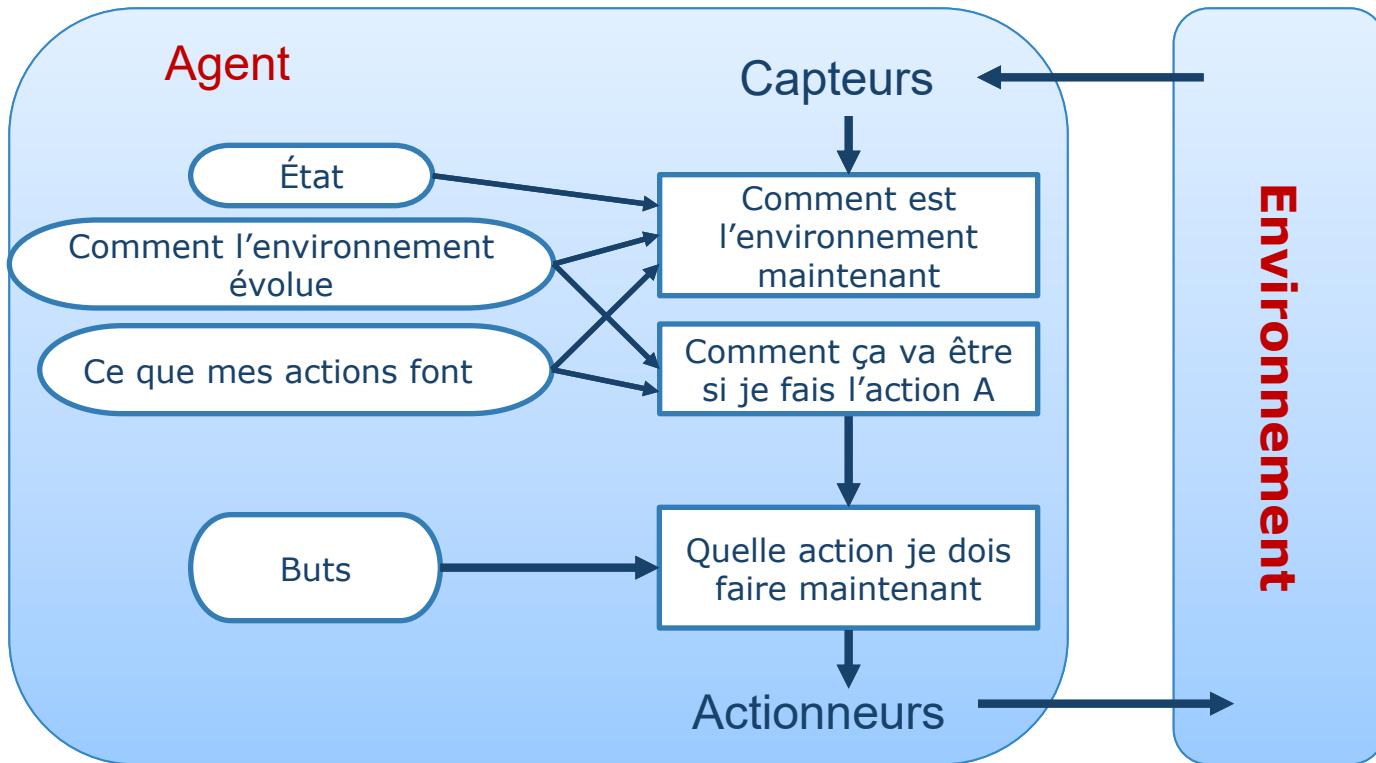
Accumule l'information dans le temps pour estimer l'état de l'environnement

Types des agents



2021-2022

□ Goal-based agents

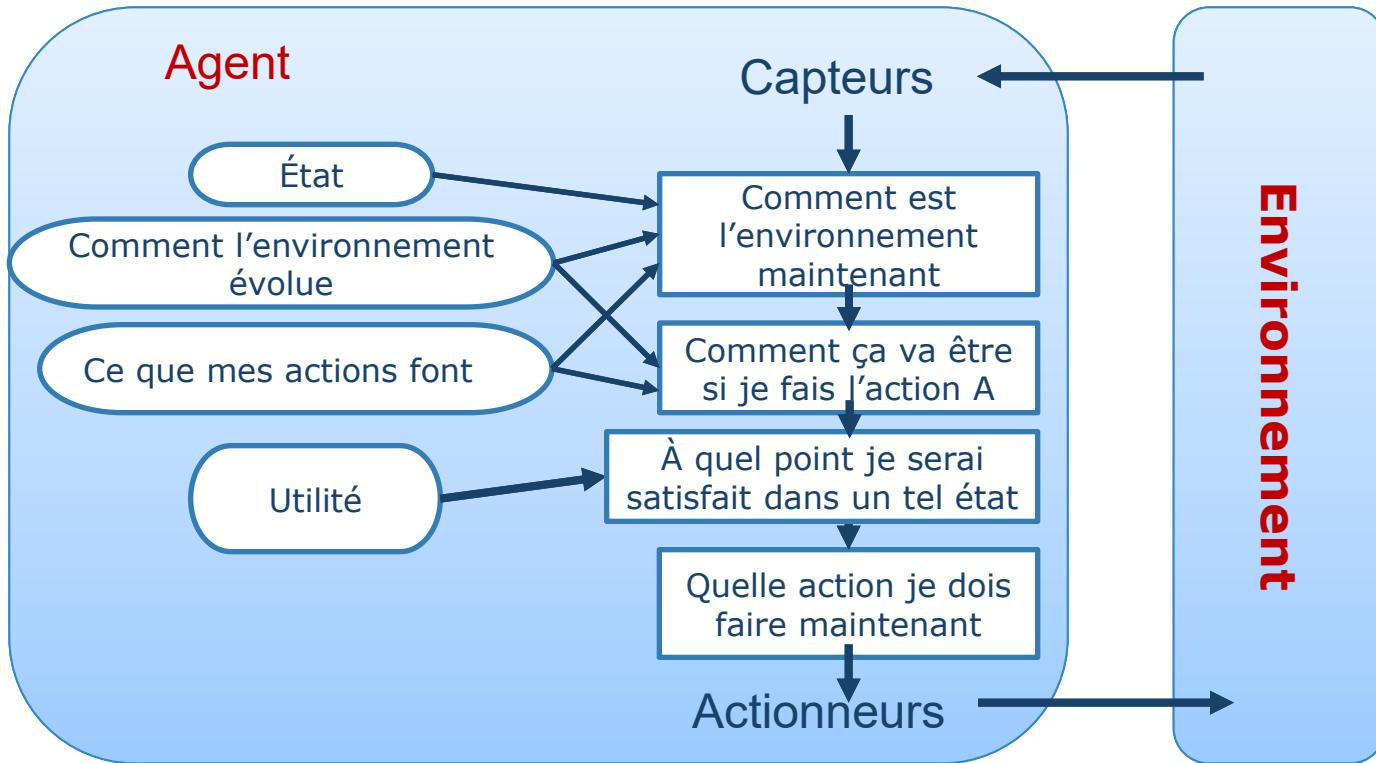


Plutôt que de spécifier une règle conditions/actions explicitement, on ne fait que spécifier un but (va pouvoir tenir compte du futur)

Types des agents



□ Utility-based agents



Intègre la notion de préférence entre les différentes actions.

Par exemple : choisir l'action qui résout une tâche donnée le plus rapidement possible

Résumé

2021-2022



Fondement de l'IA

- ❖ L'intelligence artificielle s'intéresse à tout sujet qui permettrait de reproduire toute capacité de l'intelligence humaine
- ❖ Un agent est une entité qui perçoit et agit sur son environnement
- ❖ Un agent rationnel est un agent qui cherche à maximiser sa performance espérée (moyenne)
- ❖ L'espace des agents possibles est très large
 - dépend de la tâche à résoudre
 - chaque algorithme qu'on va voir est associé à un type d'agent spécifique
- ❖ Il existe plusieurs types d'environnement:
 - leurs caractéristiques vont déterminer quel algorithme on devrait utiliser

Merci pour votre attention





Fondement de l'IA

Cours: Intélligence Artificielle

2ème année informatique de gestion

Enseignante: Dr. Saoussen Mathlouthi

Saoussen.mathlouthi@ensi-uma.tn

2021-2022

Séance 3: 07/10/2021

Séance 4: 14/10/2021



Chapitre 2:

Représentation des connaissances

Plan



Représentation des connaissances

Réseaux sémantiques

Logique des propositions

Logique des prédictats

Exercice d'application

Représentation des connaissances



- Dans le but de résoudre des problèmes complexes qui relèvent de l'intelligence artificielle, il faut un bon bagage de **connaissances** et des **outils de manipulation de ces connaissances**.
- Les connaissances concernent des **faits**, considérés vrais dans un certain monde.
- Pour représenter ces **faits** on a recours à un **formalisme** ou mode de **représentation**.
- Au niveau des **faits**, on traitera des **objets** et des **relations** qu'ils entretiennent.
- Au niveau du **formalisme**, on définira des **symboles** et des **opérations** sur ces symboles.

Représentation des connaissances

Fondement de l'IA

2021-2022



Considérons le célèbre problème du fermier, du renard, de l'oie et du sac de grains :

Un fermier sur une rive veut passer sur l'autre rive d'une rivière avec un renard, une oie et un sac de grains. Son barque ne peut contenir que 2 entités.

Contraintes: *s'il laisse le canard seul avec le renard, ce dernier va manger l'oie. S'il laisse l'oie avec le sac de grains! Plus de grains .*

Résoudre le problème du passage d'une rive à l'autre.



Représentation des connaissances

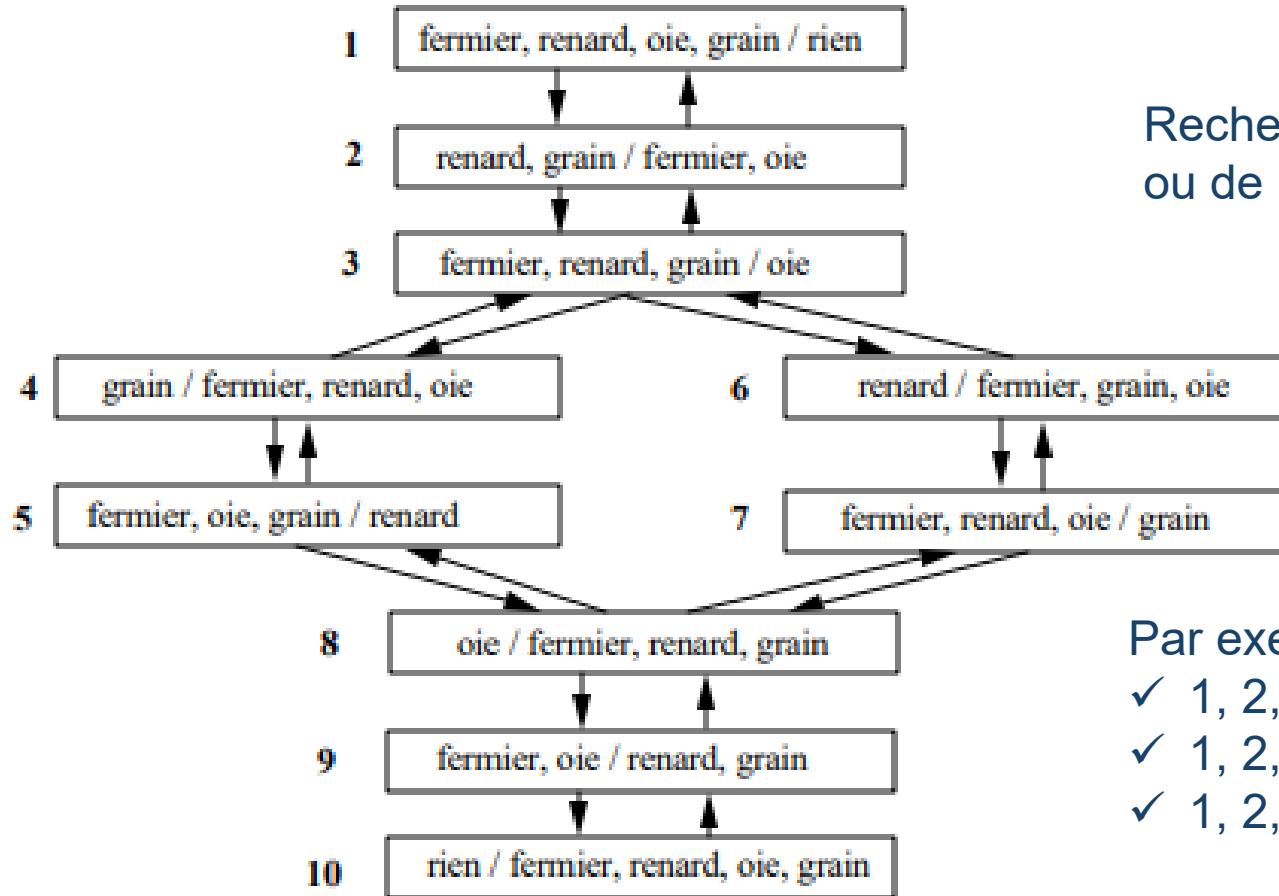


- Pour avoir une bonne perception du problème, on doit trouver une représentation simple et complète.
- Identifier les **objets** pertinents :
 - le fermier, le renard, l'oie et le sac de grains ainsi que la rivière et la barque.
- Connaître la position de chaque **objet**: à un instant donné, la situation peut être décrite par la position du fermier et de chacun de ses biens.
- Étudier les mouvements des divers objets: on parlera alors d'un **état**
- Définir un état d'une double liste : <liste rive-gauche / liste rive-droite>
- Ne considérer qu'une partie de tous les états possibles. Par exemple, l'état < fermier, renard / oie, grain > n'a pas à être considéré puisque l'oie ne doit pas être laissée seule avec le grain.
- Le changement d'un état à un autre sera représenté par une flèche

Représentation des connaissances



Le problème peut être complètement représenté au moyen du graphe suivant :



Recherche facile d'une solution ou de plusieurs solutions.

Par exemple :

- ✓ 1, 2, 3, 4, 5, 8, 9, 10
- ✓ 1, 2, 3, 6, 7, 8, 9, 10
- ✓ 1, 2, 3, 4, 5, 4, 3, 6, 7, 8, 9, 10

Représentation des connaissances

Fondement de l'IA

2021-2022



Une bonne **représentation** permet une résolution presque immédiate du problème.

Une **représentation** doit comprendre quatre parties fondamentales :

- Lexicale : Quels sont les symboles autorisés pour représenter **objets** et **relations** ?
- Structurelle : Quelles sont les **contraintes d'arrangement** de ces symboles ?
- Procédurale : Comment **créer et modifier** l'information ?
- Sémantique : Comment associer un sens aux **descriptions formelles** ?

Représentation des connaissances

Fondement de l'IA

2021-2022



Dans le problème du fermier, la représentation comprenait :

- au niveau lexical : les **nœuds** et les **flèches** pour représenter le problème ;
- au niveau structurel : une **flèche** reliait deux **nœuds** ;
- au niveau sémantique : un **nœud** était associé à chaque couple de **listes d'objets** et une **flèche** à chaque changement **d'état** ;
- au niveau procédural : des mécanismes mentaux permettent de dessiner le **graphe** en fonction de la perception du problème ainsi que de l'interprétation qui serait faite de **la représentation symbolique**.

Réseaux sémantiques

Fondement de l'IA

2021-2022



Les **réseaux sémantiques**: formalisme de représentation des connaissances

Le formalisme de réseaux sémantiques servira d'introduction à la **logique clausale**

La logique clausale offre plus de flexibilité pour la **représentation des connaissances**.

Les éléments intervenant dans le domaine des connaissances sont:

objets, fonctions, relations, prédictats



Éléments de la connaissance:

- Les **objets** peuvent être:
 - Entités concrètes comme un livre, la lune, Napoléon, ou concepts abstraits comme l'amour, la peur, la justice.
 - Objet primitif comme un ordinateur objet composé comme la double liste dans le problème du fermier.
- Une **fonction** n-aire **f** fait donc correspondre un seul objet à une liste de n objets, par exemple:
 - âge(Pierre) : l'âge de Pierre
 - père(Andrée) : le père d'Andrée
 - entier_entre(2, 4) : l'entier entre 2 et 4



Éléments de la connaissance:

- Un **prédictat** exprime une **relation** entre objets.
 - Les prédictats père, mère, homme, femme, étudiant, professeur expriment des relations évidentes.
 - On peut associer une valeur de vérité (vraie ou fausse) à une relation.
 - L'arité d'une relation est le nombre de ses arguments.

Par exemple, la relation a_des_griffes(Garfield) a une arité de 1; la relation muni(Garfield, griffes) a une arité de 2.



- **Proposition**

- La Rochelle est en Charente-Maritime (V)
- La hauteur de la Tour Eiffel est inférieure à 300 m(F)

Une proposition a une valeur de vérité V ou F

- **Connecteurs**

\wedge (ET), \vee (OU), \neg (NON), \Rightarrow (IMPLIQUE), \Leftrightarrow (EQUIVALENT)

- **Formules** (propositions connectées suivant des règles)

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

Logique des propositions (sémantique)



- **Table de vérité (interprétation des formules)**

A	B	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$\neg A$	$\neg(\neg A)$	$\neg A \vee B$
V	V						
F	V						
V	F						
F	F						

Logique des propositions (sémantique)



- **Table de vérité (interprétation des formules)**

A	B	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$\neg A$	$\neg(\neg A)$	$\neg A \vee B$
V	V	V	V	V	F	V	V
F	V	F	V	V	V	F	V
V	F	F	V	F	F	V	F
F	F	F	F	V	V	F	V

- **Équivalences** (formules ayant la même interprétation)

$\neg(\neg A)$ équivaut à A

$A \rightarrow B$ équivaut à $\neg A \vee B$

Lois de Morgan

$\neg(A \wedge B)$ équivaut à $\neg A \vee \neg B$

$\neg(A \vee B)$ équivaut à $\neg A \wedge \neg B$

Logique des propositions (autres équivalences)



- **Distributivité**

$(A \wedge B) \vee C$ équivaut à $(A \vee C) \wedge (B \vee C)$

$(A \vee B) \wedge C$ équivaut à $(A \wedge C) \vee (B \wedge C)$

- **Commutativité**

$A \wedge B$ équivaut à $B \wedge A$

$A \vee B$ équivaut à $B \vee A$

- **Associativité**

$A \wedge (B \wedge C)$ équivaut à $(A \wedge B) \wedge C$

$A \vee (B \vee C)$ équivaut à $(A \vee B) \vee C$

- **Contraposée**

$A \Rightarrow B$ équivaut à $\neg B \Rightarrow \neg A$



- **Règles d'inférences (règles de dérivation)**

Modus ponens

Si A et $(A \rightarrow B)$ Alors on **déduit** B (noté $A, (A \rightarrow B) \vdash B$)

Modus tollens

Si $\neg B$ et $(A \rightarrow B)$ Alors on **déduit** $\neg A$

Enchaînement

Si $A \rightarrow B$ et $B \rightarrow C$ alors on **déduit** $A \rightarrow C$

- **Axiomes, théorème, démonstration, décidabilité**

- axiomes: formules de départ ($\vdash A$)
- théorèmes: formules dont il existe une démonstration
- démonstration: enchaînement de dérivations (ou inférences)
- décidabilité: logique des proposition est décidable (il existe un procédé fini permettant de décider si une formule est un théorème ou non)

Logique des prédictats

Fondement de l'IA

2021-2022



La logique des prédictats est la représentation la plus courante des faits et règles

- MV est une sonde : $\text{sonde}(\text{mv})$
- Toutes les sondes volent : $\forall x, \text{sonde}(x) \rightarrow \text{vole}(x)$
- Toutes les sondes, qui ne sont pas en panne, volent :
$$\forall x, \text{sonde}(x) \wedge \neg \text{panne}(x) \rightarrow \text{vole}(x)$$

Logique des prédictats



Exemple d'outils pour logique des prédictats:

Le langage Prolog permet de raisonner sur base de prédictats

- MV est une sonde : **sonde(mv)**
sonde(mv).
- Toutes les sondes volent : $\forall x; \text{sonde}(x) \rightarrow \text{vole}(x)$
vole(X) :- sonde(X).

On peut poser la question, qui vole ? **? vole(X)**

Réponse : **X = mv**

- Toutes les sondes, qui ne sont pas en panne, volent :
 $\forall x; \text{sonde}(x) \wedge \neg \text{panne}(x) \rightarrow \text{vole}(x)$
vole(X) :- sonde(X), not(enPanne(X)).



Prédicat n-aire:

- Prédicat unaire :

MV est une sonde : **sonde(mv)**

- Prédicat binaire :

MV est une (is a) sonde : **is-a(mv, sonde)**

- Prédicat ternaire :

MV voyage de la Terre vers Mars : **voyage(mv, terre, mars)**

Logique des prédictats



- Les types des attributs du prédicat voyage sont : Voyageur, Origine, Destination:
 $Voyage(Voyageur, Origine, Destination)$
- Pour le voyage sur Mars ($voyageMars(mv, terre, mars)$), le Voyageur est mv, l'Origine est la terre, la Destination est mars
- On peut transformer le prédicat ternaire au binaire: $voyageur(voyageMars, mv) \wedge origine(voyageMars, terre) \wedge destination(voyageMars, mars)$

Transformation de prédicat du n-aire au binaire

Règle:

- Soit un prédicat : $\text{nomPredicat}(\text{val1}, \text{val2}, \dots, \text{valn})$
ou les types des valeurs $\text{val1}, \dots, \text{valn}$ sont $\text{typeV1}, \dots, \text{typeVn}$
- La transformation donne :
 $\text{typeV1}(\text{nomPredicat}, \text{val1}) \wedge \dots \wedge \text{typeVn}(\text{nomPredicat}, \text{valn})$



Logique d'ordre 0

- Soit l'exemple suivant:

« L'obtention d'une note de 12 en math et en physique est une condition suffisante pour avoir l'année»

- Traduction en logique d'ordre 0 :

A : avoir un 12 en math B : avoir un 12 en physique C : avoir l'année

$$A \wedge B \rightarrow C$$

- On applique la règle *Modus Ponens* pour faire les déductions :

Si A est vrai et $A \rightarrow B$ alors B est vrai



Exercice d'application:

1. Ecrivez l'énoncé suivant sous forme de formules en logique d'ordre 0 :

«En allumant votre ordinateur, si le ventilateur de l'alimentation ne démarre pas remplacez l'alimentation. Sinon, si l'écran reste noir, et la carte mère sonne de 1-3bips vérifiez votre RAM, 4pibs remplacez la carte mère, 5-7bips vérifiez le processeur, 8bips vérifiez la carte graphique. Sinon, si le test du BIOS s'affiche avec un message « no system disque » remplacez le disque dur, sinon, si le blocage est au niveau du message d'accueil du système faites une restauration avec le CD de sauvegarde. Sinon si vous avez accès à votre bureau, faites soit une analyse du disque dur avec un scandisk ou une analyse avec votre antivirus. Si aucune erreur n'a été détectée avec une analyse scandisk faites une analyse avec votre antivirus et vice-versa. Si un virus est détecté après une analyse, le mettre en quarantaine. Si aucun virus n'a été détecté après une analyse et aucune erreur n'a été détectée après un scandisk, faites une restauration du système à partir du CD de sauvegarde »

Logique des prédictats

2021-2022



2. Comment peut-on utiliser ces formules pour faire le diagnostic de la panne « Démarrage du ventilateur de l'alimentation, écran noir, 8pibs ».
3. Peut-on suggérer une restauration avec le CD de sauvegarde si : « On a accès au bureau et on n'a trouvé ni erreur ni virus ».

Logique des prédictats



Correction de l'exercice

1- Extraire les faits:

- | | |
|---|-----------------------------------|
| a: Démarrage du ventilateur de l'alimentation | i: No system disque |
| b: Remplacer l'alimentation | m: Disque dur |
| c: Ecran noir | n: Blocage à l'accueil système |
| d: 1-3bips | o: Restauration CD |
| e: RAM | p: Accès au bureau |
| f: 4bips | q: Analyse scandisk |
| g: Carte mère | r: Analyse antivirus |
| h: 5-7bips | s: Erreur détectée |
| i: Processeur | t: Virus trouvé |
| j: 8bips | u: Mettre le virus en quarantaine |
| k: Carte graphique | x : variable intermédiaire |



- Extraire les règles:

Règle 1: $\neg a \rightarrow b$

Règle 2: $a \wedge c \rightarrow x$

Règle 3: $x \wedge d \rightarrow e$

Règle 4: $x \wedge f \rightarrow g$

Règle 5: $x \wedge h \rightarrow i$

Règle 6: $x \wedge j \rightarrow k$

Règle 7: $l \rightarrow m$

Règle 8: $n \rightarrow o$

Règle 9: $p \rightarrow q$

Règle 10: $p \rightarrow r$

Règle 11: $q \wedge \neg s \rightarrow r$

Règle 12: $r \wedge \neg t \rightarrow q$

Règle 13: $q \wedge r \wedge \neg s \wedge \neg t \rightarrow o$

Règle 14: $r \wedge t \rightarrow u$

Merci pour votre attention





Fondement de l'IA

Cours: Intélligence Artificielle

2ème année informatique de gestion

Enseignante: Dr. Saoussen Mathlouthi

Saoussen.mathlouthi@ensi-uma.tn

2021-2022

Séance 5: 21/10/2021



Chapitre 2:

Représentation des connaissances

Plan



Représentation des connaissances

Les règles de production

La logique d'ordre 0 et d'ordre 1

Les réseaux sémantiques

Autres méthodes de représentation de connaissances

Représentation des connaissances

Fondement de l'IA

2021-2022



La qualité d'un système intelligent est celle de sa base de connaissance : **BC**

Dans un système intelligent on a 3 composants

- Une **BC**
- Une partie pour faire les inférences (appelé moteur d'inférence ou interpréteur **MI**)
- Une structure de contrôle pour orienter le raisonnement **SC**

Représentation des connaissances

Fondement de l'IA

2021-2022



On entend par connaissances toutes les formes de savoir de l'homme:

- des faits: des définitions (le chien est un animal)
- des événements: aspects temporels (x a rencontré y en 2000)
- des inférences :(s'il pleut ...)
- des règles de savoir faire (pour réussir l'examen il faut ...)

Représentation des connaissances

Fondement de l'IA

2021-2022

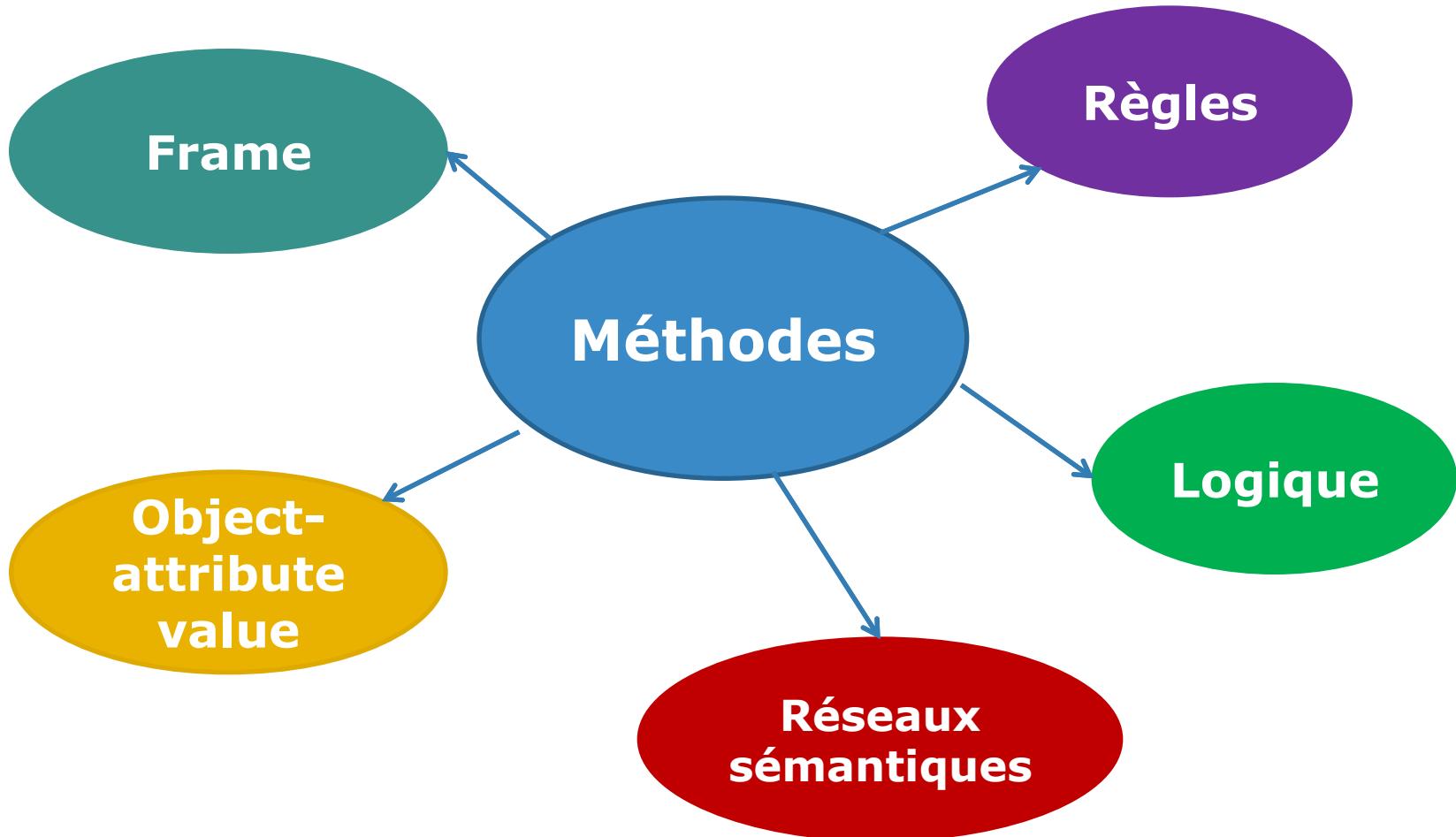


- Représentation des connaissances: action de rendre sensible quelque chose au moyen d'une figure, d'un symbole, d'un signe
- Plusieurs méthodes de représentation ont été proposées dans la littérature:
 - ⊕ Logique (logique d'ordre 0, logique d'ordre 1, floue ...)
 - ⊕ Réseaux sémantiques
 - ⊕ Règles de production
 - ⊕ Objets structurés (frame)
 - ⊕ Approche orientée objet

Représentation des connaissances

Fondement de l'IA

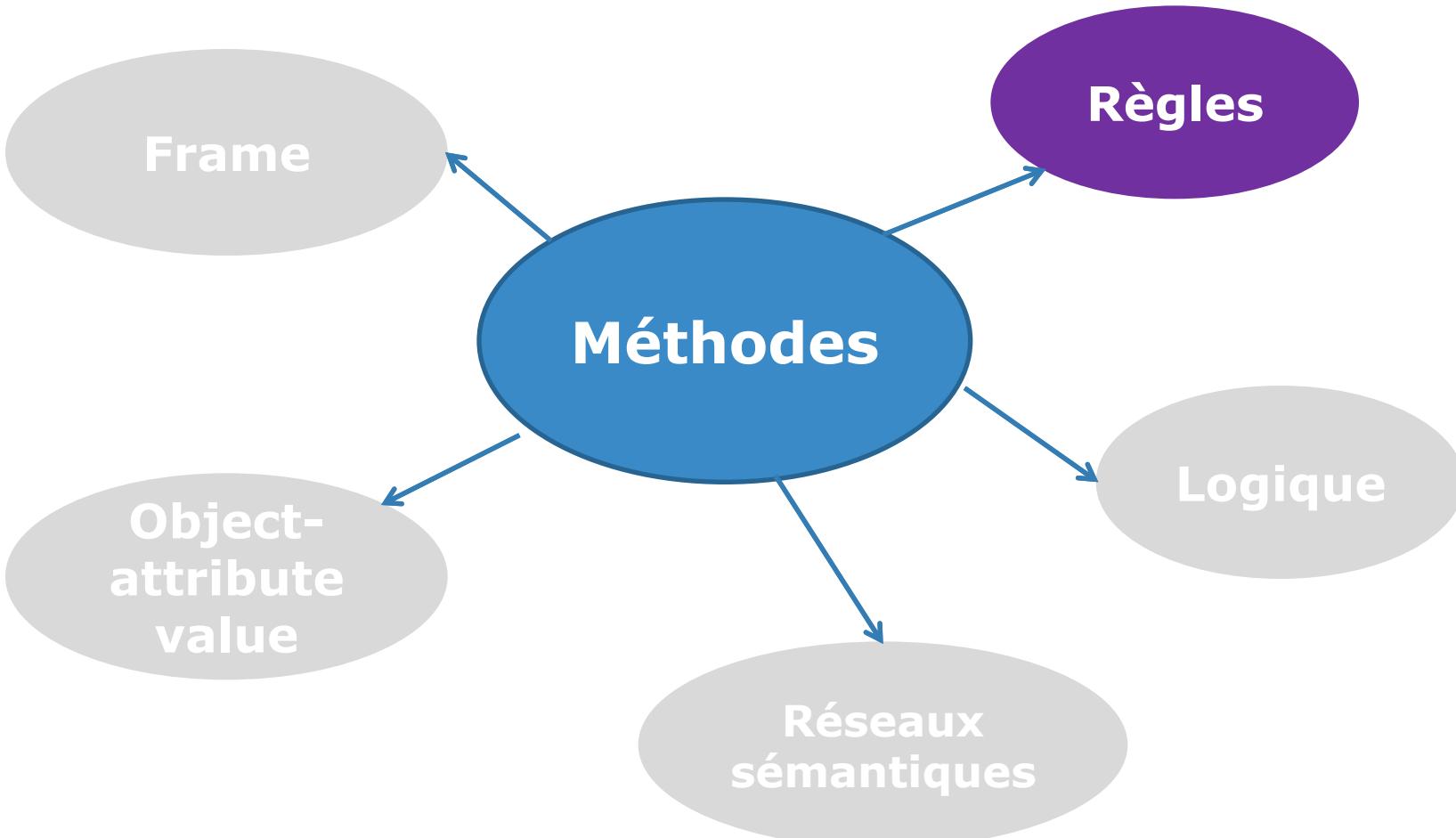
2021-2022



Représentation des connaissances

Fondement de l'IA

2021-2022



Règles de production

Fondement de l'IA

2021-2022



- C'est une quantité de connaissances autonome, de la forme:

SI condition ALORS conclusion

- ❖ Condition et conclusion sont des expressions logiques
- ❖ Condition doit être vérifiée pour que la règle s'applique
- ❖ Conclusion peut correspondre à l'ajout ou au retrait d'un fait ou d'une hypothèse, au déclenchement d'une action, etc

Règles de production



Fondement de l'IA

2021-2022

Exemple:

Règle 1: Si **x** a mal à la gorge et suspecte une infection bactérienne
Alors **x** a une angine streptococcique

Règle 2: Si température de **x** >37°
Alors **x** a de la fièvre

Règle 3: Si **x** a été malade > un mois et **x** a de la fièvre alors **x** suspecte une infection bactérienne

la température du patient = 38°

Le patient a été malade > 2 mois

Le patient a mal à la gorge

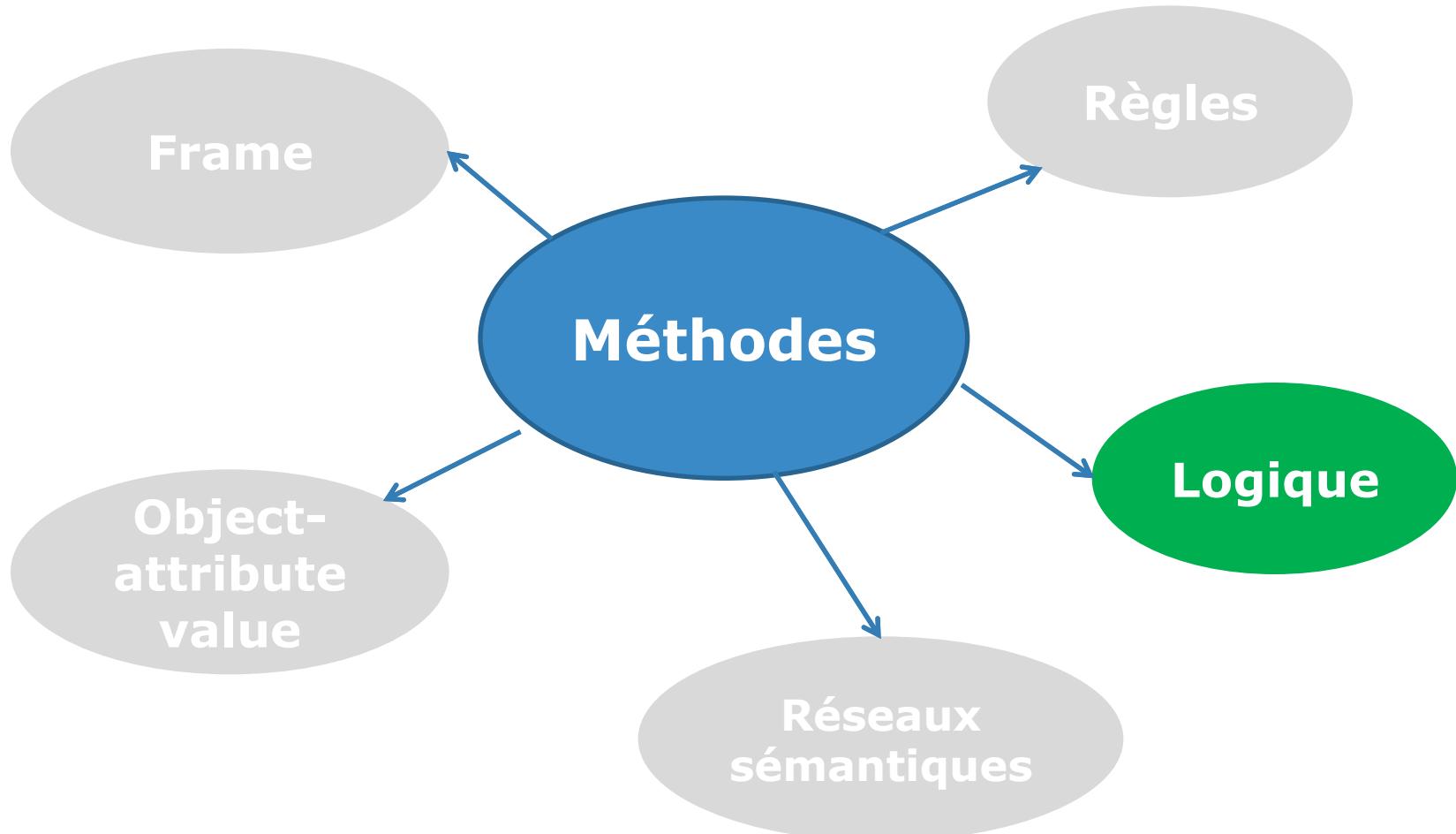
Conclusion ?

Le patient a une angine streptococcique

Représentation des connaissances

Fondement de l'IA

2021-2022





Logique d'ordre 0: Logique des propositions

Exemple 1:

S'il pleut Asma va rester à la maison

Soient :

P: Il pleut

Q: Asma va rester à la maison

P → Q

Exemple 2:

Tous les grecs sont mortels, Socrate est grec donc Socrate est mortel

P: tous les grecs sont mortels

Q: Socrate est grec

R : Socrate est mortel

P ∧ Q → R



Logique d'ordre 0: Logique des propositions

- Logique des proposition est la forme la plus simple du raisonnement logique
- Syntaxe:
 - Une proposition vrai, faux
 - Variables propositionnelles ou atome P, Q, R, etc
 - Connecteurs pour représenter des propositions plus complexes: \wedge (Et),
 \vee (Ou), \neg (Non), \rightarrow (Implique), \leftrightarrow (Equivalent)



Logique d'ordre 1: Logique des prédictats

- Insuffisance de la logique d'ordre 0 si on veut déduire des propositions pour des ensembles d'éléments
 - exprimer que tous les oiseaux volent
Vole(oiseau 1), Vole(oiseau 2), ..., Vole(oiseau n)
 - exprimer que certains oiseaux ne volent pas
???
- La table de vérité ne peut pas représenter ce genre de relations
- **La logique des prédictats** permet ce genre de représentation



Logique d'ordre 1: Logique des prédictats

Dans la logique d'ordre 1 on peut aussi utiliser des:

● Connecteurs: \Rightarrow , \vee , \wedge , $=$, \neg

● Quantificateurs \forall , \exists

1) Quantificateur Universel, \forall , vraie pour toutes les valeurs de ses variables

$\forall X \text{ aime}(X, \text{ice cream})$

2) Quantificateur existentiel, \exists , vraie pour quelques valeurs dans le domaine du discours

$\exists Y \text{ Amis}(Y, \text{Peter})$



Exercice:

Ecrire en logique des Prédictats

- S1. Pour tout crime, il y a quelqu'un qui l'a commis**
- S2. Seul les gens malhonnêtes commettent des crimes**
- S3. Ne sont arrêtés que les gens malhonnêtes**
- S4. Les gens malhonnêtes arrêtés ne commettent pas de crime**
- S5. Il y a que des crimes**
- S6. Il y a des gens malhonnêtes non arrêtés**



Correction de l'exercice:

S1. Pour tout crime, il ya quelqu'un qui l'a commis

$C(X) : X \text{ est un crime}$

$\text{Commettre}(Y, X) : Y \text{ a Commis } X$

$(\forall X), C(X) \rightarrow (\exists Y) \text{ Commettre}(Y, X)$

S2. Seul les gens malhonnêtes commettent des crimes

$M(Y) : Y \text{ est malhonnête}$

$(\forall X) (\forall Y) C(X) \wedge \text{Commettre}(Y, X) \rightarrow M(Y)$

S3. Ne sont arrêtés que les gens malhonnêtes
(tout objet/ si l'objet est arrêté alors cet objet est malhonnête)

$A(X) : X \text{ est arrêté}$

$(\forall X) A(X) \rightarrow M(X)$

Logique des prédictats



S4. Les gens malhonnêtes arrêtés ne commettent pas de crime
(pour toute personne malhonnête et arrêté, il n'existe pas de crime commis par elle)

$C(X)$: X est un crime

Commettre(Y, X) : Y a Commis X

$M(Y)$: Y est malhonnête

$A(X)$: X est arrêté

$(\forall X) M(X) \wedge A(X) \rightarrow \neg (\exists Y) (C(Y) \wedge \text{Commettre}(X,Y))$

S5. Il y a que des crimes

$(\forall X) C(X)$

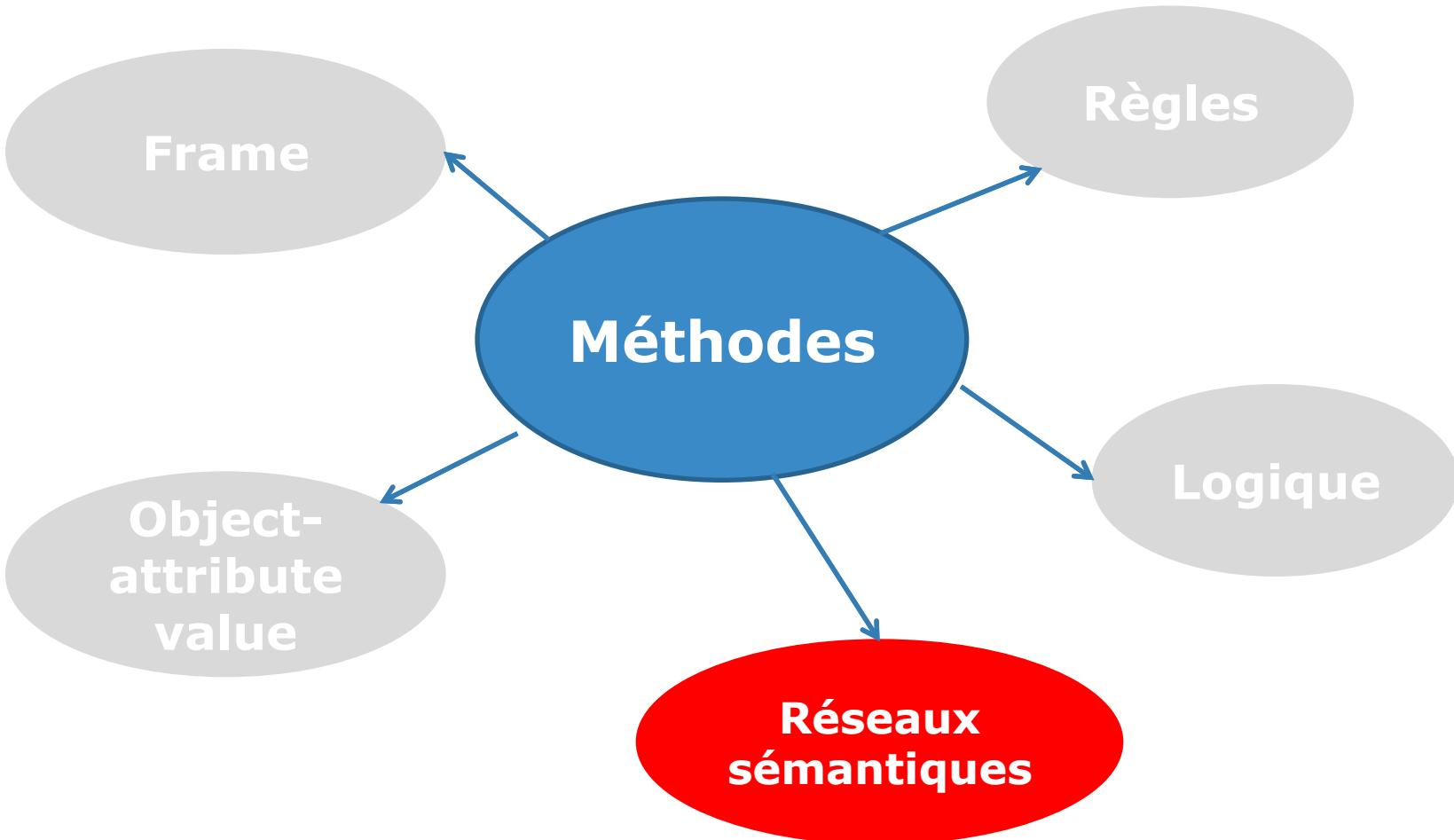
S6. Il y a des gens malhonnêtes non arrêtés

$(\exists X) M(X) \wedge \neg A(X)$

Représentation des connaissances

Fondement de l'IA

2021-2022



Les réseaux sémantiques

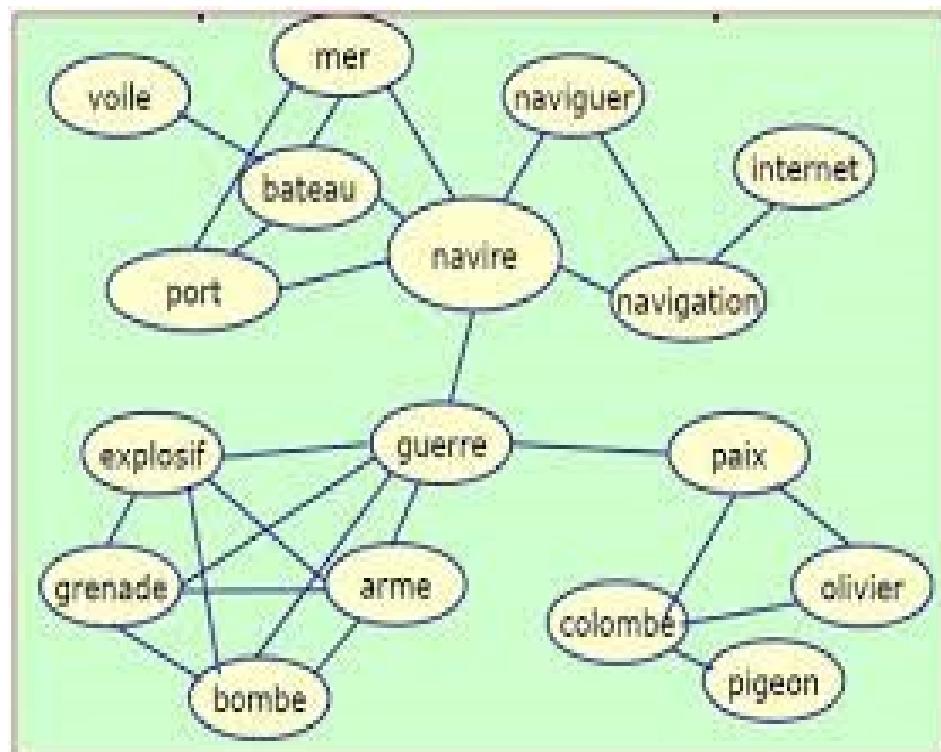
Fondement de l'IA

2021-2022



Les réseaux sémantiques se présentent comme des ensembles de points ou nœuds étiquetés.

- Chaque nœud représente un concept :
 - Objet, fait ou situation
 - Une notion générale de logique (non, et, ou, ...)
 - Un sous-réseau.
- Les nœuds sont reliés par des arcs orientés figurant les relations sémantiques qui existent entre les nœuds.



Les réseaux sémantiques



- Les réseaux sémantiques utilisent généralement deux relations très particulières concernant des objets de l'un des types individu ou classe :

- **est_un (is-a)**: relation entre un individu et une classe exprimant l'appartenance. Elle décrit le fait qu'un concept est considéré comme une "instance" d'une famille d'objets. Correspond à l'appartenance (ϵ) en théorie des ensembles.



- **sorte_de (a-K-o)**: relation entre deux classes exprimant l'inclusion. Elle décrit le fait que le réseau considéré est un sous-réseau ou encore qu'il fait partie de la famille d'un réseau donné. Correspond à l'inclusion (\subset) en théorie des ensembles.



Les réseaux sémantiques



Exemple:

Silvestre est un chat.

Le chat est un mammifère.

Le chien est un mammifère.

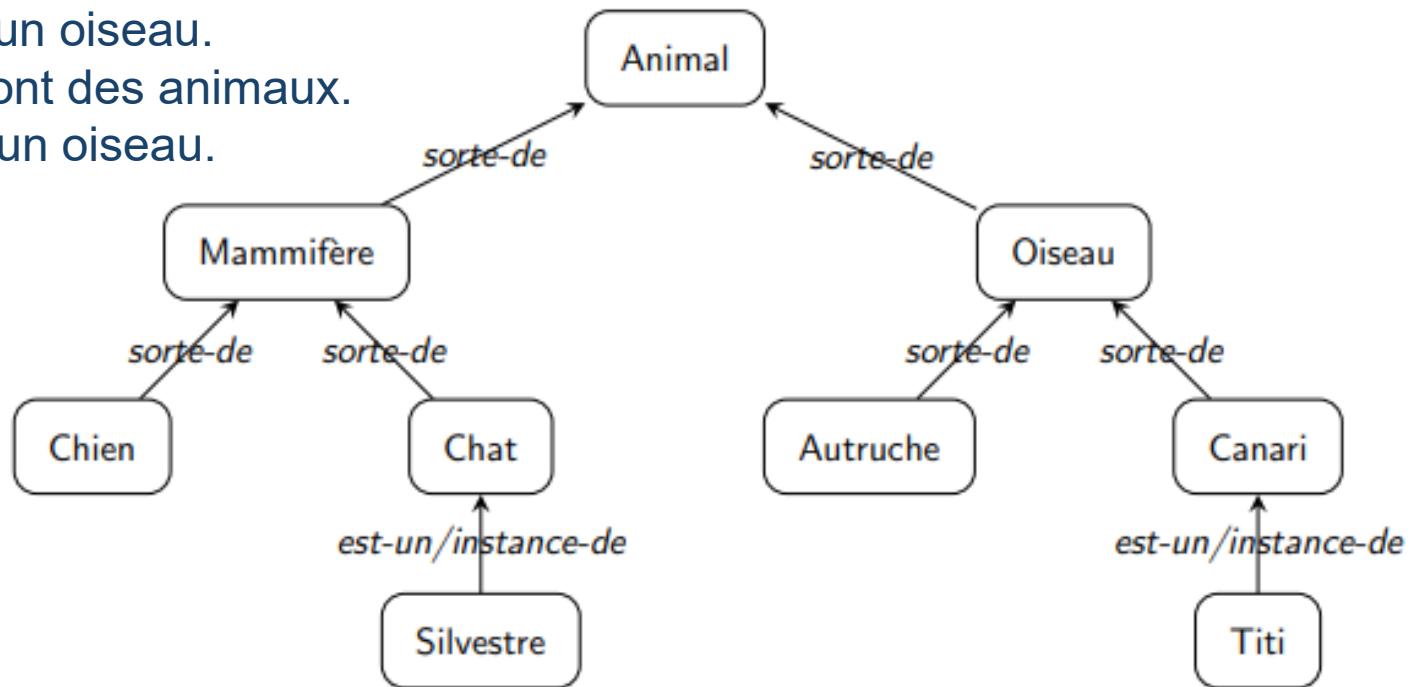
Les mammifères sont des animaux.

Titi est un Canari.

Le Canari est un oiseau.

Les oiseaux sont des animaux.

L'Autruche est un oiseau.



Représentation des connaissances

Fondement de l'IA

2021-2022



Exercice:

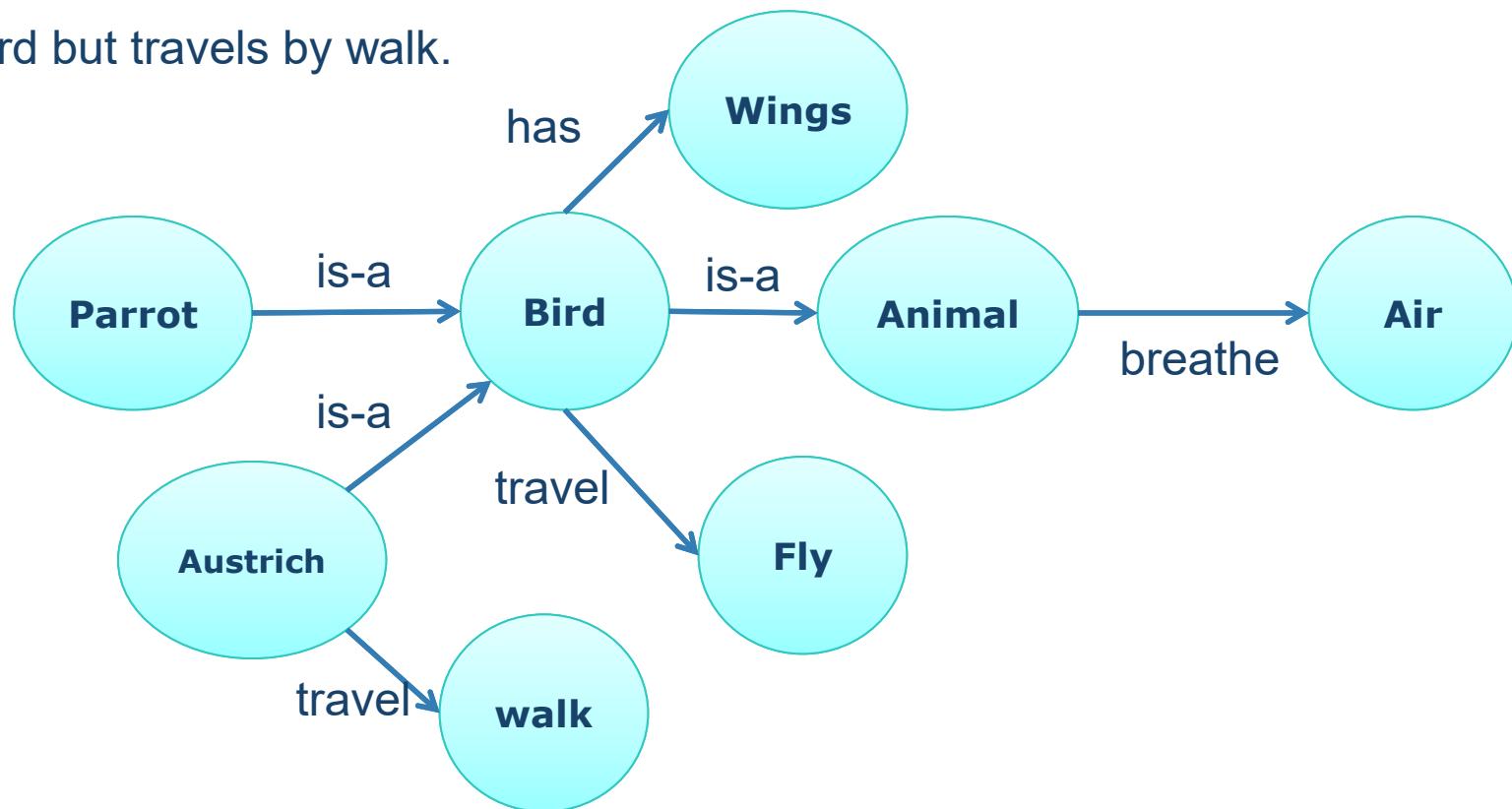
FACT : Parrot is a bird. Typically bird has wings and travels by flying. Bird category falls under animal kingdom. All animal requires air to breathe. Ostrich is a bird but travels by walk.

Représentation des connaissances



Exercice 1:

FACT : Parrot is a bird. Typically bird has wings and travels by flying. Bird category falls under animal kingdom. All animal requires air to breathe. Ostrich is a bird but travels by walk.





Exercice 2:

Un félin est un carnivore. Un carnivore est un animal qui a les yeux dirigés vers l'avant et qui mange de la viande. Les pattes d'un félin ont des griffes à leurs extrémités. Un félin est un mammifère. Grisou et Garfield sont des chats. Grisou est un chat mâle. Léo est un lion. Les chats et les lions sont des félins. Un cheval est un équidé. Un équidé est un mammifère.



Pour construire ce réseau sémantique :

1. On lit d'abord tout le texte pour repérer les **objets** et les **relations**. En effet, leur réutilisation d'une phrase à l'autre permet de mettre en évidence leur utilité.
2. On sépare les **propositions atomiques** et on les met sous forme tabulaire.
3. On dessine successivement les parties de **réseau** correspondant aux **propositions atomiques**.
4. Éventuellement, on revient sur les choix effectués et on rectifie le réseau.

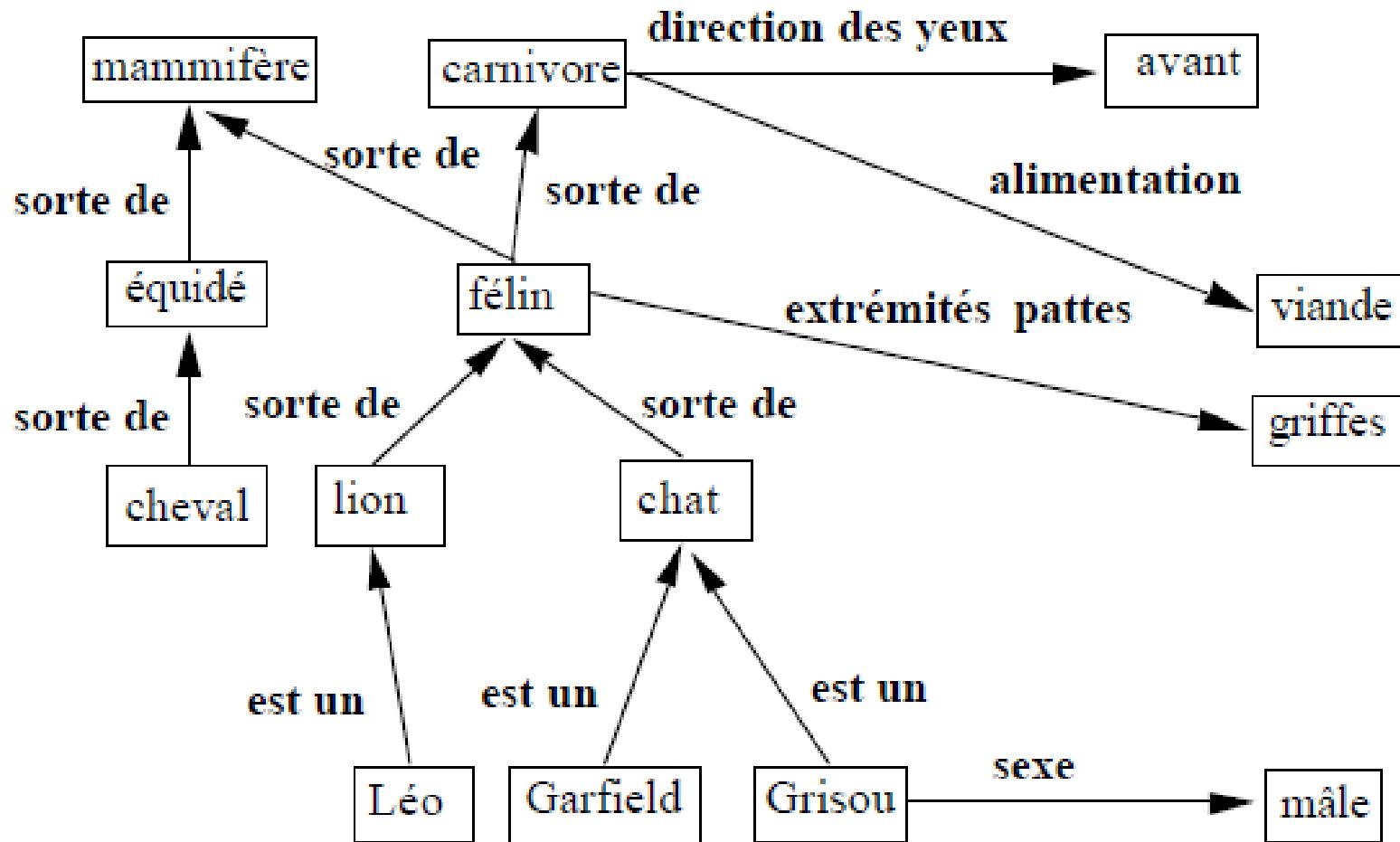
Les réseaux sémantiques



Propositions atomiques :

félin	sorte_de	carnivore
carnivore	direction des yeux	avant
carnivore	alimentation	viande
félin	extrémités pattes	griffes
félin	sorte_de	mammifère
Grisou	est_un	chat
Garfield	est_un	chat
Grisou	sexé	mâle
Léo	est_un	lion
chat	sorte_de	félin
lion	sorte_de	félin
cheval	sorte_de	équidé
équidé	sorte_de	mammifère

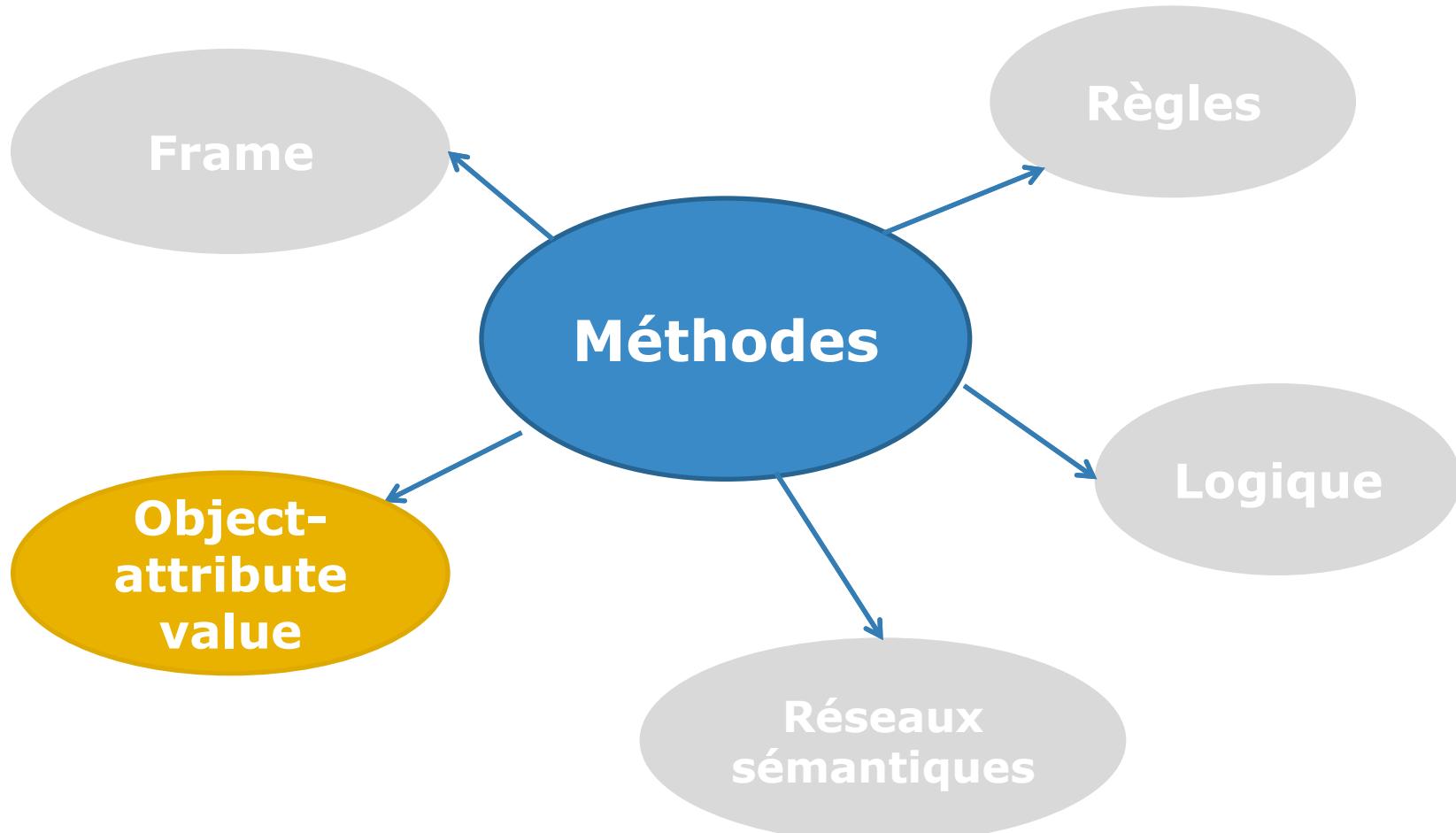
Les réseaux sémantiques



Représentation des connaissances

Fondement de l'IA

2021-2022



Object-Attribute Value

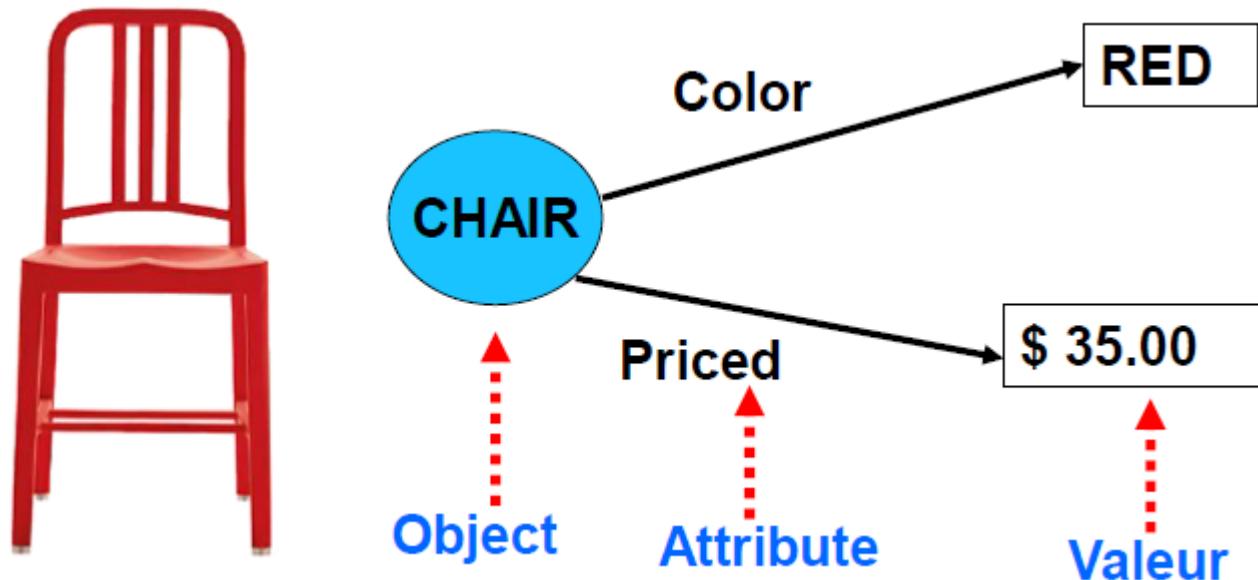


- Faiblesses des réseaux sémantiques: manque de sémantique formelle et de terminologie standard
- Adopter une approche objet dans l'IA
Object-Attribute-Value (OAV) triplets :
Assigner une valeur à un attribut d'un objet

Object-Attribute Value



Fact: The chair's color is red and priced at \$ 35.00

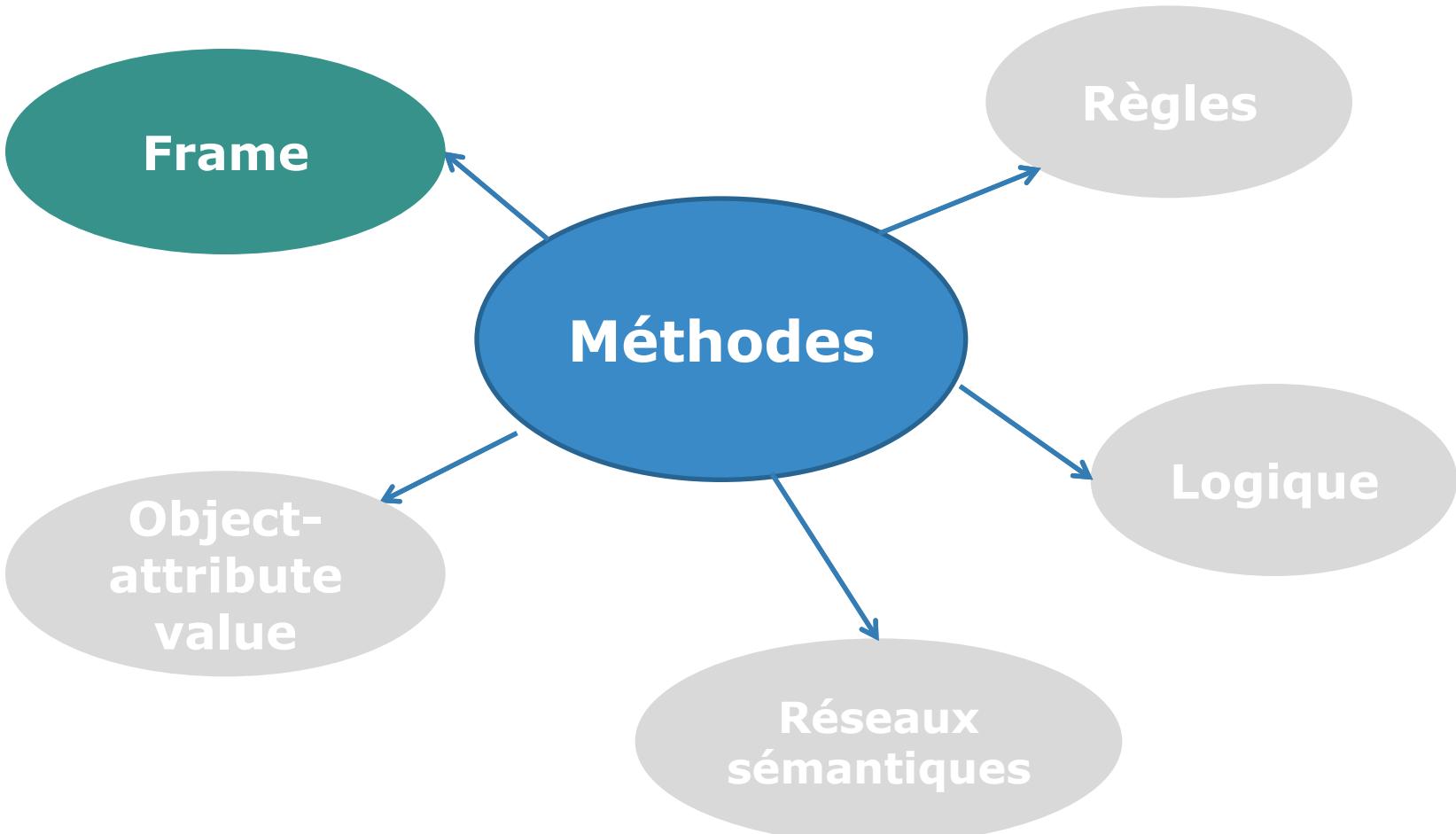


Représentation des connaissances



Fondement de l'IA

2021-2022





Fondement de l'IA

Cours: Intélligence Artificielle

2ème année informatique de gestion

Enseignante: Dr. Saoussen Mathlouthi

Saoussen.mathlouthi@ensi-uma.tn



Chapitre 3:

Recherche dans un espace d'état

Plan



Moteur d'inférence

Stratégies du moteur d'inférence

Cycle du moteur d'inférence

Chaînage avant VS Chaînage arrière

MI à chaînage avant

MI à chaînage arrière

Moteur d'inférence

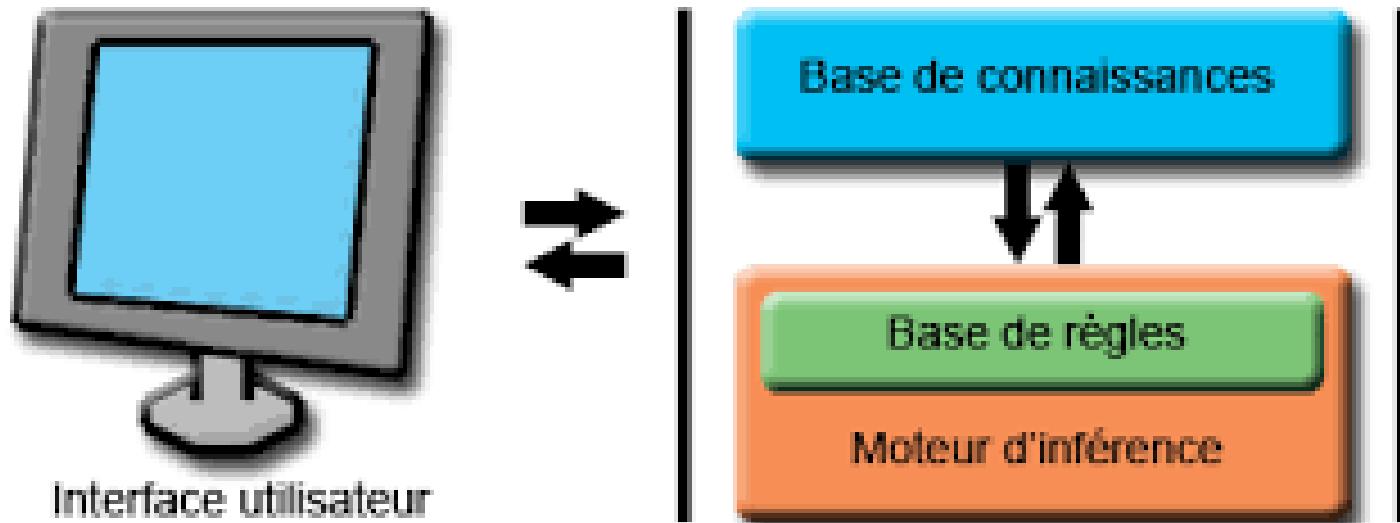
Fondement de l'IA

2021-2022



Le moteur d'inférences est le cerveau du système intelligent.

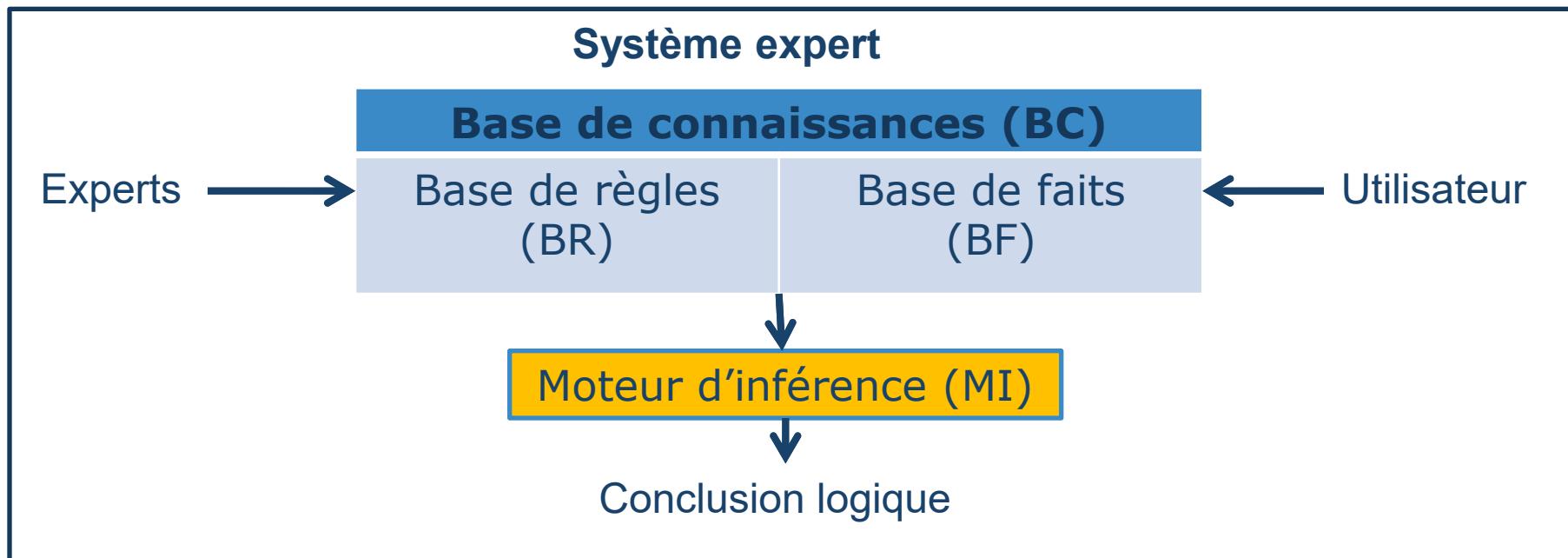
C'est un programme qui simule le raisonnement humain



Moteur d'inférence



- **Le moteur d'inférence** : il s'agit d'un programme qui parcourt la base de connaissance en appliquant les règles aux faits connus donnés.
- **Le moteur d'inférence** est l'équivalent de l'esprit logique de l'expert humain qui raisonne sur des faits connus pour en sortir de nouvelles vérités.
- **Le moteur d'inférence** est un algorithme d'exploration d'un espace d'état utilisé par un système expert





3 mode de raisonnement:

- **Chaînage avant:** commence par les faits et aboutit au but
- **Chaînage arrière:** commence par le but et aboutit aux faits existant dans la base de faits
- **Chaînage mixte:** combinaison des deux (avant et arrière)



Le moteur d'inférence fonctionne en deux phases:

❖ **Phase d'évaluation**

- 1) Etape de Filtrage ou détection: Elle consiste à définir pour l'ensemble des règles de BC, les règles potentiellement applicables → résultat ensemble de règles.
- 2) Etape de Sélection avec réduction de conflits: Elle consiste à choisir parmi l'ensemble des règles applicables, la règle à appliquer effectivement → résultat une règle.

❖ **Phase d'exécution**

Elle consiste à appliquer la règle choisie et mettre à jour la base de faits BF.



Phase d'évaluation: Réduction des conflits

Le Moteur d'Inférences choisit les Règles qui doivent être effectivement déclenchées selon une stratégie : heuristiques

Pour résoudre le conflit, entre plusieurs règles à priori applicables suivant une configuration de la BF, le système doit choisir une ou plusieurs Règles suivant **certaines heuristiques** :

- ✓ La première règle qui s'applique au contrôle
- ✓ La règle la plus propriétaire suivant un ou plusieurs critères définis par l'expert
- ✓ La règle la plus spécifique (la condition la plus détaillée qui s'applique à la BF)
- ✓ La règle se référant à un élément le plus récemment ajouté
- ✓ Sélection d'un sous-ensemble de Règles résultant de l'application de métarègles
- ✓ Déclenchement prioritaire des règles amenant le plus grand nombre de conclusions
- ✓ Ne pas choisir, explorer toutes les règles applicables en parallèle
- ✓ Arbitrairement



Phase d'exécution: Régime du contrôle du MI

- **Régime irrévocable**

Pour des MI très simples

Le MI s'arrêtera dès qu'il atteint la Saturation de la Base de Règles Déclenchables

- **Régime par tentative (Backtracking)**

Le MI examine la possibilité de déclencher d'autres règles déclenchables

Le MI opère par un retour arrière et remet en cause les règles déclenchées précédemment

Chaînage avant VS Chaînage arrière



Fondement de l'IA

2021-2022

Déductif	Inductif
Chaînage avant	Chaînage arrière
<ul style="list-style-type: none">• Raisonnement dirigé vers les données (des données vers les buts, bas-haut)• Tirer des conclusions de l'information données• Lorsque les F de la BF sont dans les conditions des R: tirer à partir des informations déjà établies	<ul style="list-style-type: none">• Raisonnement dirigé vers les buts(des buts vers les données, haut-bas)• Commence par un but à prouver ou une explicative de ce que se doit se produire• Lorsque certains F de BF sont considérés comme étant à établir ou à évaluer• On parle de problème à résoudre ou d'hypothèses à vérifier ou de buts à atteindre



Mécanisme du chaînage avant: un mécanisme simple

- ❖ Pour déduire un **F** particulier, on cherche les Règles dont les prémisses sont connus **jusqu'à ce que F à déduire soit connu** ou **qu'aucune Règles ne soit plus déclenchable**
- ❖ Utilisé quand on cherche les **conséquences de l'ajout de nouveaux faits**
- ❖ Utilise la règle d'inférence **Modus Ponens**: de **P** et de **si P alors Q** on déduit **Q**
- ❖ Le système **n'a pas de but**, il déclenche des règles **jusqu'à épuisement ou arrêt**
- ❖ Plusieurs stratégies, parmi eux::
 - Production des faits en "**largeur d'abord**"; après **épuisement du conflict set**
 - Production des faits en "**profondeur d'abord**"; privilégier les Règles qui contiennent en prémissse un fait **qui vient d'être établi**
- ❖ Le fait à établir peut ne pas être connu, saturation de la **BC** (déduire tous les faits déductibles)



Algorithme:

Soit **BF**: base de faits; **BR**: base de règles (conclusion positive); **F**: fait que l'on cherche à établir

Recherche de la déduction possible de **F**

```
ChaînageAvant(BF, BR, F)
```

DEBUT

TANT QUE ($F \notin BF$) **ET** ($\exists R \in BR, \text{applicable}(R)$)

FAIRE

choisir une règle applicable R (*par métarègles, heuristiques, ...*)

$BR = BR - R$ (*désactivation de R*)

$BF = BF \cup \text{concl}(R)$ (*déclenchement de la règle R , ajout de sa conclusion*)

FIN TANT QUE

SI $F \in BF$ **ALORS** F est établi

SINON F n'est pas établi

FIN



Exercice:

Base de Connaissances

Base de Faits :

$$BF = \{ A, B, C, D, E, F, G, H, R, S, T, U \}$$

Base de Règles :

$BR = \{$

$R_1: A, B, C \rightarrow H$

$R_2: A, U, C \rightarrow F$

$R_3: E, G, B \rightarrow S$

$R_4: D, G \rightarrow C$

$R_5: A, E \rightarrow B$

$R_6: U, S, T \rightarrow F$

$R_7: G, H \rightarrow R$

$R_8: D, E \rightarrow T$

$R_9: R, S, H \rightarrow F$

$R_{10}: A, U \rightarrow B$

$\}$

Hypothèses : $\{A, D, E, G\}$

**Chaînage
Avant**



But : $F ?$



Simulation chaînage Avant

Hypothèse 0:
 $\{A, D, E, G\}$

Quelles sont les règles déclenchables?

R1: A, B, C -> H

R2: A, U, C -> F

R3: E, G, B -> S

R4: D, G -> C

R5: A, E -> B

R6: U, S, T -> F

R7: G, H -> R

R8: D, E -> T

R9: R, S, H -> F

R10: A, U -> B



Simulation chaînage Avant

Hypothèse 0:
 $\{A, D, E, G\}$

Hypothèse 1:
 $\{A, D, E, G, C, B, T\}$

Quelles sont les règles déclenchables?

R1: A, B, C -> H

R2: A, U, C -> F

R3: E, G, B -> S

R4: D, G -> C

R5: A, E -> B

R6: U, S, T -> F

R7: G, H -> R

R8: D, E -> T

R9: R, S, H -> F

R10: A, U -> B

R1: A, B, C -> H

R2: A, U, C -> F

R3: E, G, B -> S

R4: D, G -> C

R5: A, E -> B

R6: U, S, T -> F

R7: G, H -> R

R8: D, E -> T

R9: R, S, H -> F

R10: A, U -> B



Simulation chaînage Avant

Hypothèse 0:
 $\{A, D, E, G\}$

Hypothèse 1:
 $\{A, D, E, G, C, B, T\}$

Hypothèse 2:
 $\{A, D, E, G, C, B, T, H, S\}$

Quelles sont les règles déclenchables?

R1: $A, B, C \rightarrow H$

R2: $A, U, C \rightarrow F$

R3: $E, G, B \rightarrow S$

R4: $D, G \rightarrow C$

R5: $A, E \rightarrow B$

R6: $U, S, T \rightarrow F$

R7: $G, H \rightarrow R$

R8: $D, E \rightarrow T$

R9: $R, S, H \rightarrow F$

R10: $A, U \rightarrow B$

R1: $A, B, C \rightarrow H$

R2: $A, U, C \rightarrow F$

R3: $E, G, B \rightarrow S$

R4: $D, G \rightarrow C$

R5: $A, E \rightarrow B$

R6: $U, S, T \rightarrow F$

R7: $G, H \rightarrow R$

R8: $D, E \rightarrow T$

R9: $R, S, H \rightarrow F$

R10: $A, U \rightarrow B$

R1: $A, B, C \rightarrow H$

R2: $A, U, C \rightarrow F$

R3: $E, G, B \rightarrow S$

R4: $D, G \rightarrow C$

R5: $A, E \rightarrow B$

R6: $U, S, T \rightarrow F$

R7: $G, H \rightarrow R$

R8: $D, E \rightarrow T$

R9: $R, S, H \rightarrow F$

R10: $A, U \rightarrow B$



Simulation chaînage Avant

Hypothèse 0:
 $\{A, D, E, G\}$

Hypothèse 1:
 $\{A, D, E, G, C, B, T\}$

Hypothèse 2:
 $\{A, D, E, G, C, B, T, H, S\}$

Hypothèse 3: $\{A, D, E, G, C, B, T, H, S, R\}$

Quelles sont les règles déclenchables?

R1: A, B, C -> H

R2: A, U, C -> F

R3: E, G, B -> S

R4: D, G -> C

R5: A, E -> B

R6: U, S, T -> F

R7: G, H -> R

R8: D, E -> T

R9: R, S, H -> F

R10: A, U -> B

R1: A, B, C -> H

R2: A, U, C -> F

R3: E, G, B -> S

R4: D, G -> C

R5: A, E -> B

R6: U, S, T -> F

R7: G, H -> R

R8: D, E -> T

R9: R, S, H -> F

R10: A, U -> B

R1: A, B, C -> H

R2: A, U, C -> F

R3: E, G, B -> S

R4: D, G -> C

R5: A, E -> B

R6: U, S, T -> F

R7: G, H -> R

R8: D, E -> T

R9: R, S, H -> F

R10: A, U -> B

R9: R, S, H -> F

MI à chaînage arrière

2021-2022



Fondement de l'IA

Base de Connaissances

Base de Faits :

$$BF = \{ A, B, C, D, E, F, G, H, R, S, T, U \}$$

Base de Règles :

$BR = \{$

R1: A, B, C \rightarrow H

R2: A, U, C \rightarrow F

R3: E, G, B \rightarrow S

R4: D, G \rightarrow C

R5: A, E \rightarrow B

R6: U, S, T \rightarrow F

R7: G, H \rightarrow R

R8: D, E \rightarrow T

R9: R, S, H \rightarrow F

R10: A, U \rightarrow B

$\}$

But: F?

Chaînage
Arrière



Hypothèses : {A, D, E, G}

MI à chaînage arrière

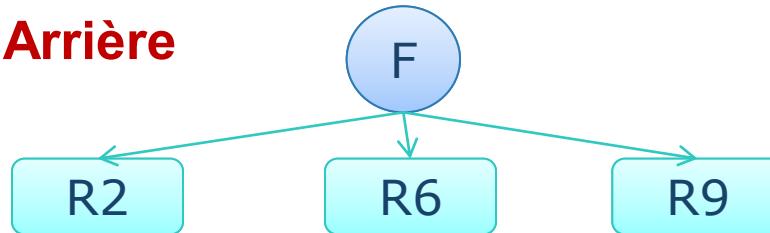


Fondement de l'IA

2021-2022

Simulation chaînage Arrière

Hypothèse:
 $\{A, D, E, G\}$



R1: A, B, C -> H

R2: A, U, C -> F

R3: E, G, B -> S

R4: D, G -> C

R5: A, E -> B

R6: U, S, T -> F

R7: G, H -> R

R8: D, E -> T

R9: R, S, H -> F

R10: A, U -> B

MI à chaînage arrière



Fondement de l'IA

2021-2022

Simulation chaînage Arrière

Hypothèse:
 $\{A, D, E, G\}$

R1: A, B, C -> H

R2: A, U, C -> F

R3: E, G, B -> S

R4: D, G -> C

R5: A, E -> B

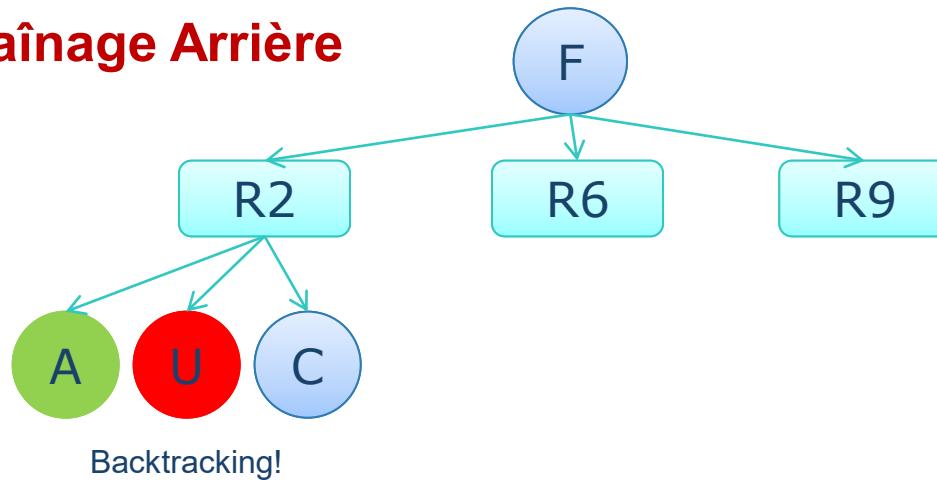
R6: U, S, T -> F

R7: G, H -> R

R8: D, E -> T

R9: R, S, H -> F

R10: A, U -> B



MI à chaînage arrière



Fondement de l'IA

2021-2022

Simulation chaînage Arrière

Hypothèse:
 $\{A, D, E, G\}$

R1: A, B, C -> H

R2: A, U, C -> F

R3: E, G, B -> S

R4: D, G -> C

R5: A, E -> B

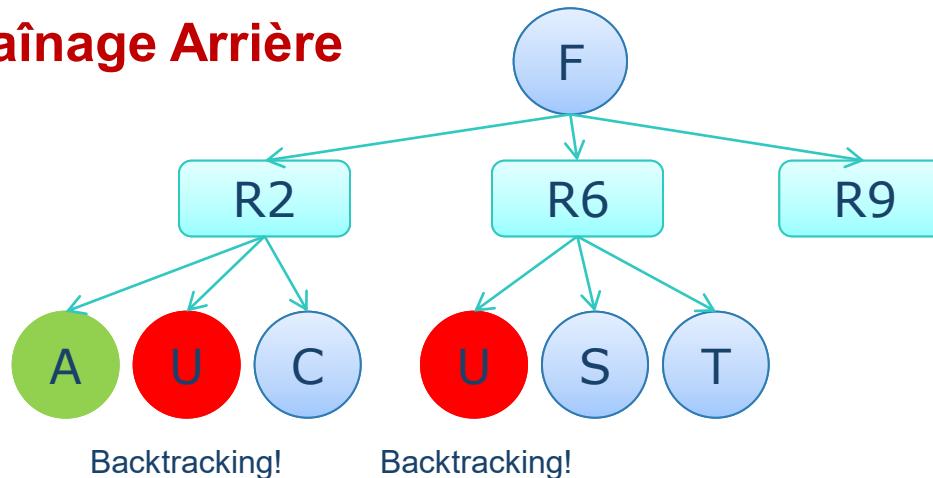
R6: U, S, T -> F

R7: G, H -> R

R8: D, E -> T

R9: R, S, H -> F

R10: A, U -> B



MI à chaînage arrière



Fondement de l'IA

2021-2022

Simulation chaînage Arrière

Hypothèse:
 $\{A, D, E, G\}$

R1: A, B, C -> H

R2: A, U, C -> F

R3: E, G, B -> S

R4: D, G -> C

R5: A, E -> B

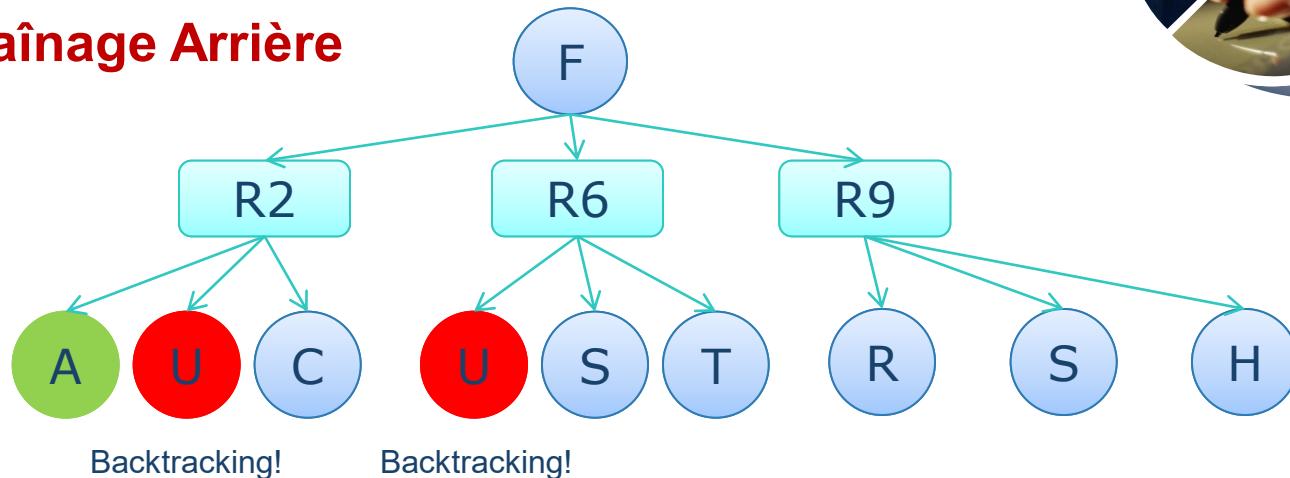
R6: U, S, T -> F

R7: G, H -> R

R8: D, E -> T

R9: R, S, H -> F

R10: A, U -> B



MI à chaînage arrière



Fondement de l'IA

2021-2022

Simulation chaînage Arrière

Hypothèse:
 $\{A, D, E, G\}$

R1: A, B, C -> H

R2: A, U, C -> F

R3: E, G, B -> S

R4: D, G -> C

R5: A, E -> B

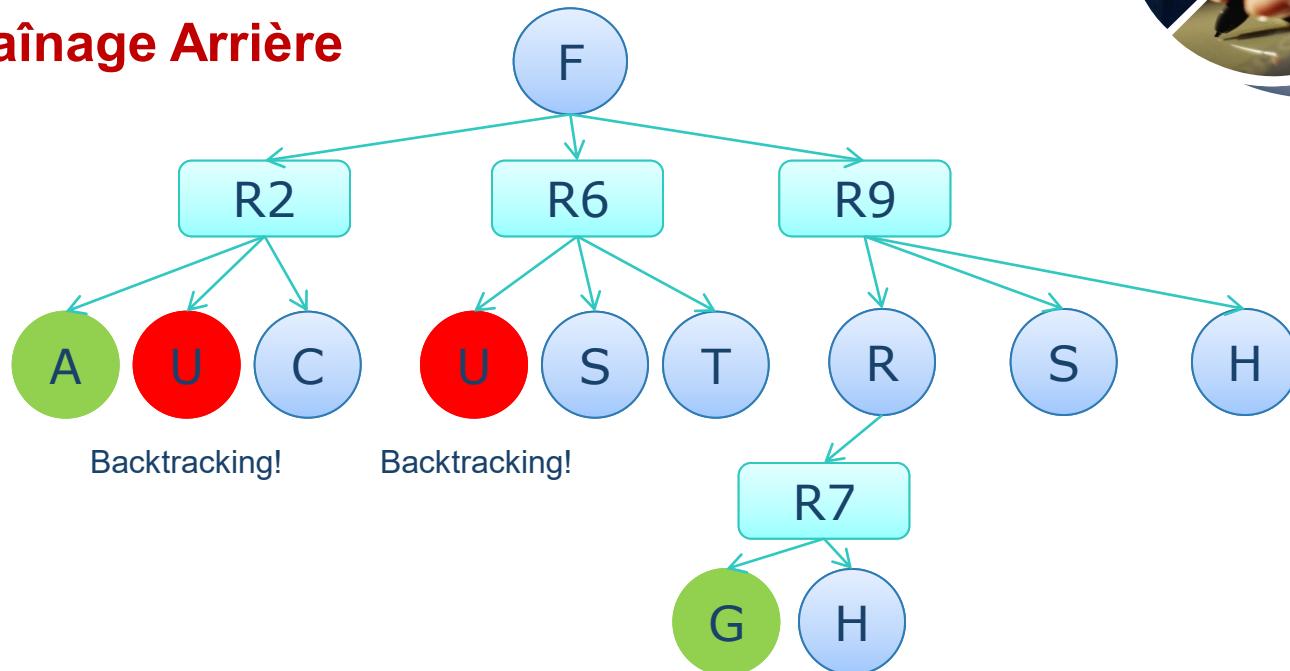
R6: U, S, T -> F

R7: G, H -> R

R8: D, E -> T

R9: R, S, H -> F

R10: A, U -> B



MI à chaînage arrière



Fondement de l'IA

2021-2022

Simulation chaînage Arrière

Hypothèse:
 $\{A, D, E, G\}$

R1: A, B, C -> H

R2: A, U, C -> F

R3: E, G, B -> S

R4: D, G -> C

R5: A, E -> B

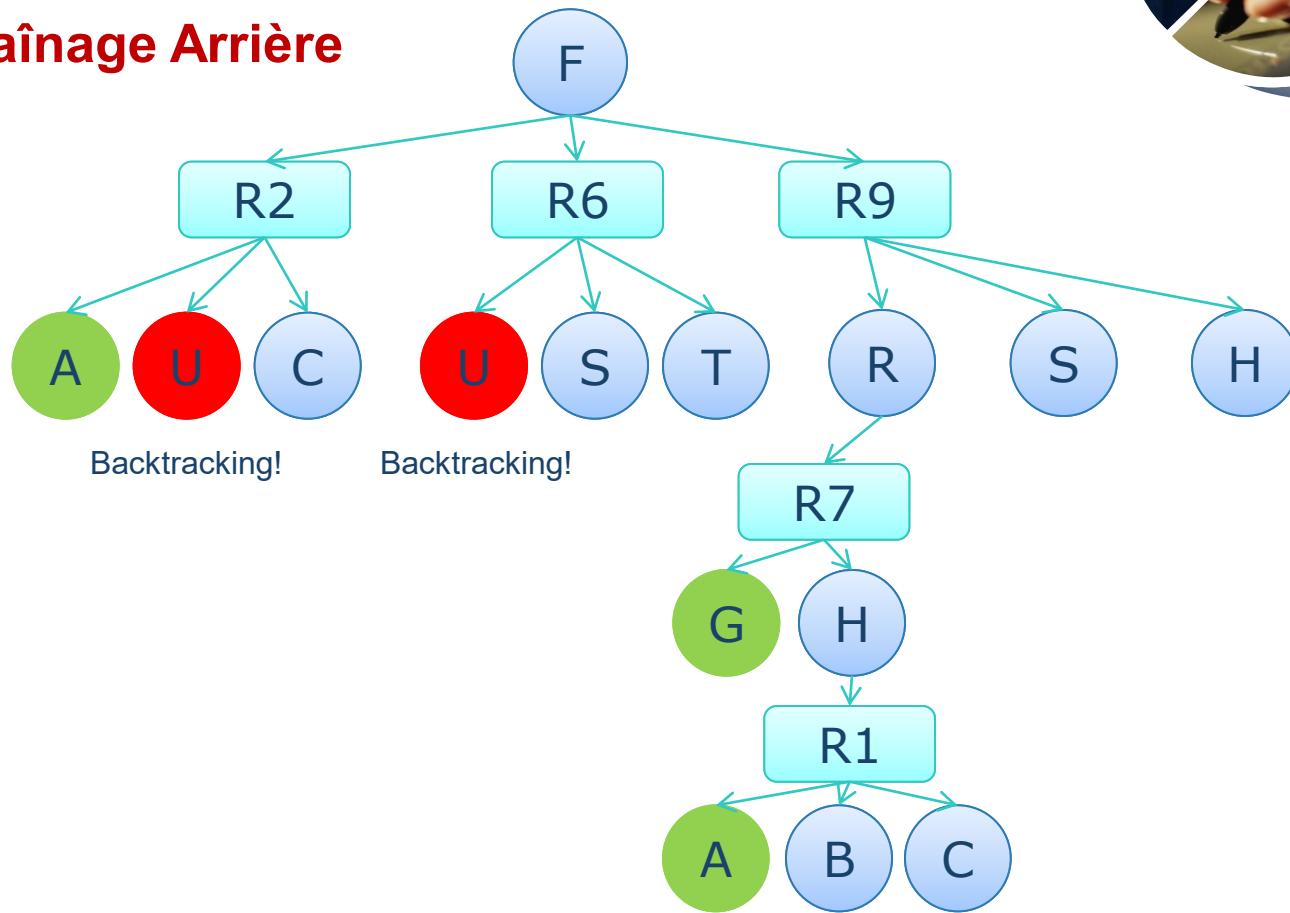
R6: U, S, T -> F

R7: G, H -> R

R8: D, E -> T

R9: R, S, H -> F

R10: A, U -> B



MI à chaînage arrière



Fondement de l'IA

2021-2022

Simulation chaînage Arrière

Hypothèse:
 $\{A, D, E, G\}$

R1: A, B, C -> H

R2: A, U, C -> F

R3: E, G, B -> S

R4: D, G -> C

R5: A, E -> B

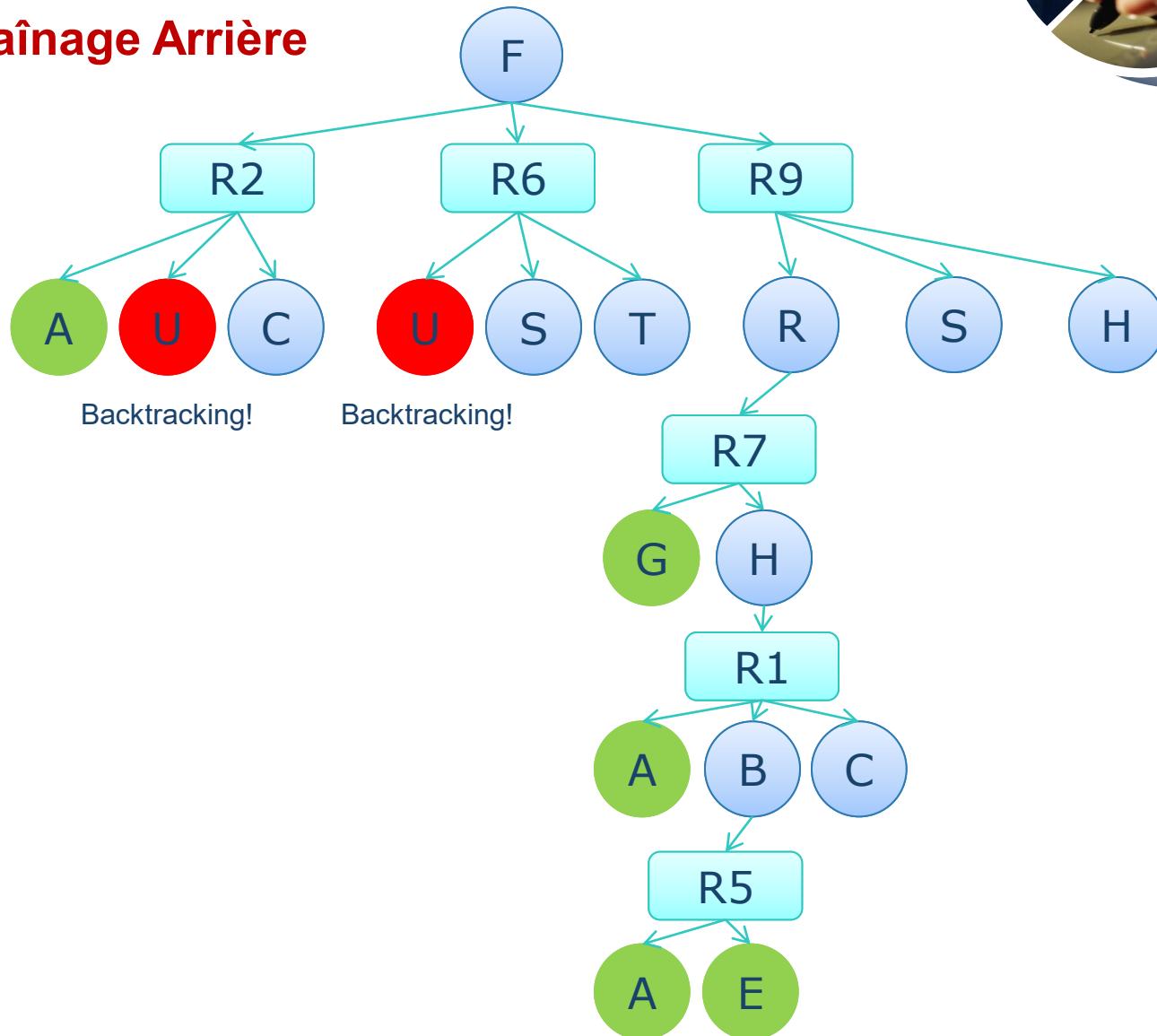
R6: U, S, T -> F

R7: G, H -> R

R8: D, E -> T

R9: R, S, H -> F

R10: A, U -> B



MI à chaînage arrière



Fondement de l'IA

2021-2022

Simulation chaînage Arrière

Hypothèse:
 $\{A, D, E, G\}$

R1: A, B, C -> H

R2: A, U, C -> F

R3: E, G, B -> S

R4: D, G -> C

R5: A, E -> B

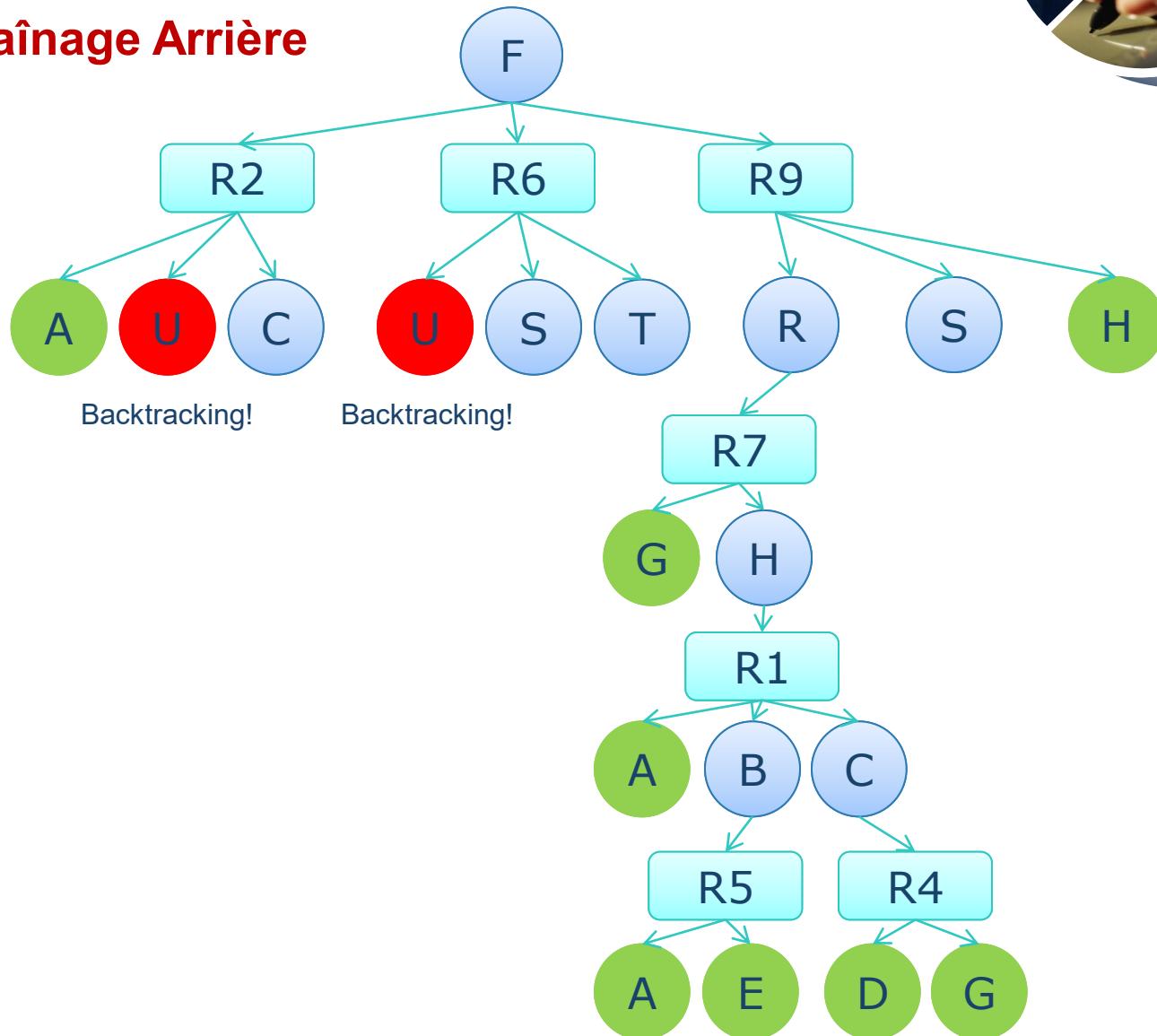
R6: U, S, T -> F

R7: G, H -> R

R8: D, E -> T

R9: R, S, H -> F

R10: A, U -> B



MI à chaînage arrière



Fondement de l'IA

2021-2022

Simulation chaînage Arrière

Hypothèse:
 $\{A, D, E, G\}$

R1: A, B, C -> H

R2: A, U, C -> F

R3: E, G, B -> S

R4: D, G -> C

R5: A, E -> B

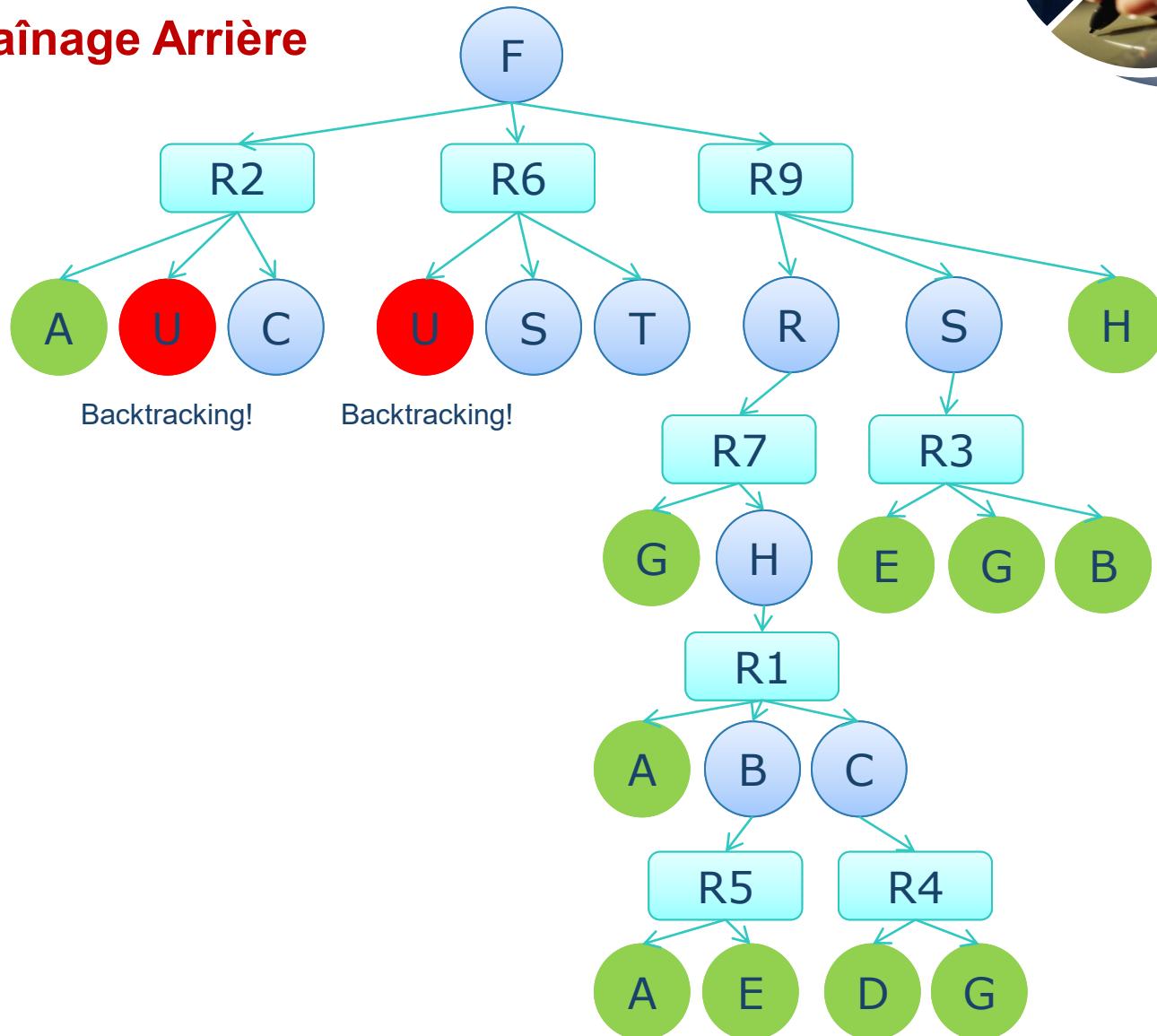
R6: U, S, T -> F

R7: G, H -> R

R8: D, E -> T

R9: R, S, H -> F

R10: A, U -> B





Exercice: Arbre ET/OU

- ❖ Soit BF = {B,C}
- ❖ Soit BR contenant :

R 1: If B and D and E then F

R 2: If G and D then A

R 3: If C and F then A

R 4: If B then X

R 5: If D then E

R 6: If X and A then H

R 7: If C then D

R 8: If X and C then A

R 9: If X and B then D

Utiliser un chaînage arrière pour prouver le fait H

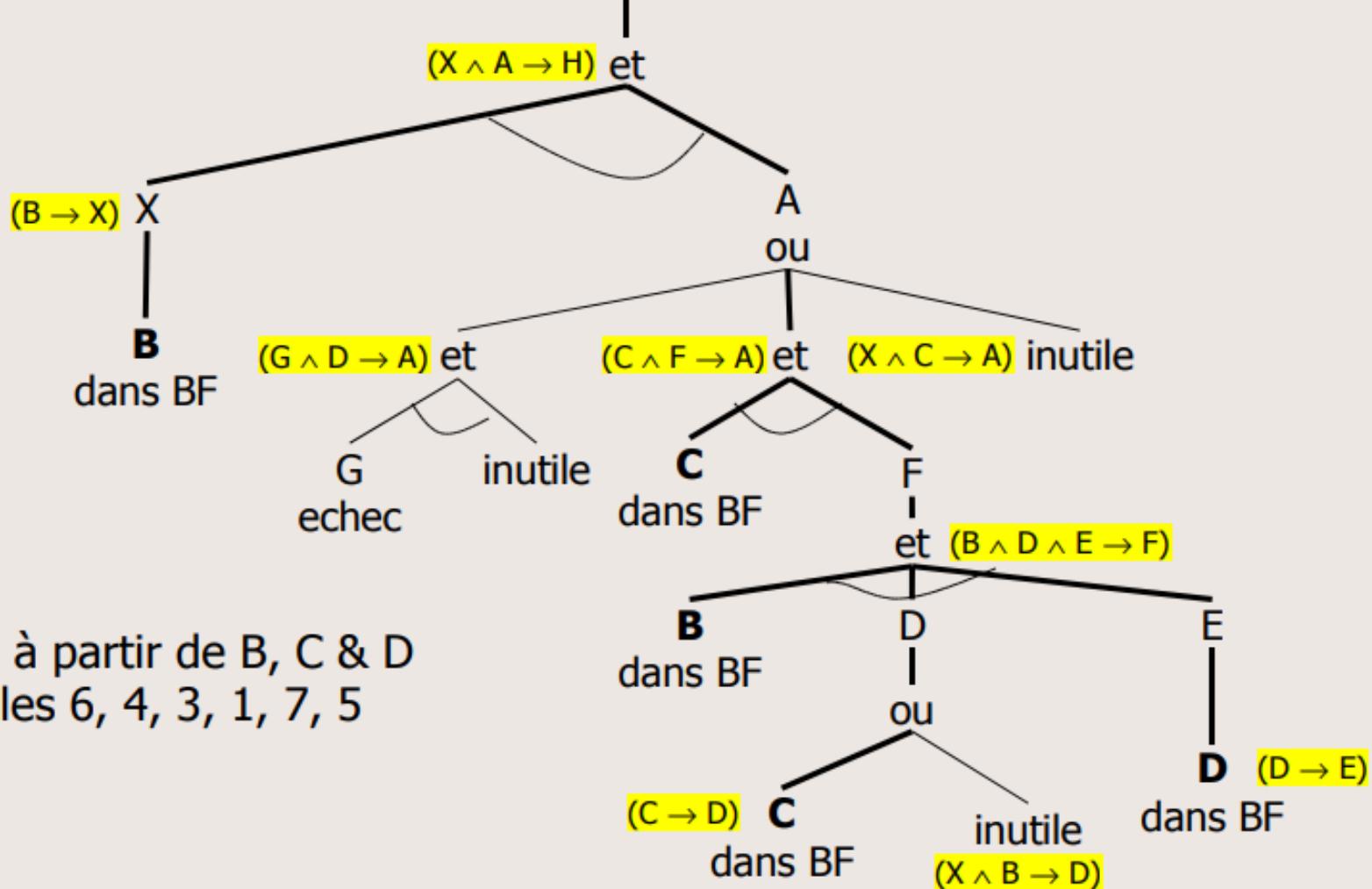
MI à chaînage arrière

2021-2022



Fondement de l'IA

- Exemple : Comment déduire le fait H ? **H**





Fondement de l'IA

Cours: Intélligence Artificielle

2ème année informatique de gestion

Enseignante: Dr. Saoussen Mathlouthi

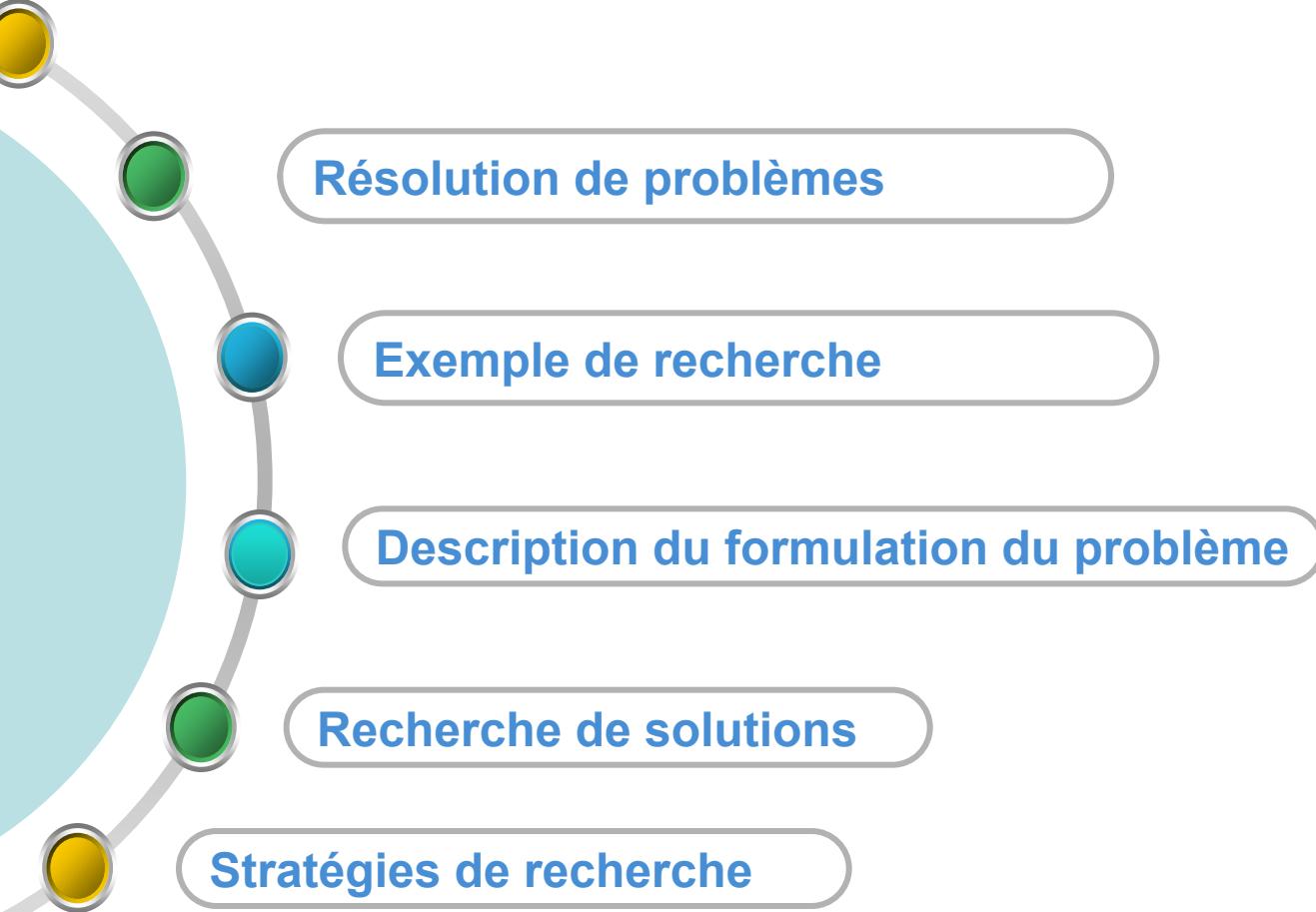
Saoussen.mathlouthi@ensi-uma.tn



Chapitre 3:

Recherche dans un espace d'état

Plan





Résolution de problèmes par recherche

- ◆ On représente un problème par un **espace d'états (arbre/graphe)**.
- ◆ Chaque état est une **configuration possible** du problème.
- ◆ Résoudre le problème consiste à trouver un **chemin** dans le graphe.
 - Parcours aveugles non informés : profondeur, largeur...
 - Parcours informés.



Comment résoudre un problème?

- ◆ Recenser les états d'un système donné et trouver parmi ces états une ou plusieurs solution.
- ◆ Le passage d'un état à un autre se fait par l'application d'une action donnée.
- ◆ Développement d'un arbre de recherche.
- ◆ Appliquer une stratégie de recherche

Résolution de problèmes

Fondement de l'IA

2021-2022



On a un problème quand:

- ◆ on se trouve dans une certaine situation: **état**
- ◆ On a un **but** à atteindre: **état ayant des caractéristiques données**
- ◆ On ne voit pas immédiatement la suite d'actions à accomplir pour atteindre ce but: **résoudre le problème**
- ◆ La résolution de problème s'effectue en progressant, de situation en situation, d'une **hypothèse** initiale jusqu'à la solution **but**

Exemple de recherche

2021-2022



Fondement de l'IA



Résoudre ce Pb :

revient à partir de P_i (pierre initiale) de proposer une stratégie permettant de parcourir l'espace d'états (les pierres) pour trouver l'état P_f (pierre finale)

- ✓ Les états sont les pierres
- ✓ Pour passer d'une pierre à l'autre, on utilise le prédictat 'sauter'
- ✓ On peut atteindre certaines pierres et pas d'autres
- ✓ A chaque état P_i on associe P_j qu'on peut atteindre à partir de P_i
- ✓ On peut se tromper et arriver sur une pierre à partir de laquelle nos forces nous interdisent d'atteindre la suivante
- ✓ On peut revenir en arrière et essayer un autre chemin (backtracking strategy)
- ✓ Chaque opérateur nous fait passer d'un état à un autre qui nous rapproche, en principe, de P_f

Exemple de recherche



Exemple: le problème des cruches (**WATER JUGS problem**)

Vous disposez de trois cruches, la première de **3 litres**, la deuxième de **5 litres** et la troisième de **9 litres**. Une source d'eau est à disposition, à volonté.

Comment obtenir exactement 7 litres ?

Exemple de recherche



WATER JUGS problem

Solution 1:

3L	5L	9L	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	
3	0	6	
0	3	6	
3	3	6	
1	5	6	
0	5	7	goal

Solution 2:

3L	5L	9L	
0	0	0	start
0	5	0	
3	2	0	
3	0	2	
3	5	2	
3	0	7	goal



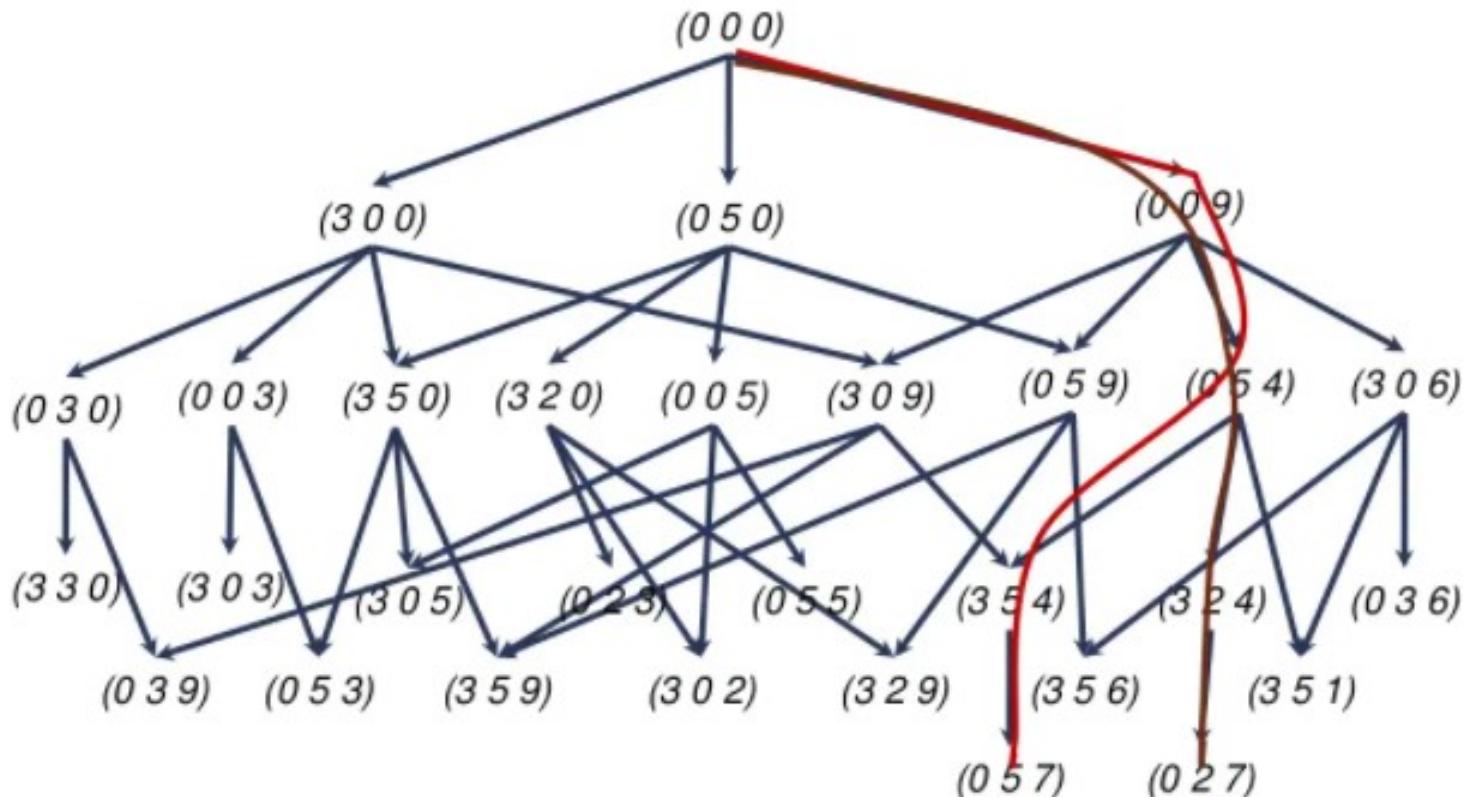
7L ?

La solution 2 est une solution optimale

Exemple de recherche



Arbre d'état du problème WATER JUGS



Résolution de problèmes



Fondement de l'IA

2021-2022

But: état(s) à atteindre

Formulation du problème: processus qui cherche quels états/quelles actions considérer pour atteindre un but donné

Exploration: processus de recherche d'une séquence d'actions qui mènent au but

Agent de résolution de problème:

Boucle : agent commence par formuler un but et un problème, recherche une séquence d'actions qui résout le problème puis les exécute une à une

Description du formulation du problème



État initial: l'état n où se trouve l'agent

Actions possibles: la fonction $S(n)$ donne la séquence des actions à faire dans l'état n = (action, état successeur)

Test de but: un test afin de voir si l'état donné est un état but

Coût du chemin: fonction (à valeurs ≥ 0) qui attribut un coût à un chemin.
Elle reflète la mesure de performance de l'agent

une **solution** est une séquence d'opérateurs menant de l'état initial à l'état final (**but**)

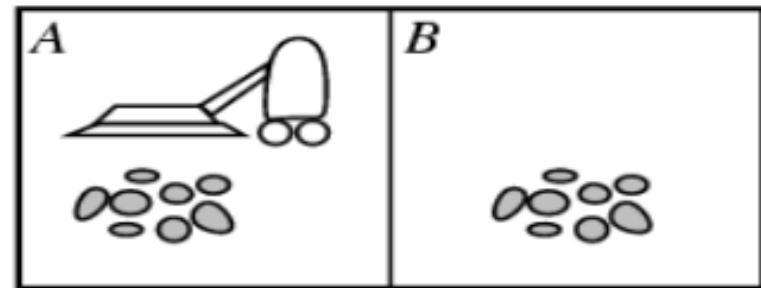
Description du formulation du problème



Exemple 1: Agent aspirateur

Formuler le problème pour le monde de l'aspirateur

- États?
- Actions?
- Test du but?
- Coût du chemin?

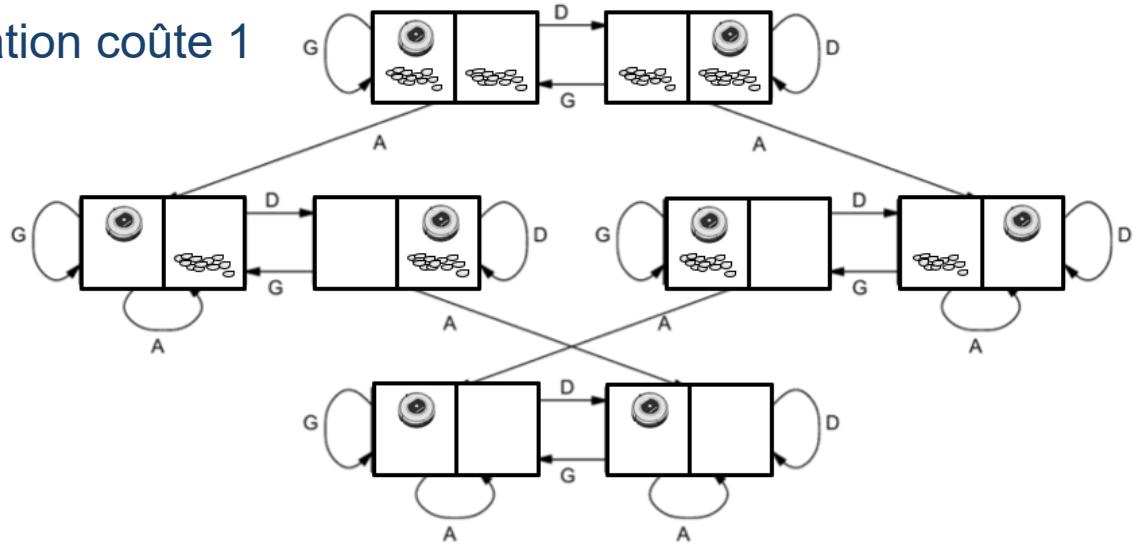


Si la case courante est sale → aspirer
Sinon : se déplace vers l'autre case

Description du formulation du problème



- **États:** position de l'agent et position de la saleté
 - soit $2 \times 2^2 = 8$ états possibles
 - état initial: n'importe quel état
- **Actions:** Chaque état possède 3 actions gauche (G), droite (D), aspire (A)
- **Test du but:** toutes les cases propres?
- **Coût du chemin:** chaque opération coûte 1



Description du formulation du problème



Exemple 2: The 8-puzzle

Formuler le problème pour le monde du 8-puzzle

- États?
- Actions?
- Test du but?
- Coût du chemin?

2	8	3
1	6	4
7	■	5

situation initiale



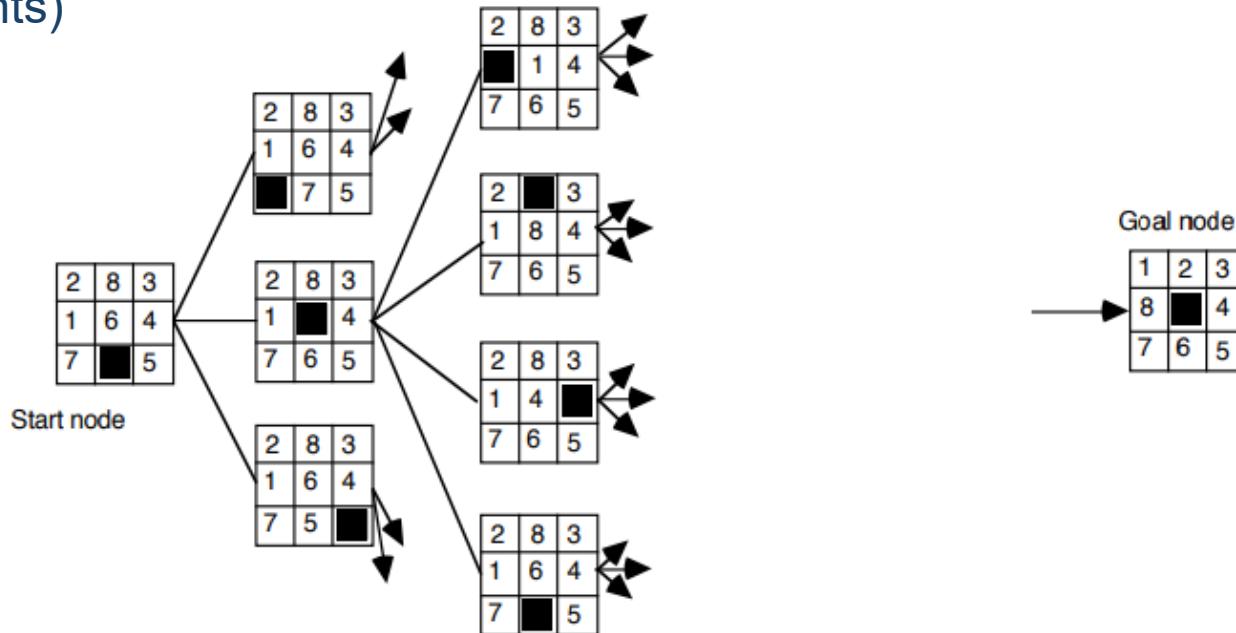
1	2	3
8	■	4
7	6	5

situation finale

Description du formulation du problème



- **États**: position des 8 pièces et du vide (une matrice 3×3 + les coordonnées de la case vide)
- **Actions**: déplacement de la case vide (HAUT, BAS, GAUCHE et DROITE).
- **Test du but**: est-ce qu'on atteint la configuration finale?
- **Coût du chemin**: chaque déplacement coûte 1 (coût total nombre de déplacements)



Description du formulation du problème



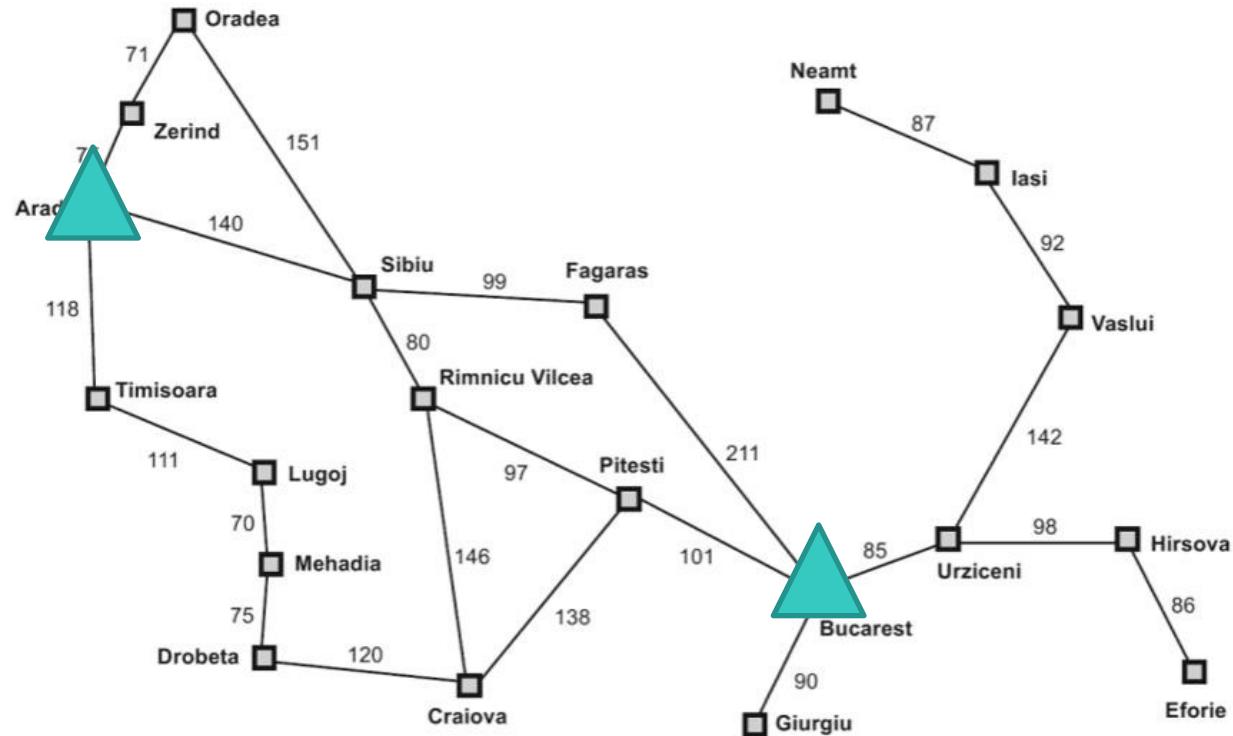
Fondement de l'IA

2021-2022

Exemple 3: recherche d'un trajet

Formuler le problème pour la recherche dans l'arbre

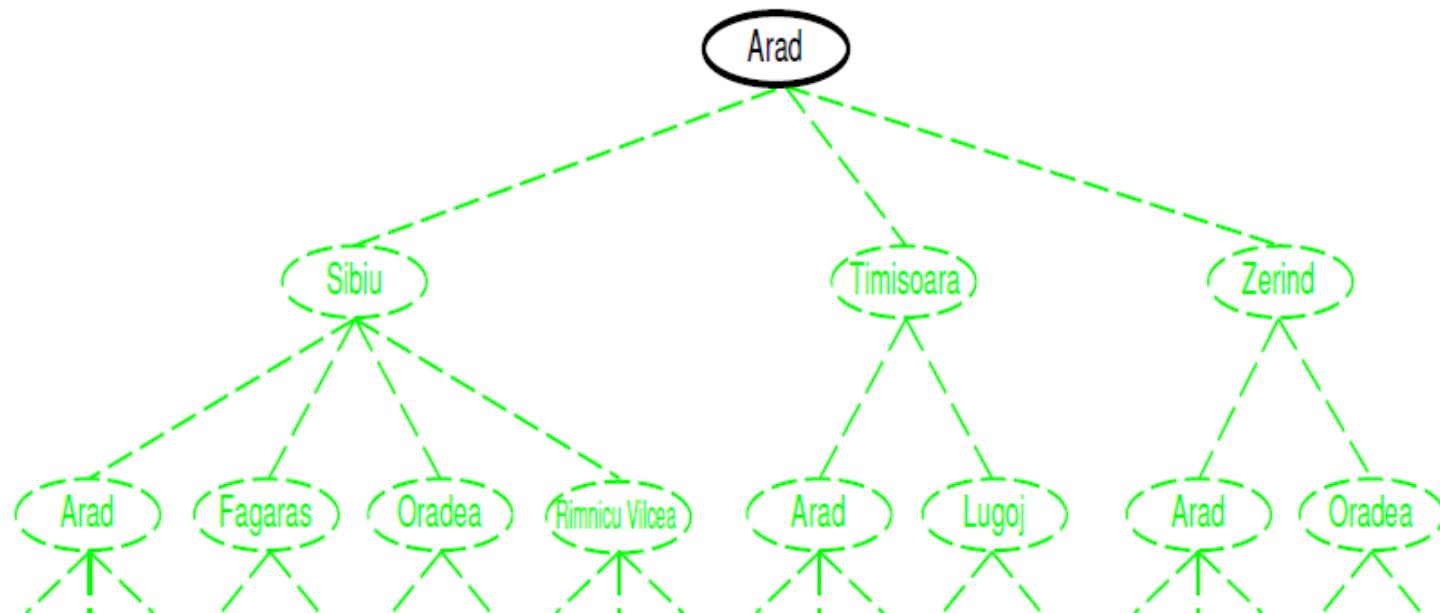
- États?
- Actions?
- Test du but?
- Coût du chemin?



Description du formulation du problème



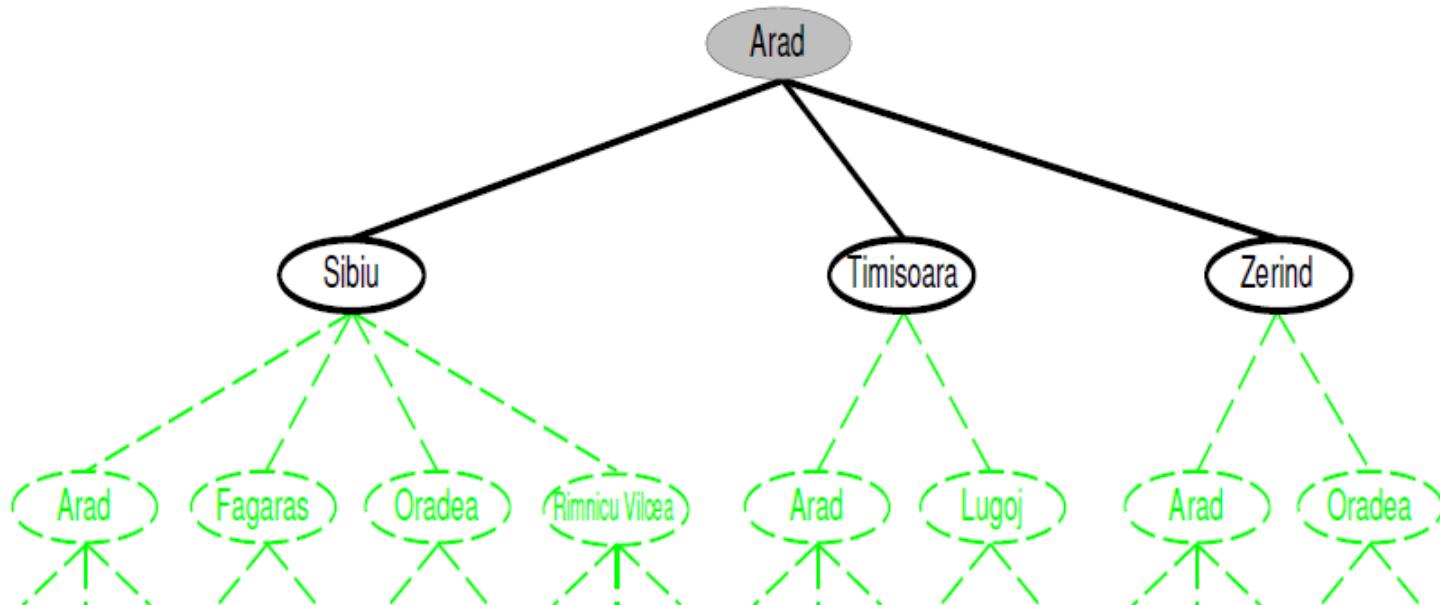
- État initial: A(Arad)
- Actions: S(Arad)= (Aller(Sibiu), Aller(Fagaras)),etc.
- Test de but: A(Bucharest)?
- Coût du chemin: longueur en kilomètres



Description du formulation du problème



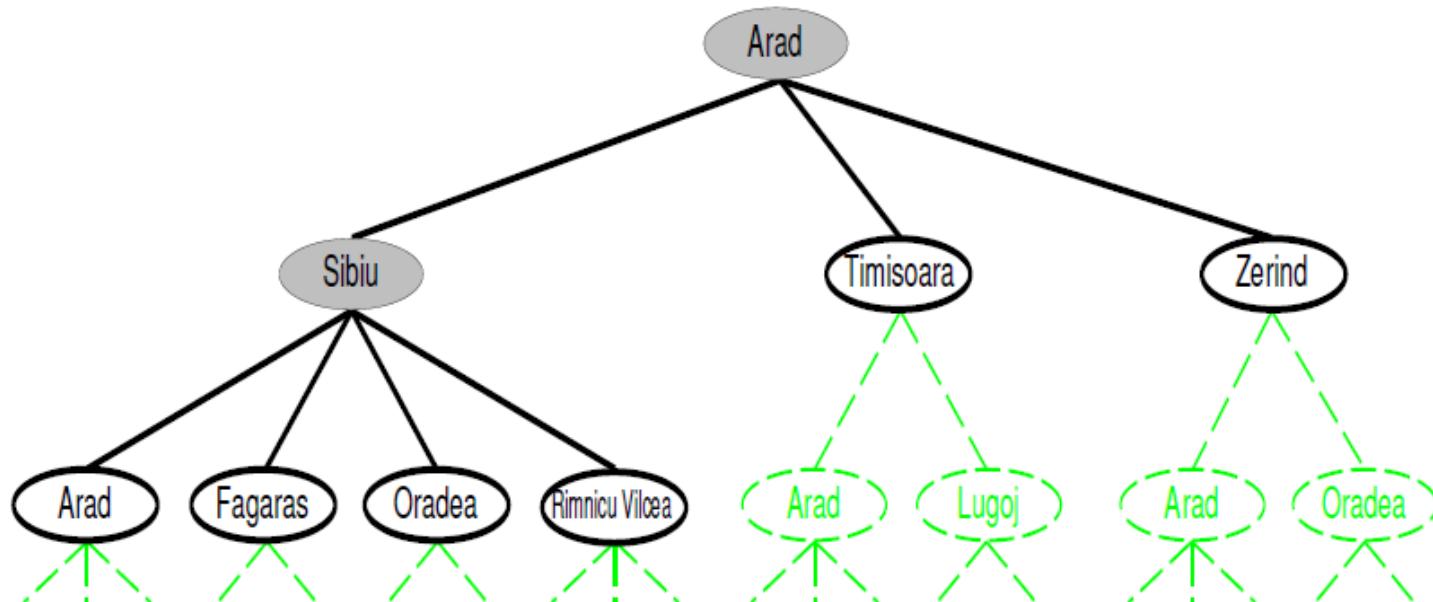
- État initial: A(Arad)
- Actions: S(Arad)= (Aller(Sibiu), Aller(Fagaras)),etc.
- Test de but: A(Bucharest)?
- Coût du chemin: longueur en kilomètres



Description du formulation du problème



- État initial: A(Arad)
- Actions: S(Arad)= (Aller(Sibiu), Aller(Fagaras)),etc.
- Test de but: A(Bucharest)?
- Coût du chemin: longueur en kilomètres



Recherche de solutions



Fondement de l'IA

2021-2022

Les algorithmes d'exploration modélisent les problèmes par des **arbres d'exploration**

État initial = racine de l'arbre

Actions = branches de l'arbre

États de l'espace d'état = nœuds de l'arbre

L'ensemble de tous les nœuds développables = **frontière**

Fonction Explorer-arbre (problème) **retourne** une solution ou échec

Initialiser la frontière avec l'état initial du problème

Faire en boucle

Si la frontière est vide alors **retourner** échec

choisir un nœud feuille et l'enlever de la frontière

si le nœud contient un état but alors **retourner** la solution correspondante

sinon développer le nœud choisi, en ajoutant les nœuds obtenus à la frontière

Comment choisir le nœud prochain?

Quelle stratégie d'exploration ou de recherche?

Stratégies de recherche



Fondement de l'IA

2021-2022

Explorent et développent les nœuds selon un ordre précis



Recherches non informées (aveugles)

- Ne sont pas capables de décider si un nœud est meilleur qu'un autre
- Elles sont seulement capables de tester si l'état est un but ou non
- Pas d'informations additionnelles

VS



Recherches informées (heuristiques)

- Estimation d'une fonction heuristique qui décide si un nœud est plus important que d'autres

En largeur d'abord

À coût uniforme

En profondeur d'abord

En profondeur limitée

Algorithme A*

Stratégies de recherche

2021-2022



1

2

3

4

Exploration en largeur d'abord BFS

Les nœuds d'un niveau sont explorés avant ceux du niveau suivant
L'arborescence est construite horizontalement niveau après niveau

Exploration à coût uniforme UCS

Exploration du nœud qui a le coût de chemin le plus faible

Exploration en profondeur d'abord DFS

Exploration du nœud le plus profond de la frontière courante de l'arbre

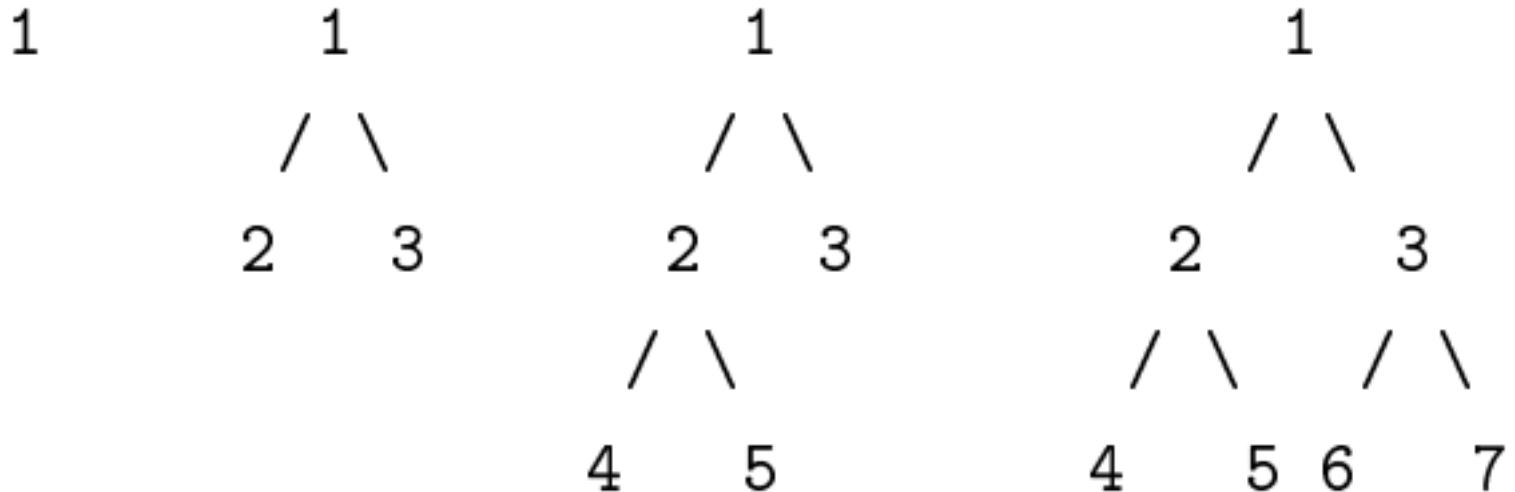
Exploration en profondeur limitée LDS

Recherche en profondeur avec une limite de profondeur d'exploration L

Recherche en largeur d'abord (BFS)



Breadth First Search



Recherche en largeur d'abord (BFS)



► Principe:

- i. Chercher le but souhaité parmi tous les nœuds d'un niveau(i) donné avant d'aller consulter leurs fils (niveau*i*+1)
- ii. Continuer, niveau par niveau jusqu'au but

Algorithme: (Représentation File)

1. Mettre la racine en tête de file
2. Jusqu'à ce que la file soit vide ou but atteint

Déterminer si le 1^{er}élément de la file est le but

- 2-a Si oui ne rien faire
- 2-b Sinon le retirer de la file et enfiler ses fils, s'il en a, de g à d

3. La recherche aboutit si nœud but a été atteint et échoue sinon



Uniform Cost Search

Problème : On veut aller de S à G en utilisant le chemin le plus court, où les distances sont données par le tableau suivant :

$$\text{dist}(S,A) = 1$$

$$\text{dist}(S,B) = 5$$

$$\text{dist}(S,C) = 15$$

$$\text{dist}(A,G) = 10$$

$$\text{dist}(B,G) = 5$$

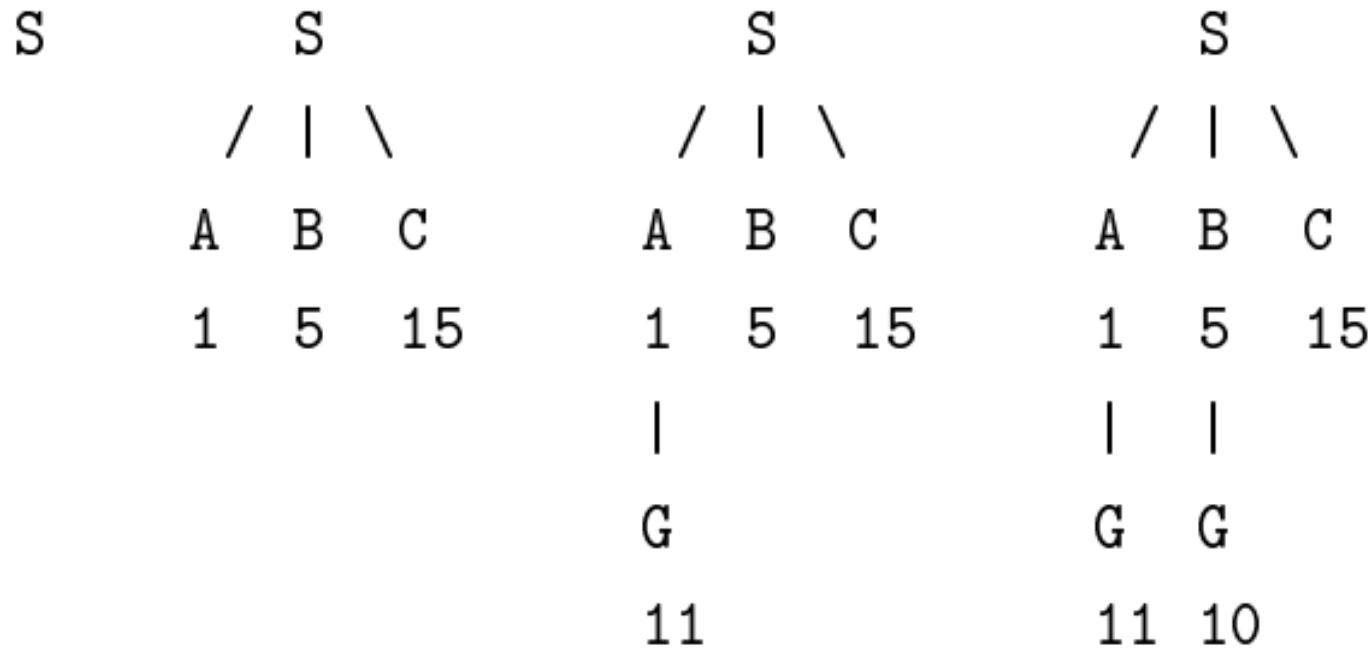
$$\text{dist}(C,G) = 5$$

Recherche à coût uniforme (UCS)



Fondement de l'IA

2021-2022



Le coût est associé à chaque arc. Si coût de chaque arc = 1,
alors recherche à coût uniforme = recherche en largeur d'abord

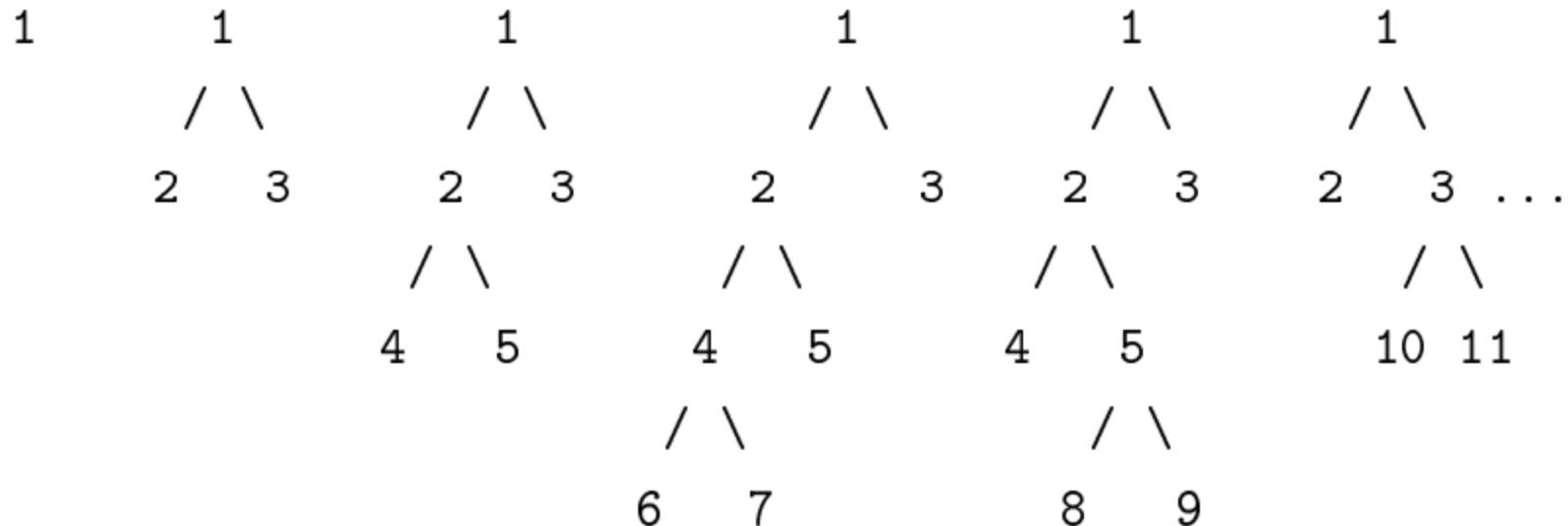
Recherche en profondeur d'abord (DFS)



Fondement de l'IA

2021-2022

Depth First Search



Recherche en profondeur d'abord (DFS)



► Principe:

- i. Sélectionner une possibilité à chaque nœud
- ii. Descendre jusqu'à ce qu'on atteigne le but ou une feuille
- iii. En cas d'impasse, reprendre la recherche de l'ancêtre le plus proche dont au moins un fils n'a pas encore été exploré

Algorithme: (Représentation Pile)

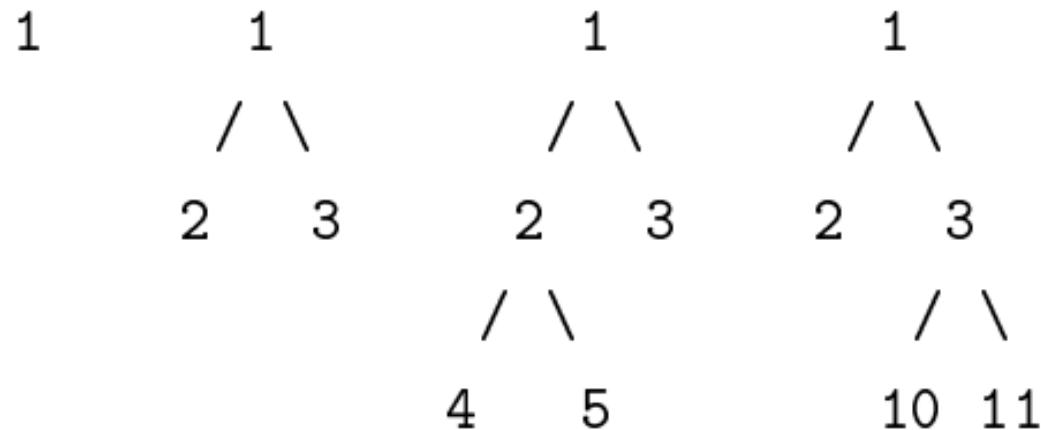
1. Empiler la racine
2. Jusqu'à ce que la pile soit vide ou nœud but soit atteint
 - Déterminer si sommet pile=but
 - 2-a Si oui ne rien faire
 - 2-b Sinon dépiler le 1^{er} élément et empiler ses fils, s'il en a, de d à g
3. La recherche aboutit si nœud but a été atteint et échoue sinon



Limited Depth Search

C'est une recherche en profondeur d'abord où l'on explore les états jusqu'à **une profondeur limitée**.

Exemple avec limite 2:



On rajoute les successeurs d'un état uniquement si on n'a pas dépassé la limite 2

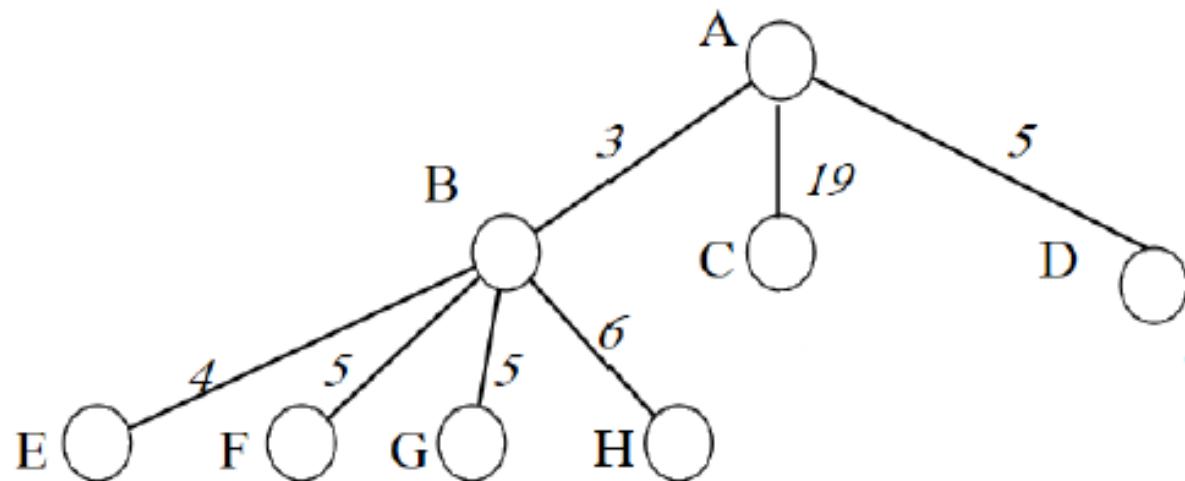
Exercice

2021-2022



Fondement de l'IA

Donner l'ordre du visite des nœuds jusqu'au nœud but G en utilisant des stratégies non informées





Fondement de l'IA

Cours: Intélligence Artificielle

2ème année informatique de gestion

Enseignante: Dr. Saoussen Mathlouthi

Saoussen.mathlouthi@ensi-uma.tn



Chapitre 3:

Recherche dans un espace d'état

Plan



Rappel

Exemples d'agents basés sur le but

Objectif de la recherche informée

Propriétés des heuristiques

Recherche meilleur d'abord (BFS)

L'algorithme A*

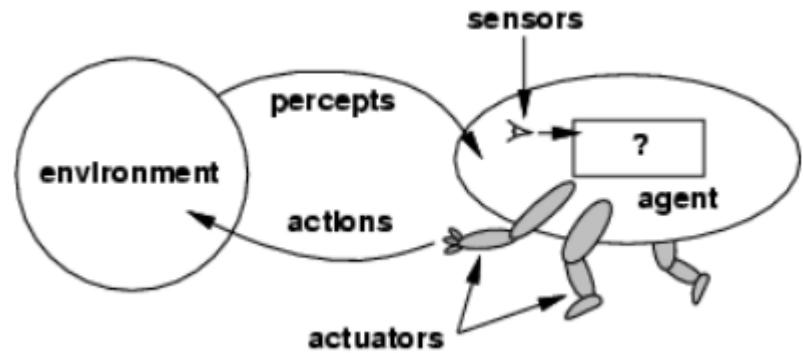
Rappel : Agents rationnels



Agent :

- Perçoit son environnement.
- Agit dans son environnement
- Se fait une représentation du monde (modèle).
- Mesure de performance.
- (Modèle PEAS).

Un agent a une fonction $f: P^* \rightarrow A$.





● Caractéristiques d'un environnement :

- Complètement observable vs partiellement observable.
- Déterministe vs stochastique.
- Épisodique vs séquentiel.
- Statique vs dynamique.
- Discret vs continu.
- Agent unique vs multi-agent.

Rappel : Résolution de problème



- La fonction **f** d'un agent peut être implémentée à l'aide du paradigme «résolution de problèmes par recherche».
- Construction d'un modèle du monde à l'aide des données sensorielles provenant des capteurs.
- Un graphe est étendu à partir de **l'état initial** (actuel) en simulant les **actions** de l'agent.
- Résoudre le problème consiste à trouver un **chemin** dans le graphe.
 - Parcours aveugles non informés (non guidés) : profondeur d'abord (**BFS**), largeur d'abord (**DFS**), coût uniforme (**UCS**), profondeur limitée (**LDS**)
 - Parcours informés (guidés) basés sur des heuristiques: Recherche meilleur en premier (**BFS**), Algorithme **A***

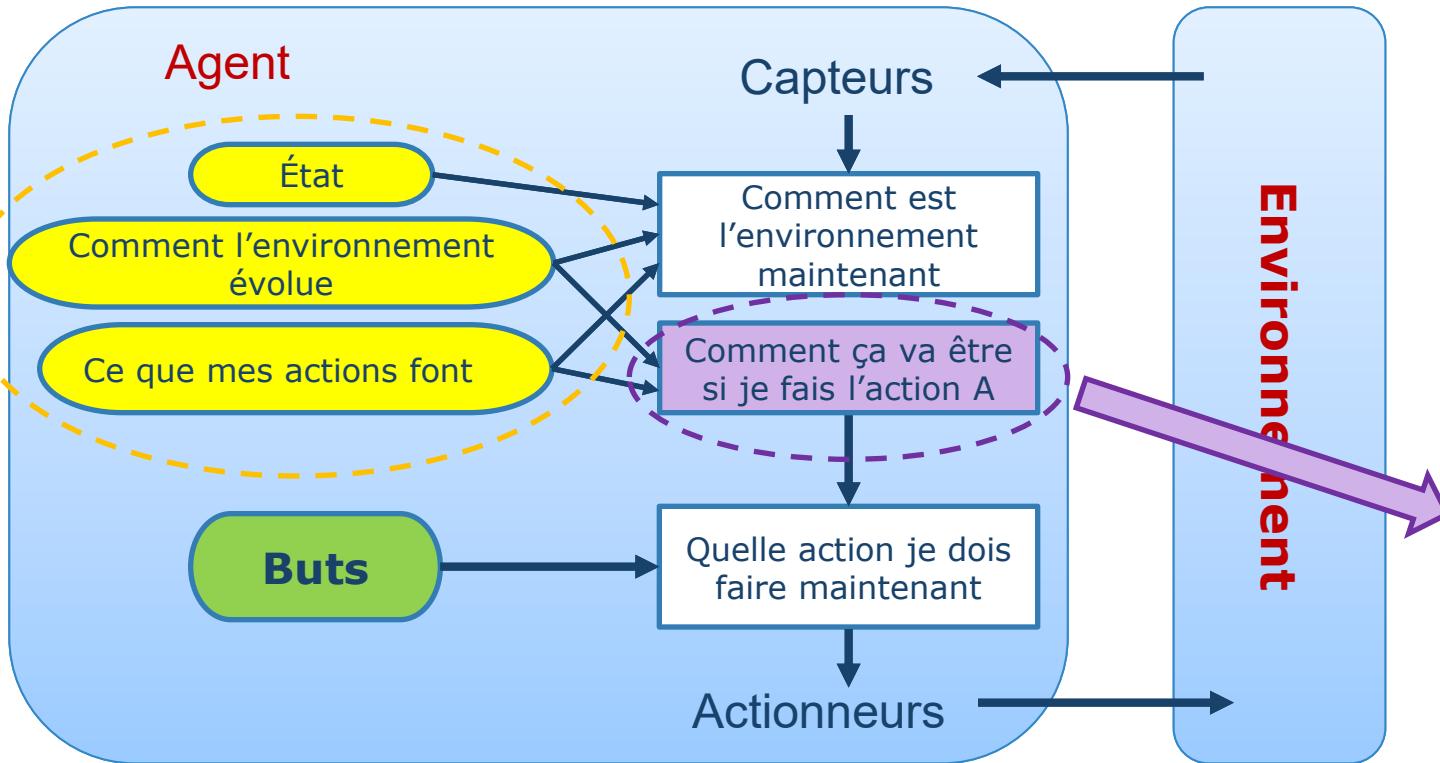
Rappel: Agent basé sur le but



Fondement de l'IA

2021-2022

□ Goal-based agents



Considération des conséquences futures de mes actions

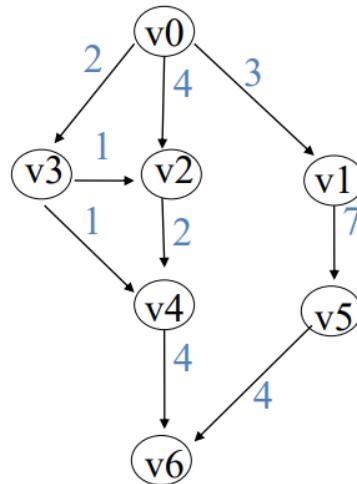
- Spécifier un but (tenir compte du futur)

Exemples d'agents basés sur le but



Exemple 1 – Agent sur une carte

- Monde: Villes et routes.
- Problème posé (état initial, but):
 - v0: ville de départ (état initial)
 - v6: destination (but)



Exemple 2 – Jeu de taquin (puzzle)

- Matrice: position des pièces.
- Problème posé (état initial, but):

Position initiale des pièces (état initial)

2	8	3
1	6	4
7		5



1	2	3
8		4
7	6	5

Configuration finale (but)



Pourquoi

- Éviter le parcours complet de l'espace d'état
- Éviter le parcours complet des actions
- Éviter l'énumération complète des solutions

Comment

- Arrêter les solutions partielles mauvaises
- Explorer d'abord les solutions partielles prometteuses
- Estimer si un nœud est plus prometteuse qu'un autre
- Utiliser un critère de choix liés au problème
- Employer le critère lorsque le coût est primordial

Propriétés des heuristiques



Choix d'un nœud nécessite une heuristique:

- Un algorithme de recherche **efficace** doit **guider** la recherche du chemin solution en faisant des choix et en gérant la révision de ces choix pour éviter l'explosion combinatoire
- On utilise des **heuristiques** pour guider ces choix en ordonnant la liste des successeurs selon leur « **promesse de rapprocher d'un but** »
- Connaissance spécifique au problème à résoudre
- **Une règle d'estimation, une stratégie, une astuce, une simplification** ou autre règle permettant de guider les choix
- Permet de détecter grâce à une fonction d'évaluation le nœud qui semble potentiellement meilleur que les autres et de se concentrer sur ce nœud par la suite en ordonnant la liste de successeurs d'un état
- A la différence des algorithmes aveugles, les heuristiques sont tirées de **l'expérience**, de **l'abstraction** ou d'un **apprentissage**.

Exemple introductif



Largeur d'abord :

Chemin : S, B, G

Coût : 8

Profondeur d'abord :

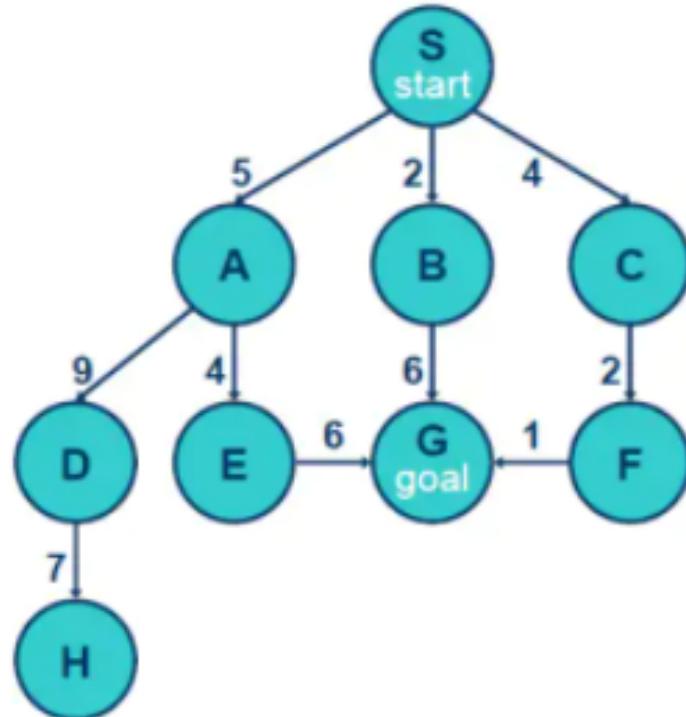
Chemin : S, A, E, G

Coût : 15

À propos ?

Chemin : S, C, F, G

Coût : 7



Recherche meilleur d'abord (Best First Search)



- Combinaison entre recherche en profondeur et en largeur
 - En **profondeur**: solution trouvée sans avoir besoin de calculer tous les nœuds
 - En **largeur**: ne risque pas de rester pris dans une impasse
- L'algorithme « recherche meilleur d'abord » permet d'explorer les nœuds dans l'ordre (croissant / décroissant) de leurs valeurs heuristiques
- Étendre le nœud le plus prometteur selon sa valeur **heuristique**

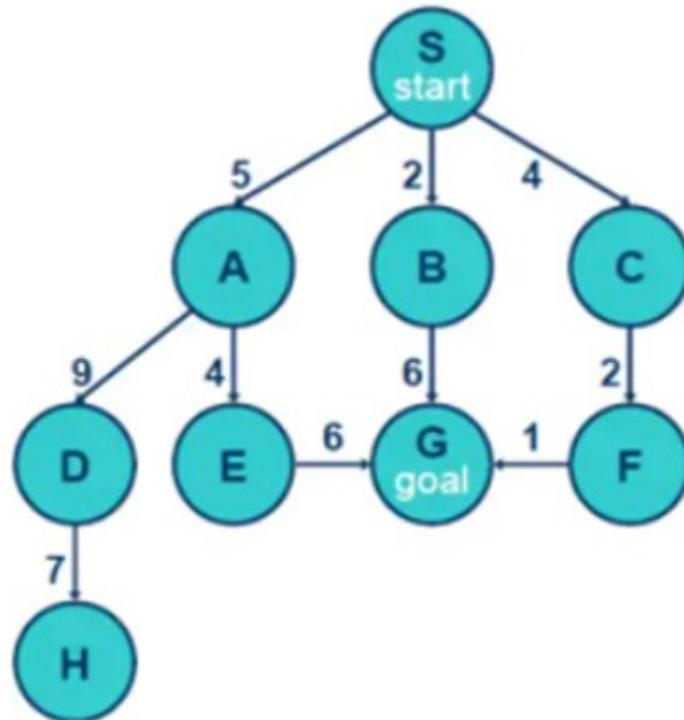
Simulation Best Fist Search



Closed list

Open list

S



Simulation Best Fist Search

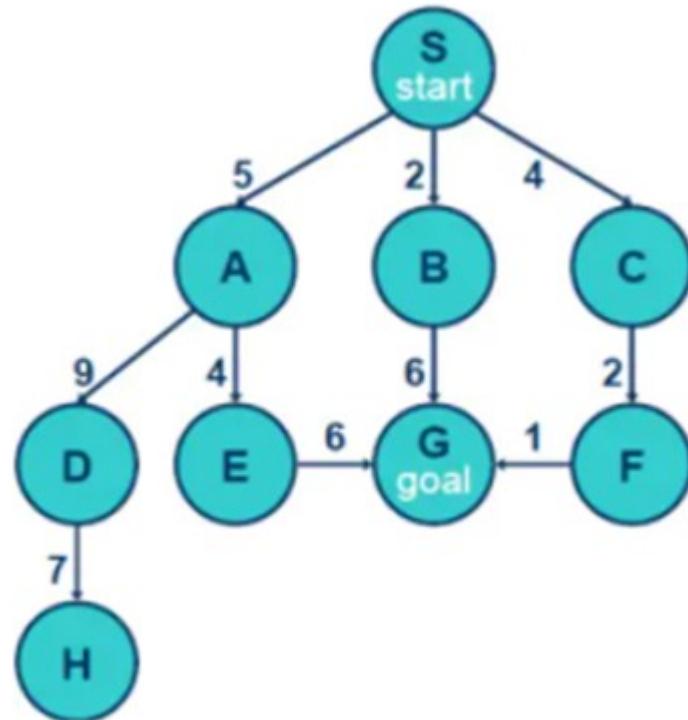


Closed list	Open list
-------------	-----------

S

S

B, S, 2
C, S, 4
A, S, 5



Simulation Best Fist Search

2021-2022

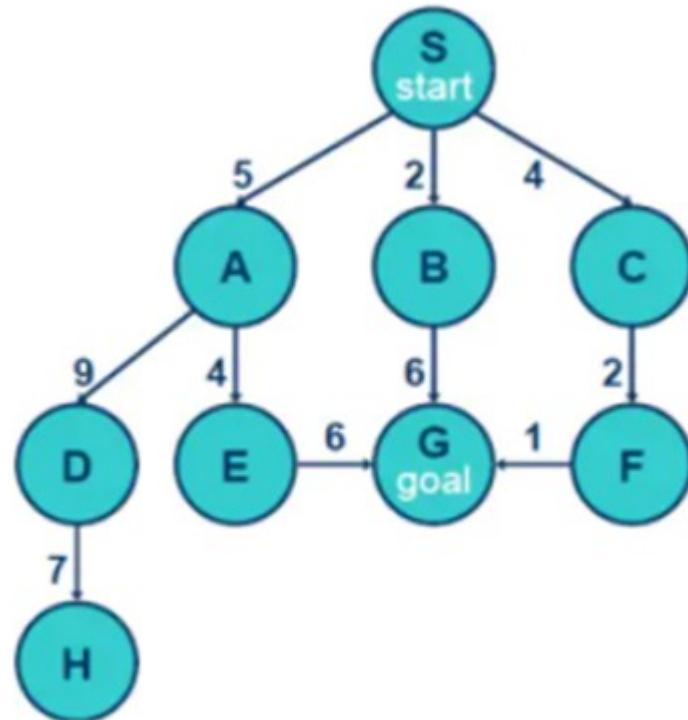


IA

Closed list	Open list
-------------	-----------

S
B, S, 2

S
B, S, 2
C, S, 4
A, S, 5
G, B, 8



Simulation Best Fist Search

2021-2022

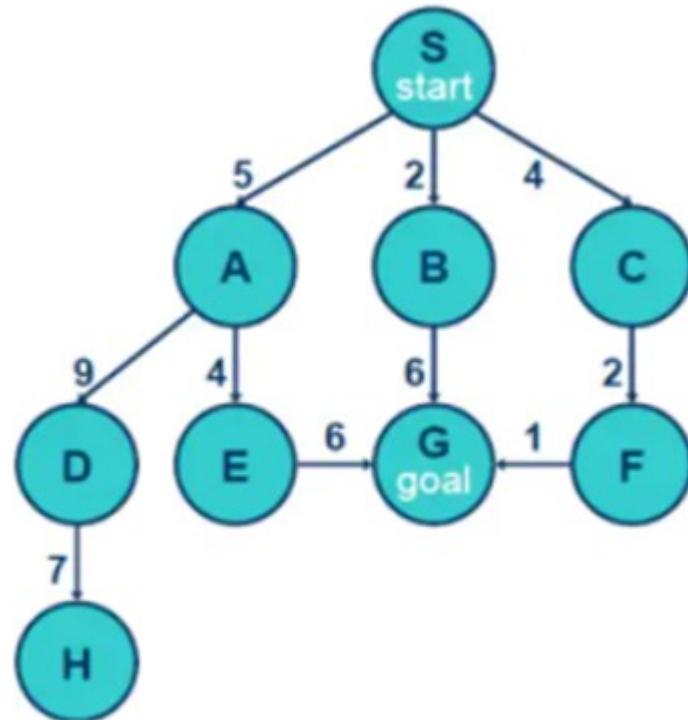


IA

Closed list	Open list
-------------	-----------

S
B, S, 2
C, S, 4

S
~~B, S, 2~~
~~C, S, 4~~
A, S, 5
G, B, 8
F, C, 6



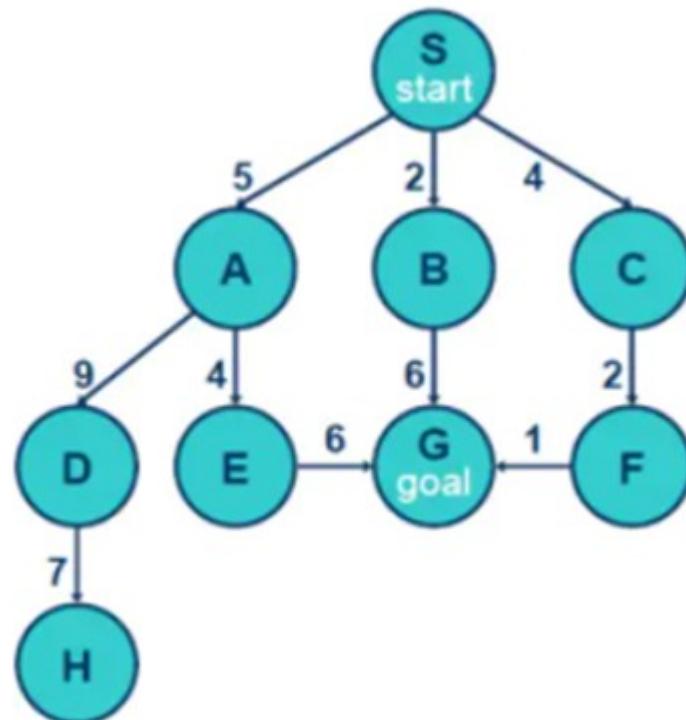
Simulation Best Fist Search



Closed list	Open list
-------------	-----------

S
B, S, 2
C, S, 4
A, S, 5

S
B, S, 2
C, S, 4
A, S, 5
G, B, 8
F, C, 6
E, A, 9
D, A, 14



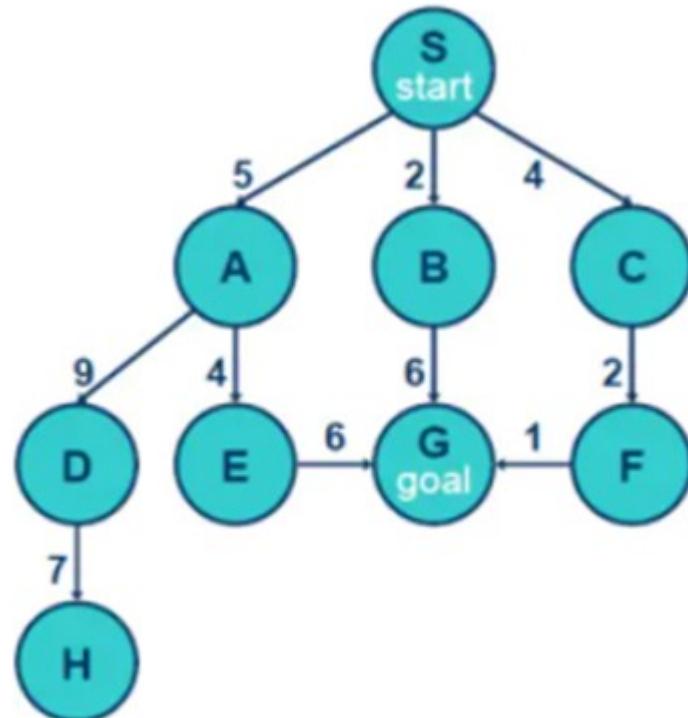
Simulation Best Fist Search



Closed list	Open list
-------------	-----------

S
B, S, 2
C, S, 4
A, S, 5
F, C, 6

S
B, S, 2
C, S, 4
A, S, 5
~~G, B, 8~~
~~F, C, 6~~
~~E, A, 9~~
~~D, A, 14~~
~~G, F, 7~~



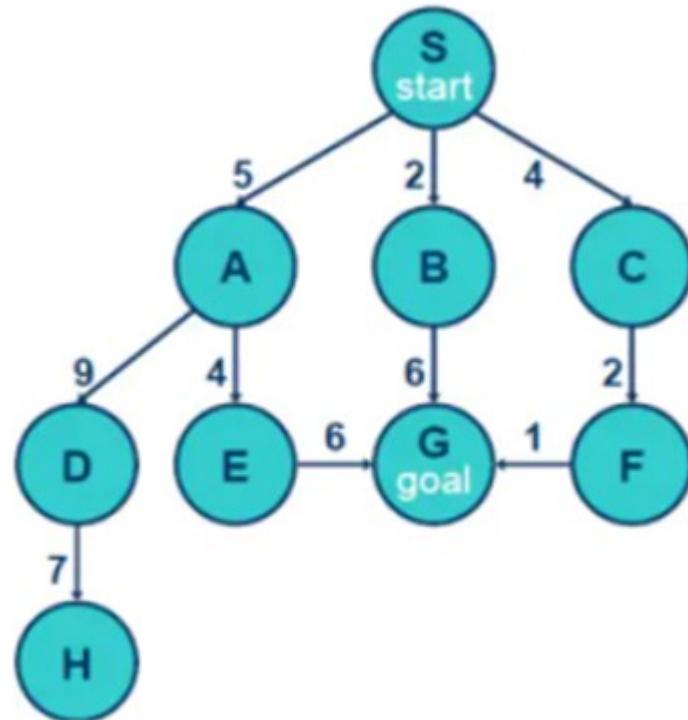
Simulation Best Fist Search



Closed list	Open list
-------------	-----------

S
B, S, 2
C, S, 4
A, S, 5
F, C, 6
G, F, 7

S
~~B, S, 2~~
~~C, S, 4~~
~~A, S, 5~~
G, B, 8
~~F, C, 6~~
E, A, 9
D, A, 14
~~G, F, 7~~



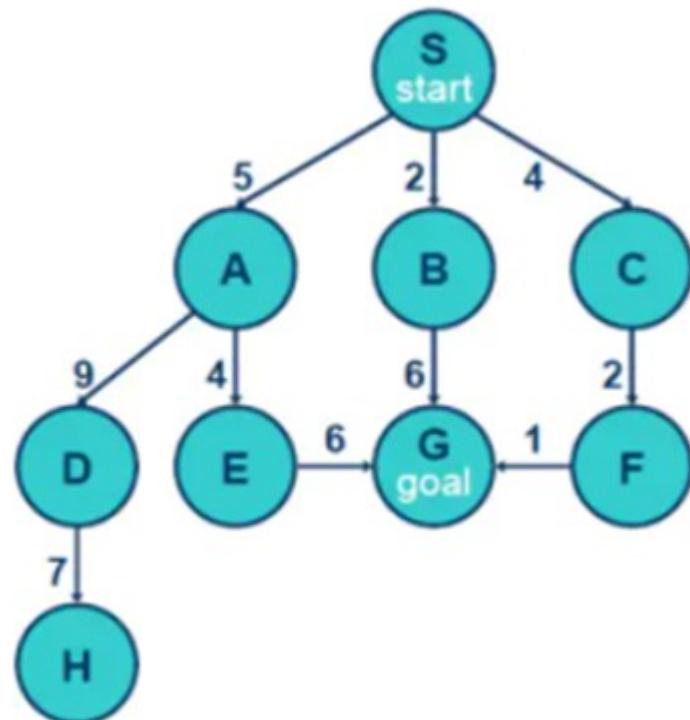
Simulation Best Fist Search

2021-2022



Fondement de l'IA

Closed list	Open list
→ S	S
→ B, S, 2	B, S, 2
→ C, S, 4	C, S, 4
A, S, 5	A, S, 5
→ F, C, 6	G, B, 8
→ G, F, 7	F, C, 6
	E, A, 9
	D, A, 14
	G, F, 7



Chemin: S, C, F, G
Coût: 7

Best Fist Search

2021-2022



- Exploiter les caractéristiques de chaque état, afin d'estimer à quel point ce nœud est prometteur
- Une fonction d'évaluation F est utilisées au cours de la recherche, et associé à chaque nœud N : $F(N)$
- Généralement, la plus petite valeur de $F(N)$ est celle la plus prometteuse,
 - Le terme « **best** » réfère à la valeur de F , et non pas à la qualité actuel traversé
 - L'algorithme « **Best First Search** » ne génère pas nécessairement le **chemin optimal**



Recherche Guidée : Définitions



- ➊ Une solution est **optimale** s'il n'existe pas aucune solution de coût strictement inférieur.
- ➋ Une méthode est **admissible** si chaque solution optimale est trouvée en un temps fini.
- ➌ Une **heuristique** est une mesure associée à un état donné qu'on notera $h(n)$.
- ➍ Une **heuristique** $h(n)$ est une **fonction d'estimation** du coût entre un nœud n d'un graphe et le but (le nœud à atteindre)

L'algorithme A* : Principe



- Utilisation d'une **heuristique** pour pendant l'exploration de l'arbre.
- Deux files sont utilisées pour stocker les nœuds visités et à visiter :
Inactif (**closed**) & Actif (**open**)
 - Parmi tous les successeurs d'un état, sont ajoutés à actifs Actif :
 - les nœuds non visités,
 - les nœuds déjà visités ayant un coût actuel moins élevé.
 - Une mesure **$f(n) = g(n) + h(n)$** est associée à chaque état n.
 - La file Actif est triée par ordre de f croissant.

L'algorithme A*

2021-2022



Variables importantes : open et closed

● **open** :

- contient les nœuds à traiter;
- c'est à dire à la frontière de la partie explorée jusqu'à présent dans le graphe.

● **closed** :

- contient les nœuds déjà traités;
- c'est à dire à l'intérieur de la frontière délimitée par open.



Insertion des nœuds dans open

- Les nœuds dans open sont ordonnés selon une estimation **f(n)** de la qualité d'une solution passant par ce nœud.
- Une **fonction f(n)** donne ou estime la qualité de la meilleure solution passant par le nœud n.
- Pour chaque nœud n, **f(n)** est un nombre réel positif ou nul, estimant le coût pour un chemin partant de l'état initial n_0 , passant par n, et arrivant dans un état n' satisfaisant le but.
- L'[open list] sera ordonnée selon la fonction d'évaluation **$f(n)=g(n) + h(n)$**
- C'est un algorithme « Best-First-search » utilisant la fonction d'évaluation **f**



Définition de la fonction $f(n)$

- La fonction **f** désigne la distance entre le nœud initial et le but.
- En pratique on ne connaît pas cette distance : c'est ce qu'on cherche !
- Par contre on connaît la distance optimale dans la partie explorée entre le nœud initial n_0 et un nœud déjà exploré.
- Ainsi, on peut séparer $f(n)$ en deux parties: $f(n) = g(n) + h(n)$
 - $g(n)$: coût réel du chemin optimal partant du nœud initial n_0 à n dans la partie déjà explorée.
 - $h(n)$: coût estimé du reste du chemin partant de n jusqu'à un état satisfaisant le but. $h(n)$ est une fonction heuristique
 - $f(n)$ est une fonction d'évaluation.

L'algorithme A*



Algorithme Recherche-Dans-Graphe(n_0)

1. **open** \leftarrow Créer un ensemble ordonnées par $f(n)$ // vide au départ
2. **closed** \leftarrow Créer un ensemble // vide au départ
3. insérer n_0 dans open
4. **while** (true) // la condition de sortie (exit) est déterminée dans la boucle
 5. si open est vide, sortir de la boucle avec échec (aucune solution n'existe)
 6. $n_1 =$ noeud au début de open avec le plus petit $f(n)$
 7. enlever n_1 de open et l'ajouter dans closed
 8. si n_1 est le but, sortir de la boucle avec succès en retournant le chemin;
 9. pour chaque noeud successeur n_2 de n_1
 10. Initialiser la valeur $g(n_2) = g(n_1) + \text{coût de la transition } (n_1, n_2)$
 11. mettre $\text{parent}(n_2) = n_1$
 12. si open ou closed contient un noeud n_3 équivalent à n_2 (même état) avec $f(n_2) < f(n_3)$, enlever n_3 de open ou closed et insérer n_2 dans open.
 13. sinon (c-à-d., n_2 n'est pas dans open ni dans closed)
 14. insérer n_2 dans open en triant les noeuds en ordre croissant selon $f(n)$



Exemples de fonctions heuristiques

- Chemin dans une ville
 - distance **Euclidienne** ou distance de **Manhattan** pour un chemin sur une carte
 - éventuellement pondéré par la qualité des routes, le prix du billet, etc.
- Probabilité d'atteindre l'objectif en passant par le nœud
- Qualité de la configuration d'un jeu par rapport à une configuration gagnante
- N-Puzzle
 - nombre de tuiles mal placées
 - somme des distances des tuiles

L'algorithme A*

IA



Exemple A* avec recherche dans une ville

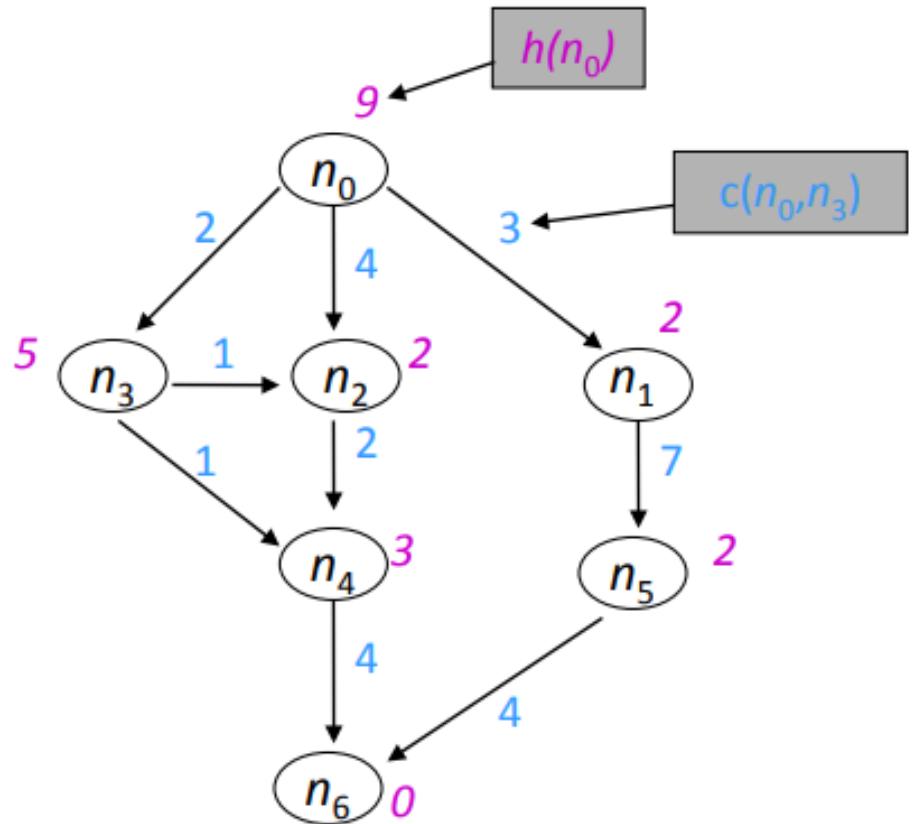
Routes entre les villes :

n_0 : ville de départ

n_6 : destination

h : distance à vol d'oiseau

c : distance réelle entre deux ville



L'algorithme A*

IA

Fondement de l'IA

2021-2022



Exemple A* avec recherche dans une ville

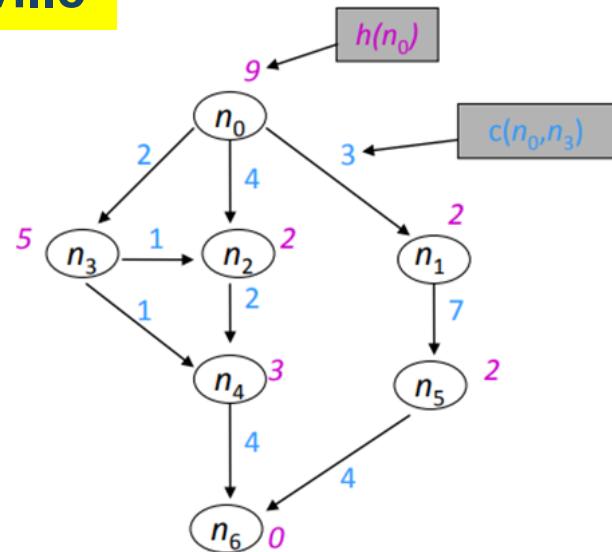
But: chemin menant à n_6

Contenu de *open* à chaque itération (état, f, parent) :

1. $(n_0, 9, \text{void})$
2. $(n_1, 5, n_0), (n_2, 6, n_0), (n_3, 7, n_0)$
3. $(n_2, 6, n_0), (n_3, 7, n_0), (n_5, 12, n_1)$
4. $(n_3, 7, n_0), (n_4, 9, n_2), (n_5, 12, n_1)$
5. $(n_2, 5, n_3), (n_4, 6, n_3), (n_5, 12, n_1)$
6. $(n_4, 6, n_3), (n_5, 12, n_1)$
7. $(n_6, 7, n_4), (n_5, 12, n_1)$
8. Solution : n_0, n_3, n_4, n_6

Contenu de *closed* à chaque itération :

1. Vide
2. $(n_0, 9, \text{void})$
3. $(n_0, 9, \text{void}), (n_1, 5, n_0)$
4. $(n_0, 9, \text{void}), (n_1, 5, n_0), (n_2, 6, n_0)$
5. $(n_0, 9, \text{void}), (n_1, 5, n_0), (n_3, 7, n_0)$
6. $(n_0, 9, \text{void}), (n_1, 5, n_0), (n_3, 7, n_0), (n_2, 5, n_3)$
7. $(n_0, 9, \text{void}), (n_1, 5, n_0), (n_3, 7, n_0), (n_2, 5, n_3), (n_4, 6, n_3)$
8. $(n_0, 9, \text{void}), (n_1, 5, n_0), (n_3, 7, n_0), (n_2, 5, n_3), (n_4, 6, n_3), (n_6, 7, n_4)$

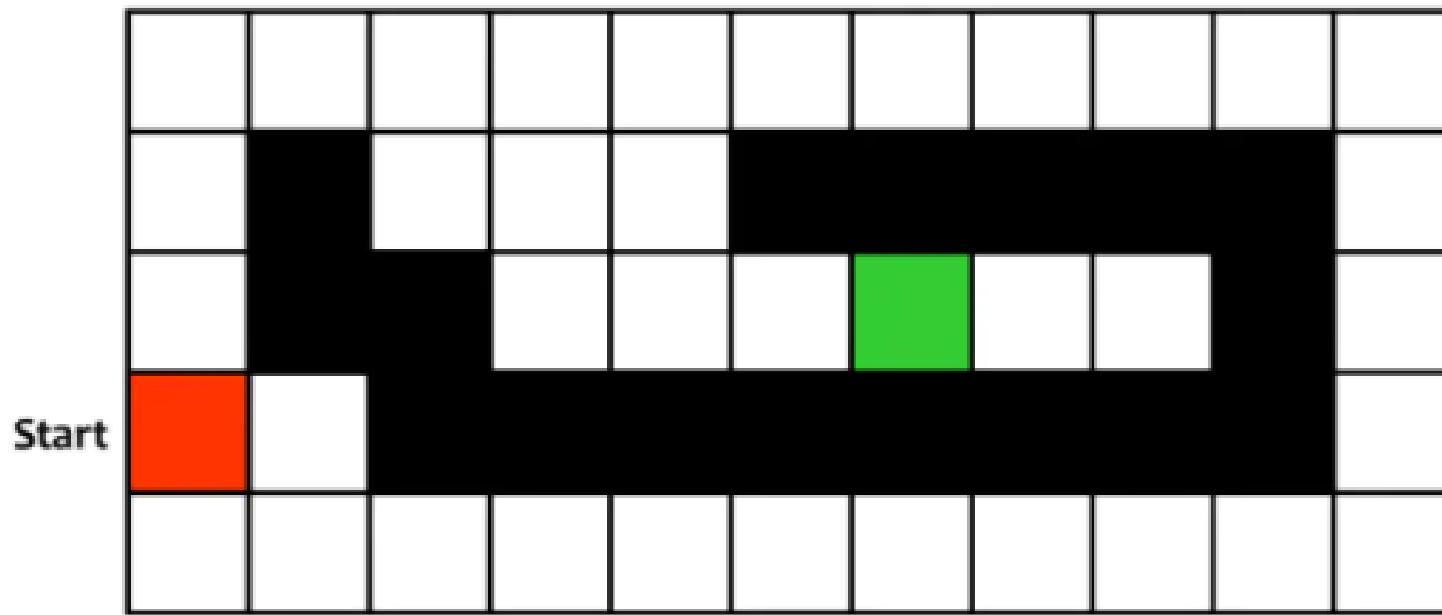


L'algorithme A*

2021-2022



Exemple A* Pour « Robot Navigation »



L'algorithme A*

IA



Exemple A* Pour « Robot Navigation »

$f(n) = h(n)$, avec $h(n)$ =distance Manhattan du nœud courant au but

8	7	6	5	4	3	2	3	4	5	6
7			5	4	3					5
6				3	2	1	0	1	2	4
Start	7	6								5
8	7	6	5	4	3	2	3	4	5	6

L'algorithme A*



Exemple A* Pour « Robot Navigation »

$f(n) = h(n)$, avec $h(n)$ =distance Manhattan du nœud courant au but

Sans l'algorithme A*

8	7	6	5	4	3	2	3	4	5	6
7		5	4	3						5
6			3	2	1	0	1	2		4
Start	7	6								5
8	7	6	5	4	3	2	3	4	5	6

L'algorithme A*

IA



Exemple A* Pour « Robot Navigation »

$f(n) = g(n) + h(n)$, avec:

$h(n)$ =distance Manhattan du nœud courant au but

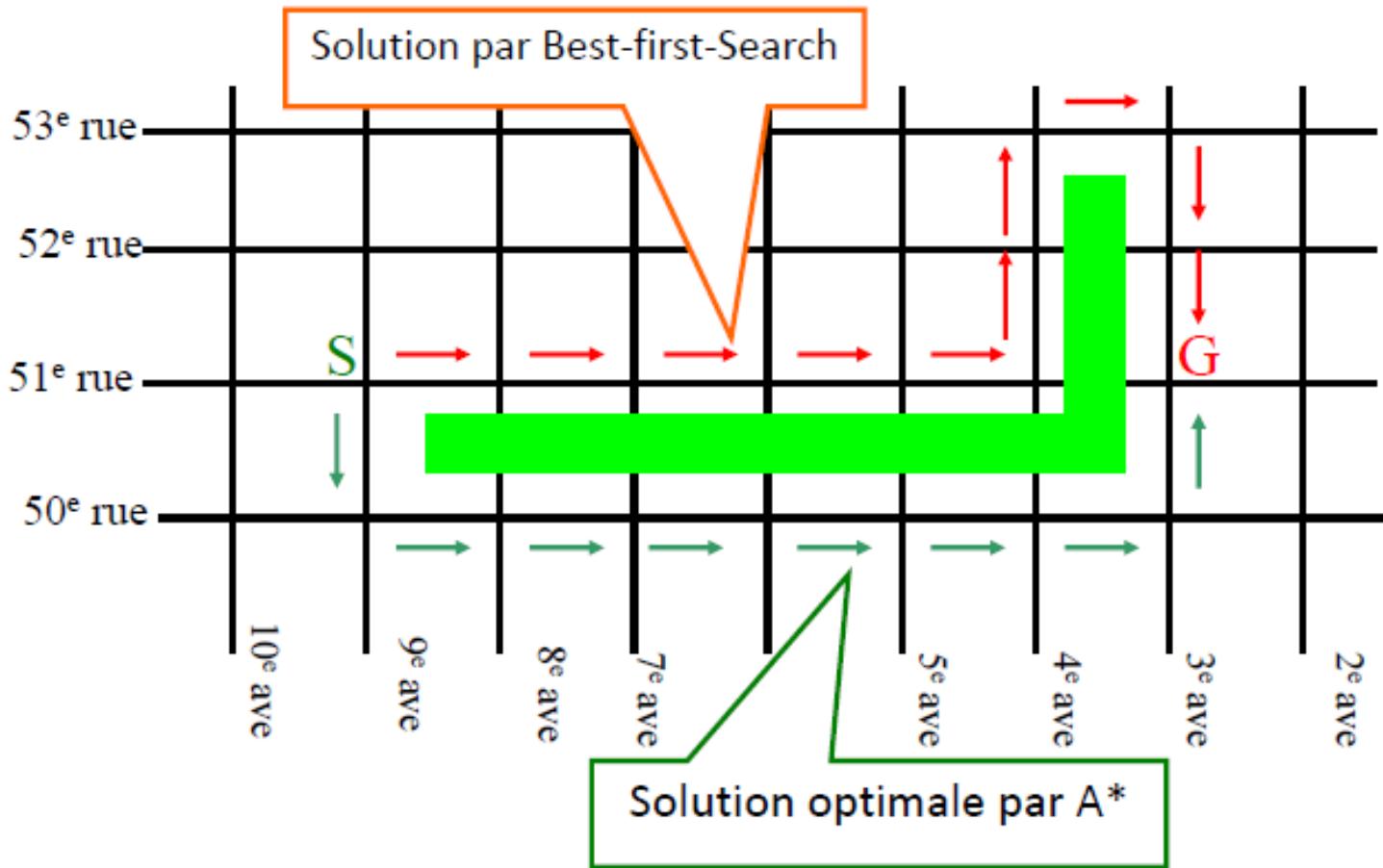
$G(n)$ = coût du chemin du nœud initial au nœud courant

Avec l'algorithme A*

8+3	7+4	6+3	5+6	4+7	3+8	2+9	3+10	4	5	6
7+2		5+6	4+7	3+8						5
6+1		3	2+9	1+10	0+11	1	2			4
Start	7+0	6+1								5
8+1	7+2	6+3	5+4	4+5	3+6	2+7	3+8	4	5	6

L'algorithme A*

IA





Fondement de l'IA

Cours: Intélligence Artificielle

2ème année informatique de gestion

Enseignante: Dr. Saoussen Mathlouthi

Saoussen.mathlouthi@ensi-uma.tn

2021-2022

Séance 9: 02/12/2021

Séance 10: 09/12/2021



Chapitre 4:

Jeux à deux adversaires

Plan



La théorie des jeux

Jeux à deux adversaires

Arbre de jeux

Algorithme MINI MAX

Élagage alpha-bêta

La théorie des jeux

Fondement de l'IA

2021-2022



- La **théorie des jeux** est un domaine des mathématiques qui s'intéresse aux interactions des choix des joueurs
- Un **jeu** est un ensemble de règles qui permettent à des joueurs en interactions de faire des choix → entraînant des **gains/pertes**
- Cette théorie considère les environnements **multi-agents** comme des jeux
- Dans les jeux, l'environnement n'est plus statique ni mono-agent le joueur adverse est un agent qui peut modifier l'environnement

Jeux à deux adversaires



- On appelle jeu à somme nulle jeu strictement compétitif, **les jeux à deux joueurs** dans lesquels l'intérêt de l'un des deux joueurs est strictement opposé à l'intérêt de l'autre joueur
- Si les récompenses des joueurs sont représentées par une **fonction de gain = une fonction d'utilité** → la somme des deux fonctions est toujours égale à 0
- On s'intéresse dans ce cours **aux jeux à somme nulle** et à **deux joueurs Max et Min** qui jouent à tour de rôle
- Nous supposons qu'à tout moment, c'est au tour d'un joueur qui peut choisir parmi les coups disponibles → Cela nous permet de présenter chaque jeu via un **arbre**

Arbre de jeux



Un problème de jeu peut être vu comme **un problème de recherche dans un arbre** :

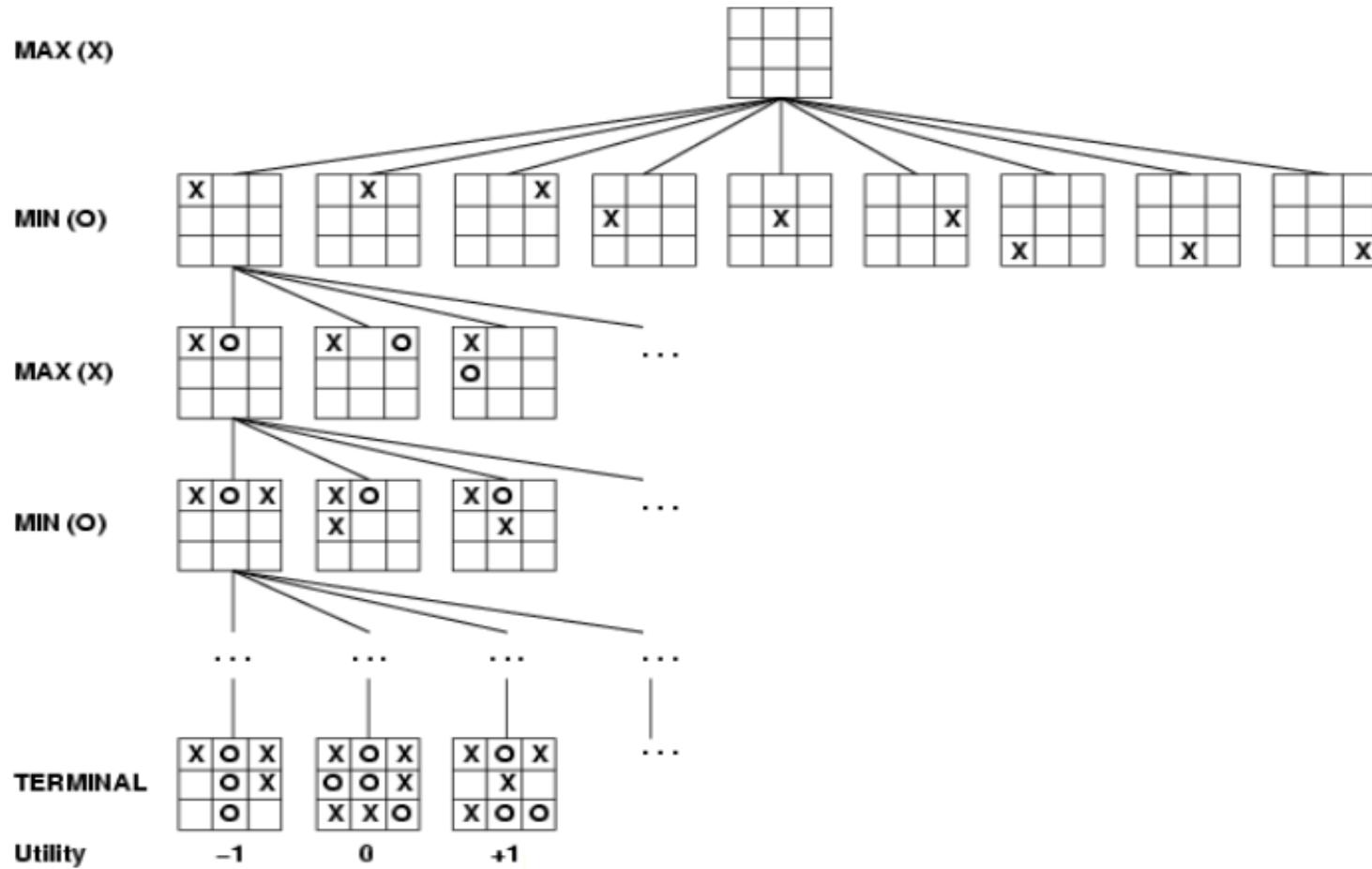
- Un **nœud (état) initial** : configuration initiale du jeu
- Une **fonction de transition** : retournant un **ensemble de paires** (action, nœud successeur)
 - action possible légale
 - nœud (état) résultant de l'exécution de cette action
- Un **test de terminaison**: indique si le jeu est terminé
- Une **fonction d'utilité** pour les états finaux (c'est la récompense reçue par le joueur Max)

Arbre de jeux

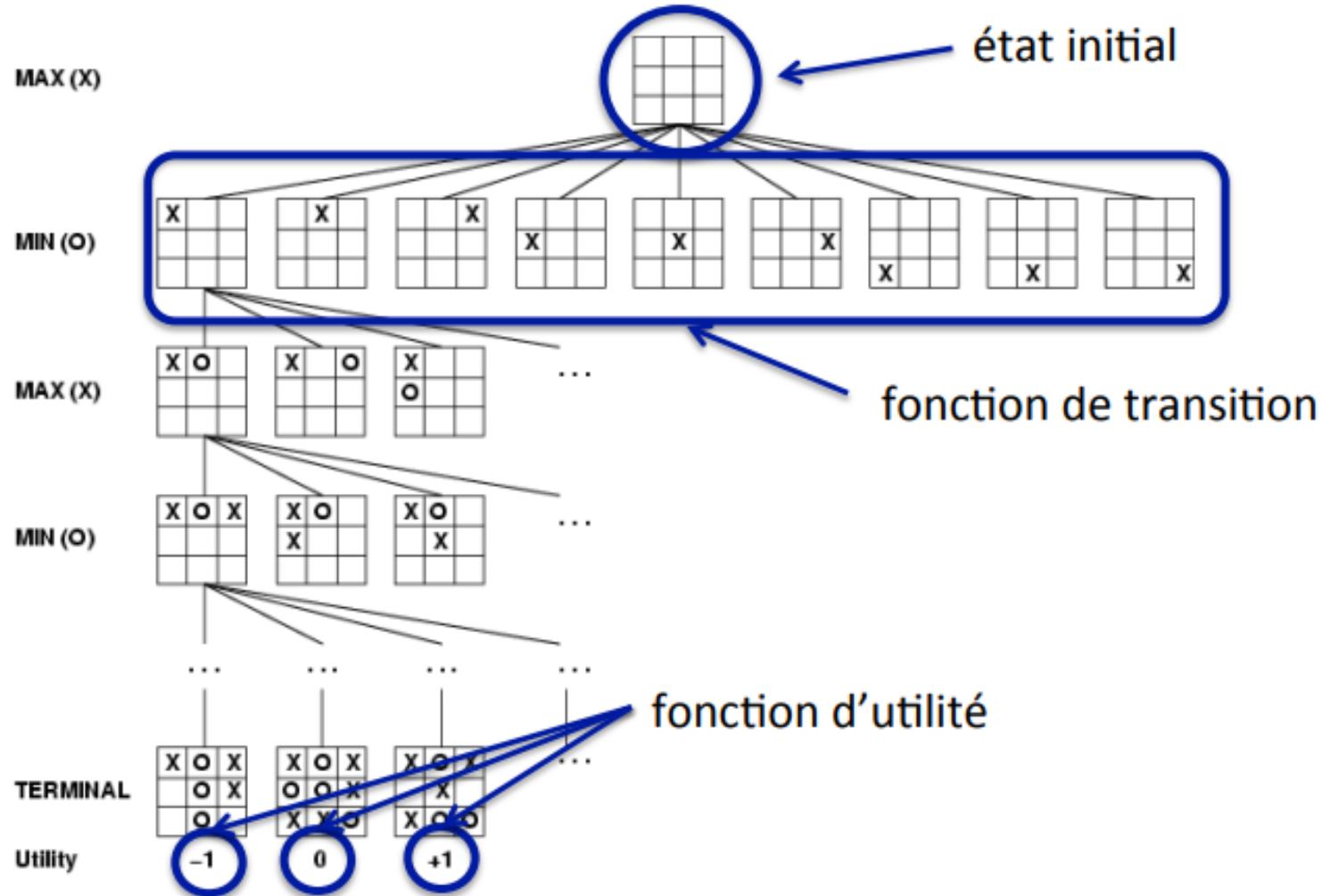


Fondement de l'IA

2021-2022



Arbre de jeux



Algorithme MINI MAX

Fondement de l'IA

2021-2022



Principe de l'algorithme: les meilleures actions pour **MAX** sont celles qui

maximise la fonction d'utilité en tenant compte que les meilleures actions

pour **MIN** sont celles **qui minimise cette même fonction**

Étapes de l'algorithme:

- Tracer l'arbre du jeux
- Évaluer les nœuds terminaux
- Partager en remontant
 - **La valeur maximale des nœuds du MAX**
 - **La valeur minimale des nœuds du MIN**

Algorithme MINI MAX

2021-2022



Fondement de l'IA

Idée: à chaque tour, on suppose que l'action la plus profitable pour le joueur Max ou le joueur Min est prise, c.-à-d. la plus grande valeur minimax

$$\text{VALEUR-MINIMAX}(n) = \begin{cases} \text{UTILITÉ}(n) & \text{Si } n \text{ est un nœud terminal} \\ \max_{n' \text{ successeur de } n} \text{VALEUR-MINIMAX}(n') & \text{Si } n \text{ est un nœud Max} \\ \min_{n' \text{ successeur de } n} \text{VALEUR-MINIMAX}(n') & \text{Si } n \text{ est un nœud Min} \end{cases}$$

Ces équations donnent un **programme récursif** pour calculer les valeurs pour tous les nœuds dans l'arbre de recherche



Algorithme MINIMAX(*noeudInitial*)

1. retourne l'action choisie par tourMax(*noeudInitial*)

Algorithme TOUR-MAX(*n*)

1. si *n* correspond à une fin de partie, alors retourner UTILITÉ(*n*)
2. $u = -\infty$, $a = \text{void}$
3. pour chaque paire (a', n') donnée par TRANSITION(*n*)
 4. si l'utilité de TOUR-MIN(*n'*) $> u$ alors affecter $a = a'$, $u = \text{utilité de TOUR-MIN}(n')$
5. retourne l'utilité *u* et l'action *a* // *L'action ne sert qu'à la fin (la racine)*

Algorithme TOUR-MIN(*n*)

1. si *n* correspond à une fin de partie, alors retourner UTILITÉ(*n*)
2. $u = \infty$, $a = \text{void}$
3. pour chaque paire (a', n') donnée par TRANSITION(*n*)
 4. si l'utilité de TOUR-MAX(*n'*) $< u$ alors affecter $a = a'$, $u = \text{utilité de TOUR-MAX}(n')$
5. retourne l'utilité *u* et l'action *a*

Algorithme MINI MAX



Soit J_1 et J_2 les deux joueurs et l'on se place dans la situation c'est à J_1 de jouer.

une situation gagnante pour J_1 vaut +1 ;

une situation perdante pour J_1 vaut -1 ;

une situation nulle vaut 0.

L'idée est de l'algorithme: est de développer complètement l'arbre de jeu, de noter chaque feuille avec sa valeur, puis de faire remonter ces valeurs avec l'hypothèse que chaque joueur choisit le meilleur coup pour lui.

Cela signifie en pratique que :

J_1 choisit le coup amenant à l'état de *plus grande* valeur ;

J_2 choisit le coup amenant à l'état de *plus petite* valeur.

Algorithme MINI MAX: Exemple 1

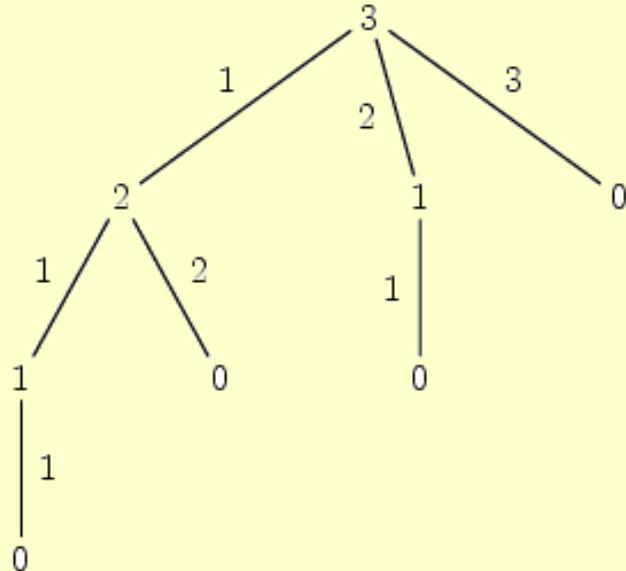


Fondement de l'IA

2021-2022

On applique le principe du **Min-Max** sur **le Jeu de Nim à 3 allumettes** (chaque joueur peut à chaque coup prendre 1, 2 ou 3 allumettes et celui qui prend la dernière allumette a perdu).

1) On commence par développer l'arbre de jeu :



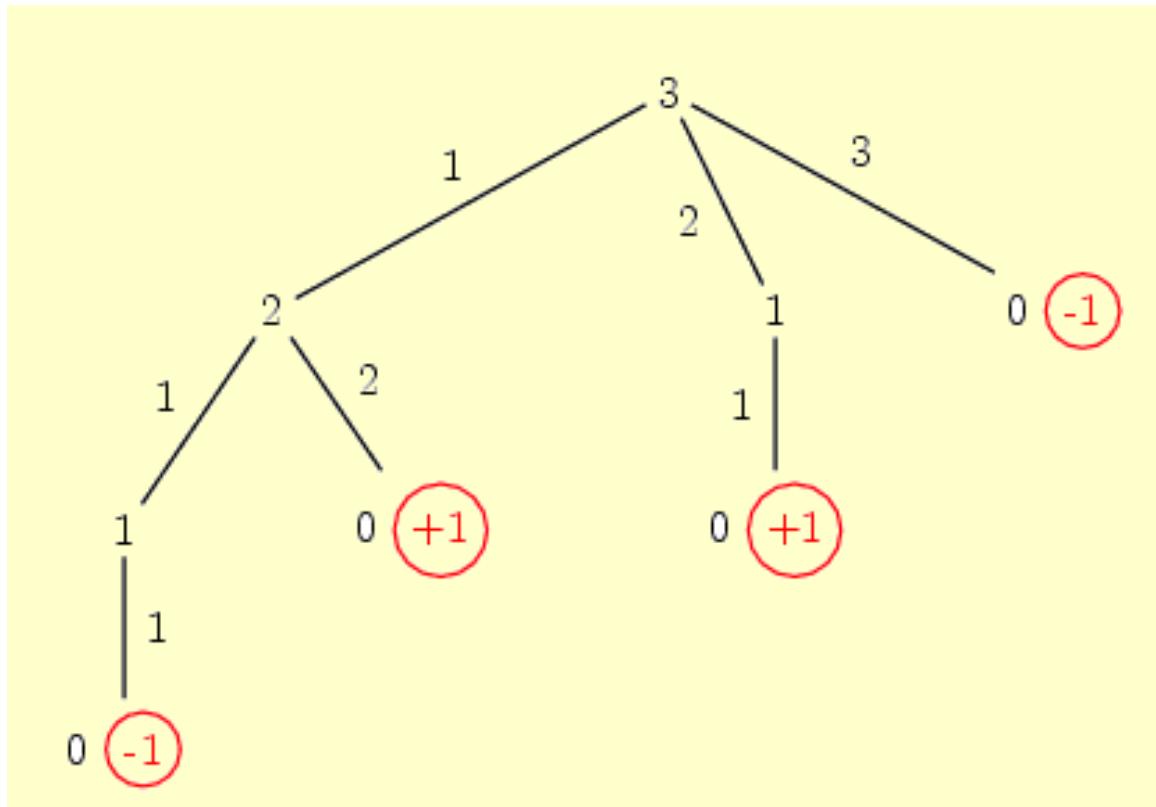
Algorithme MINI MAX: Exemple 1

2021-2022



Fondement de l'IA

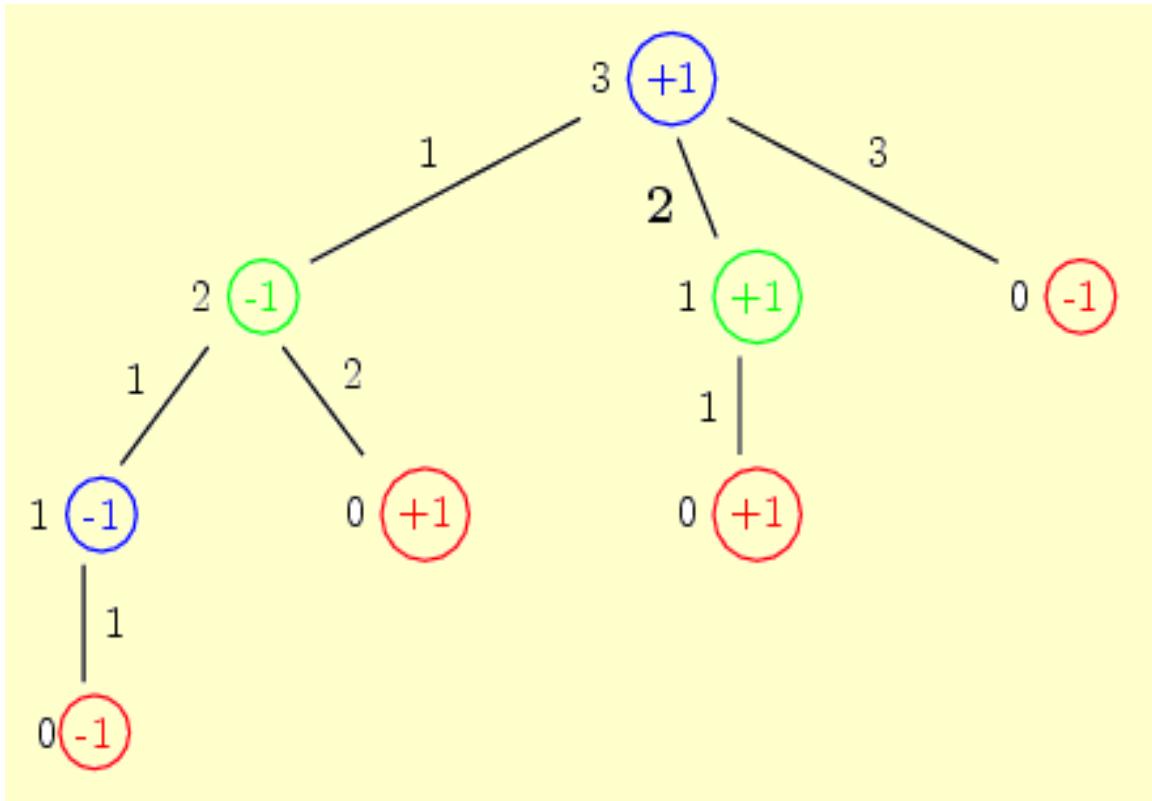
2) Puis on étiquette les feuilles selon qu'il s'agit d'une partie gagnante ou perdante :



Algorithme MINI MAX: Exemple 1



2) Enfin, on fait remonter les évaluations depuis les feuilles jusqu'à la racine (en bleu les nœuds où l'on maximise, en vert les nœuds où l'on minimise) :

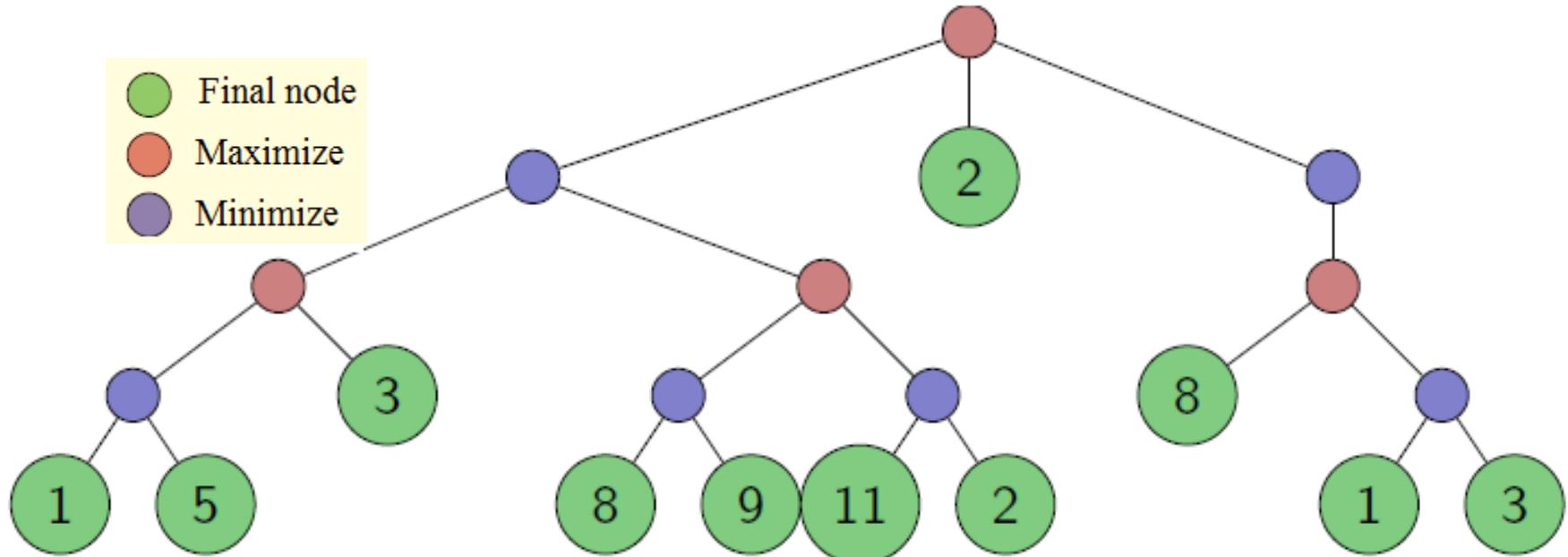


- On conclut que le premier joueur doit enlever deux allumettes.

Algorithme MINI MAX: Exemple 2



Remonter les évaluations depuis les feuilles jusqu'à la racine



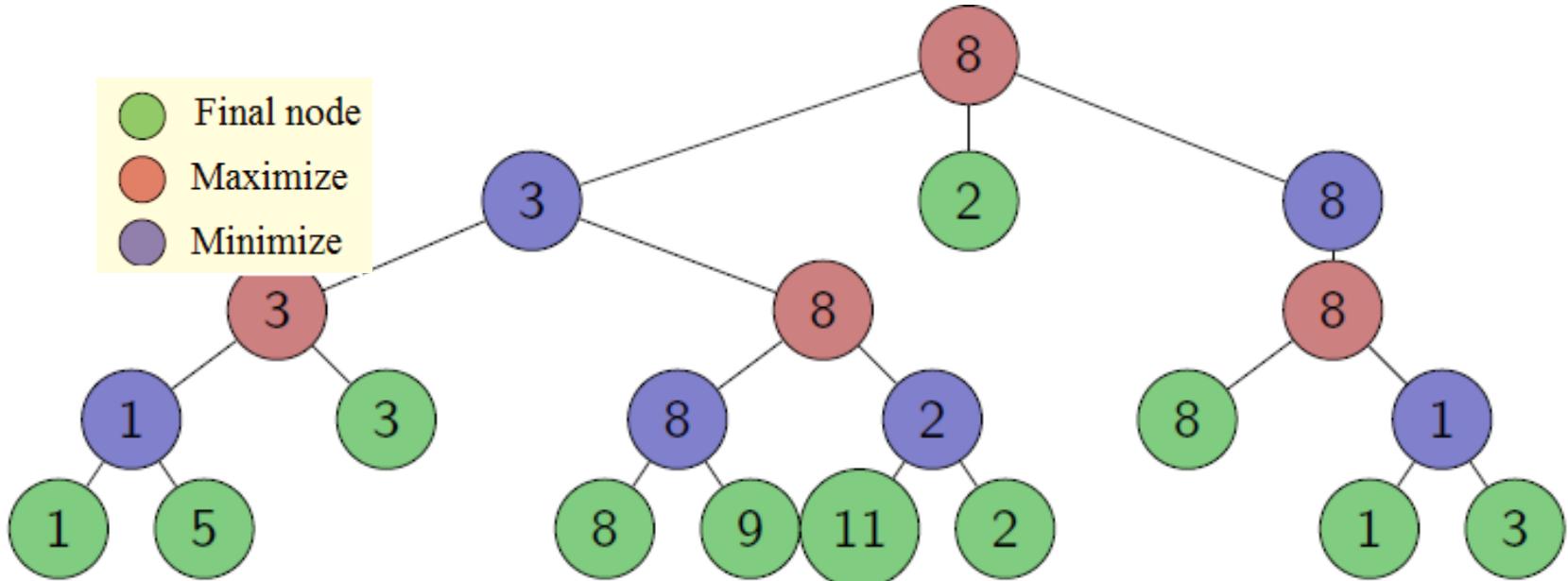
Algorithme MINI MAX: Exemple 2

2021-2022



IA

Conclure le meilleur chemin



Algorithme MINI MAX: Exemple 3

2021-2022



En 1997, Deep-Blue bat Kasparov

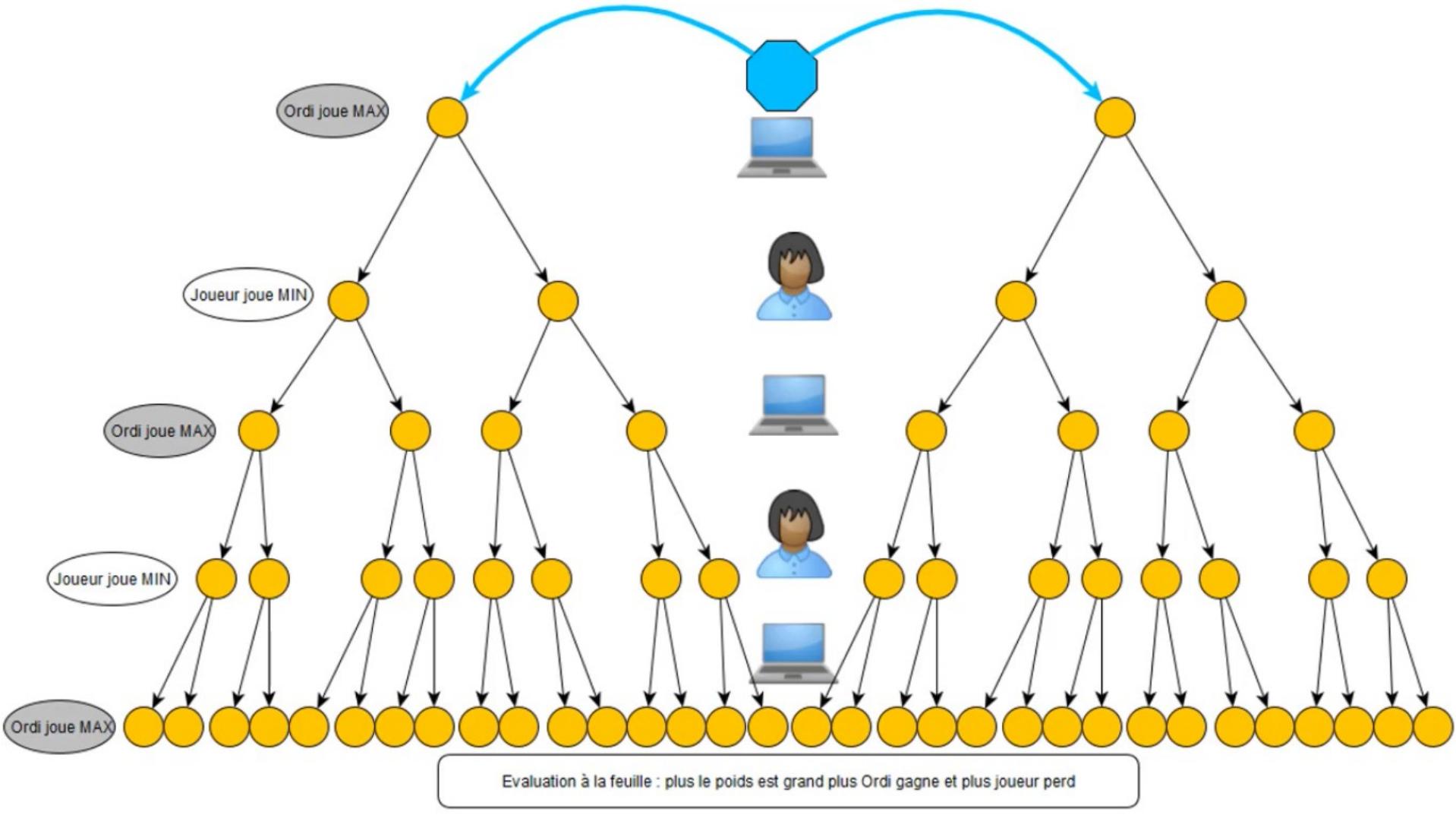
Un pas en avant énorme pour l'intelligence artificielle

Solution basée sur **l'algorithme Min Max**

Exemple de simulation du jeu d'échec avec l'algorithme Mini Max:

- Soit un arbre de recherche binaire (à chaque coup il y a deux choix)
- Soit une profondeur = 5

Algorithme MINI MAX: Exemple 3

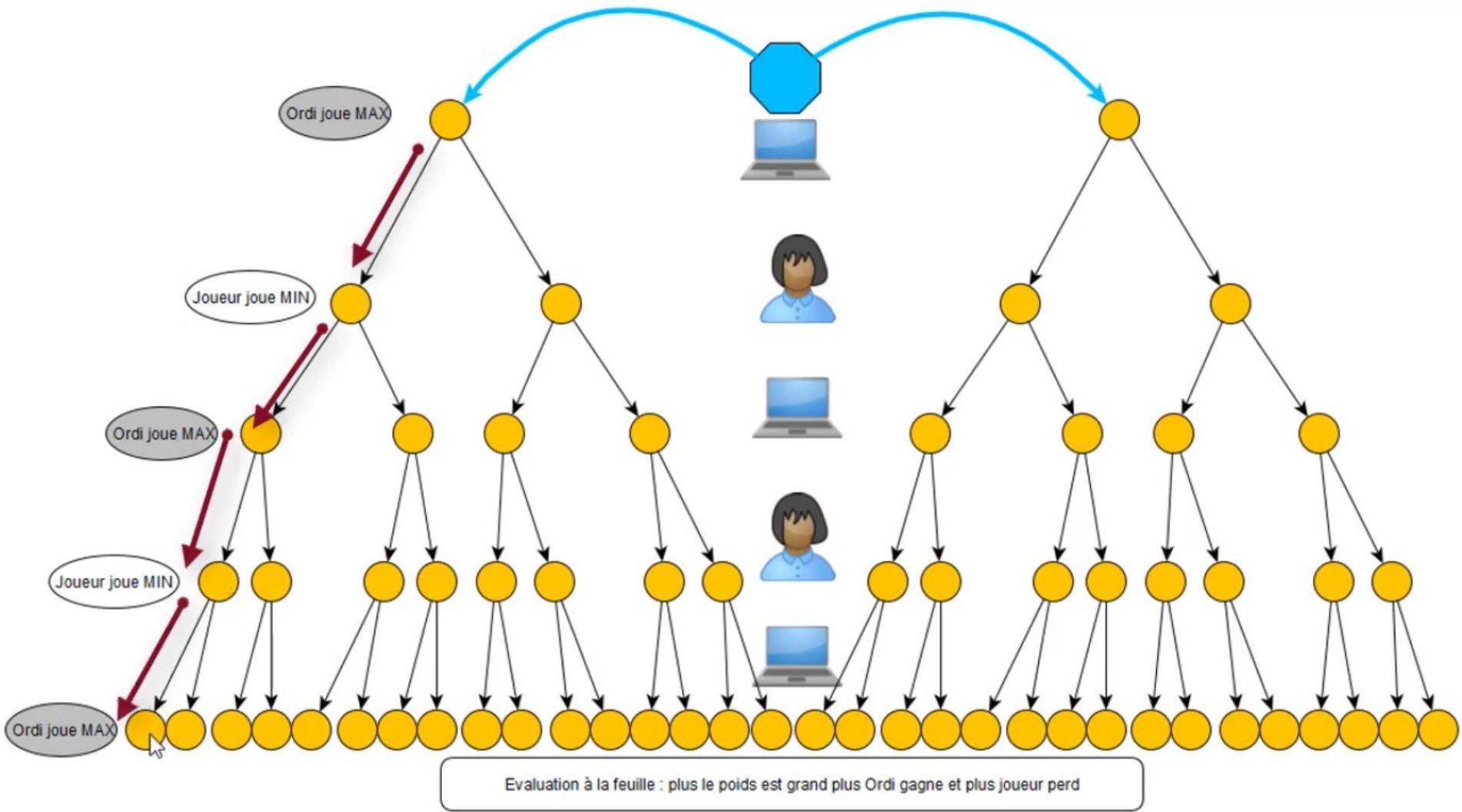


Algorithme MINI MAX: Exemple 3

2021-2022



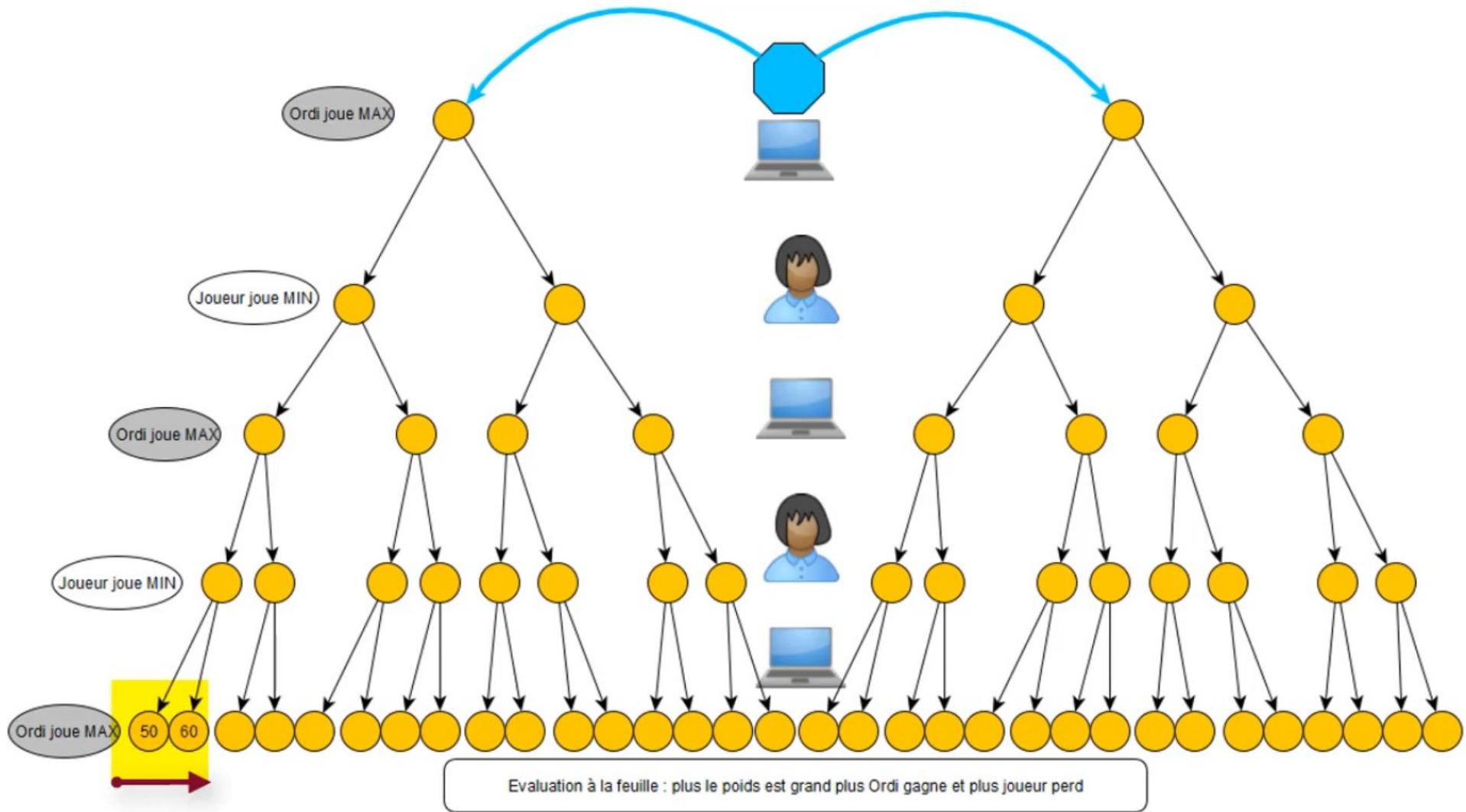
Fondement de l'IA



Algorithme MINI MAX: Exemple 3



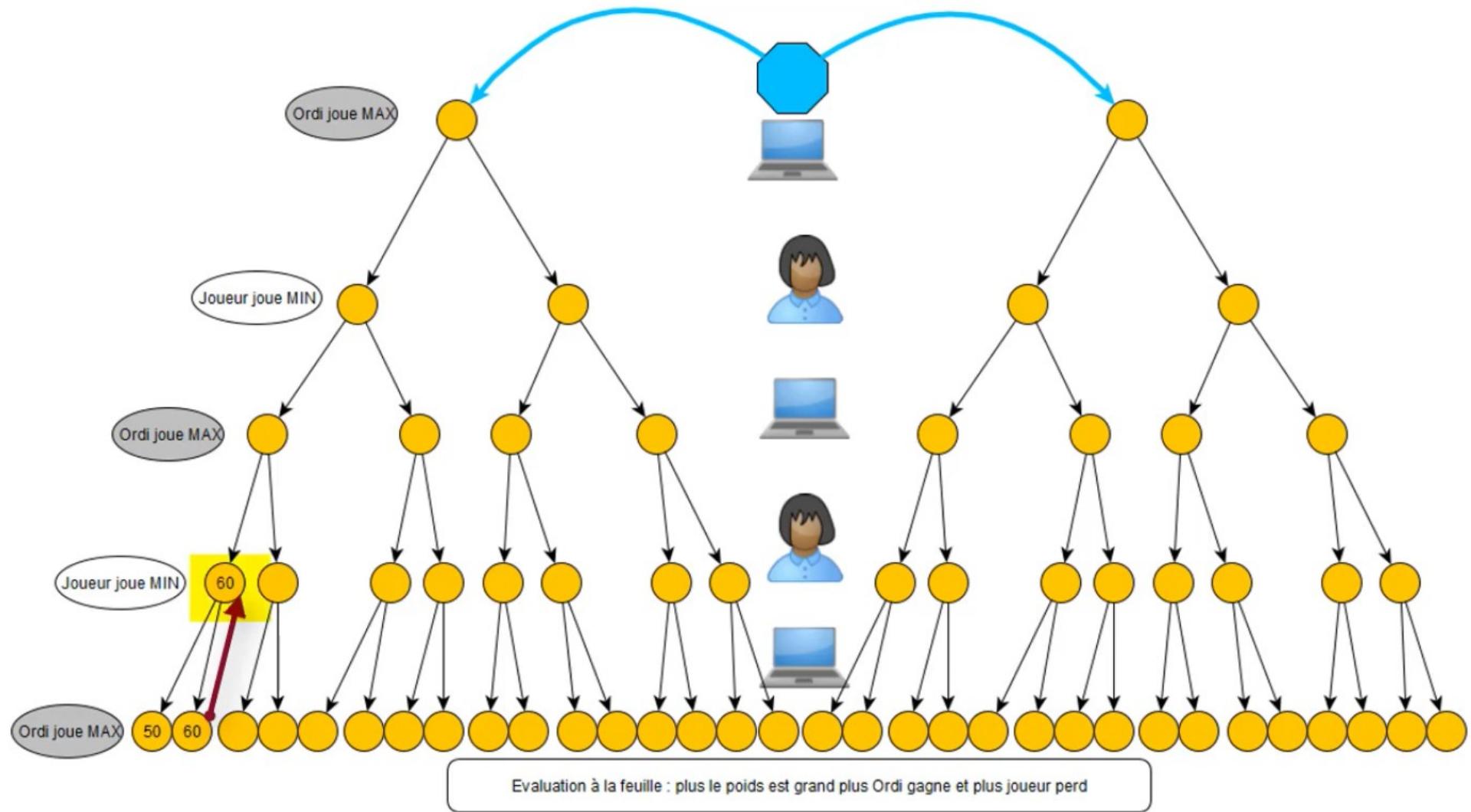
IA



Algorithme MINI MAX: Exemple 3

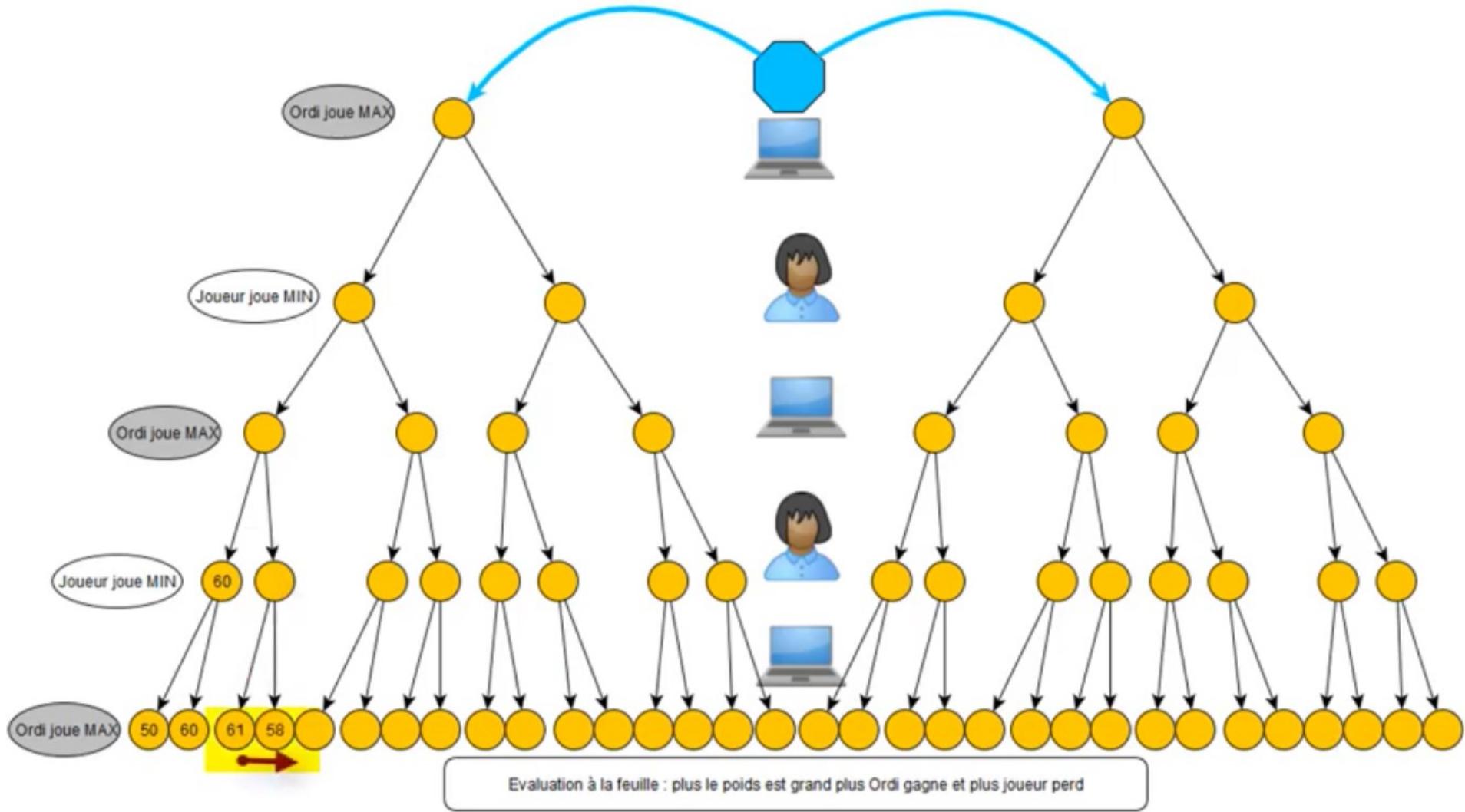
Fondement de l'IA

2021-2022



Evaluation à la feuille : plus le poids est grand plus Ordi gagne et plus joueur perd

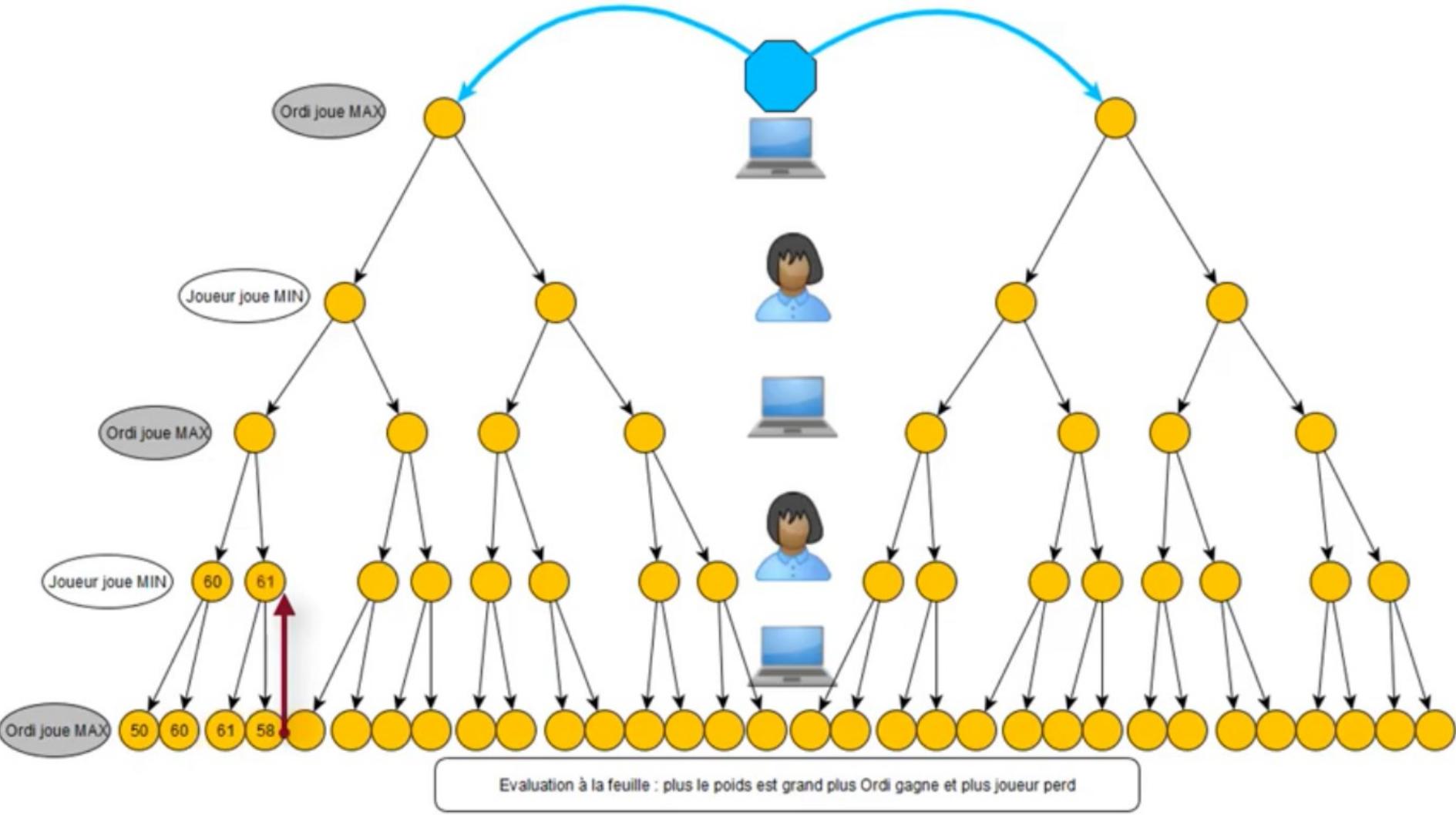
Algorithme MINI MAX: Exemple 3



Algorithme MINI MAX: Exemple 3



IA

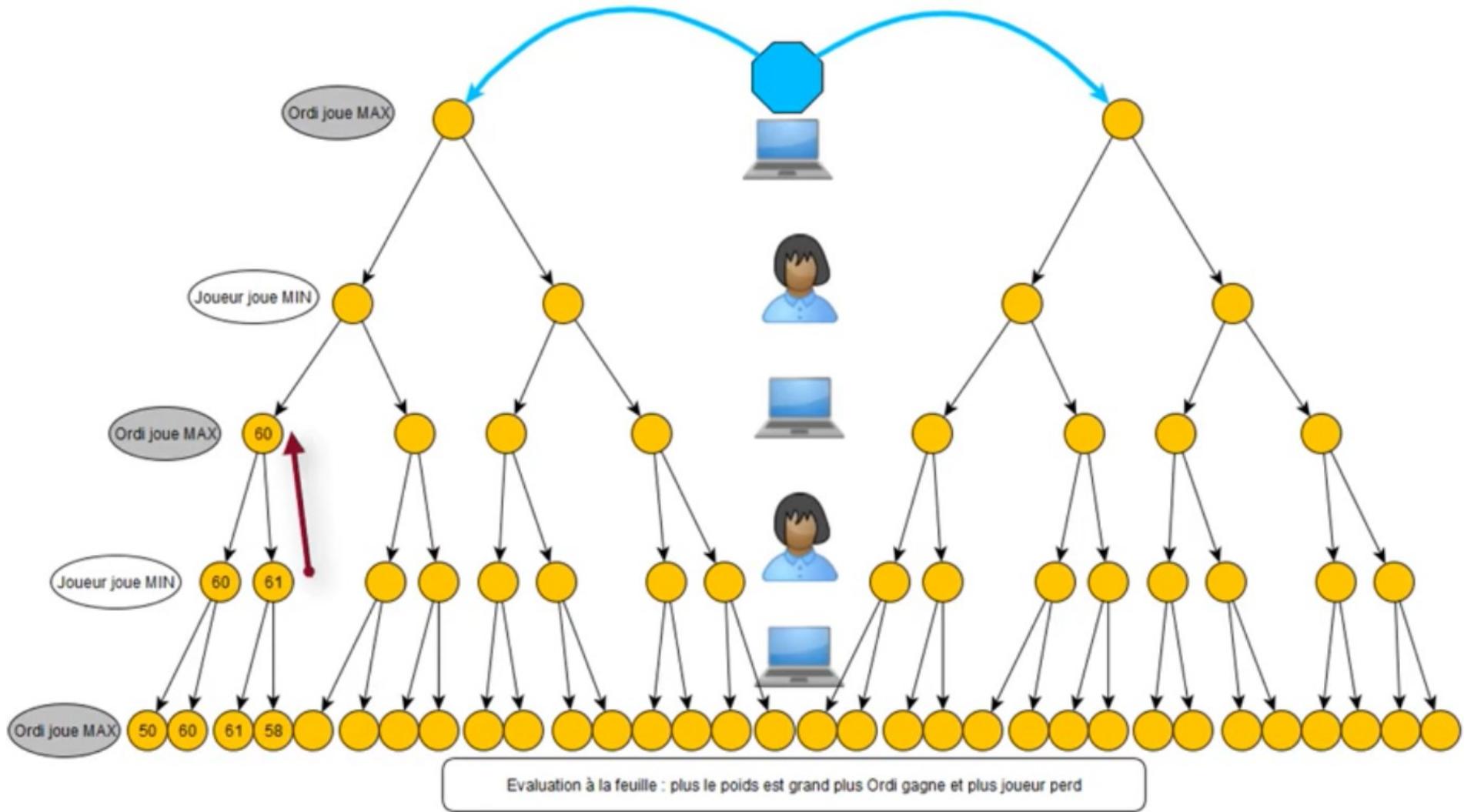


Evaluation à la feuille : plus le poids est grand plus Ordi gagne et plus joueur perd

Algorithme MINI MAX: Exemple 3



IA

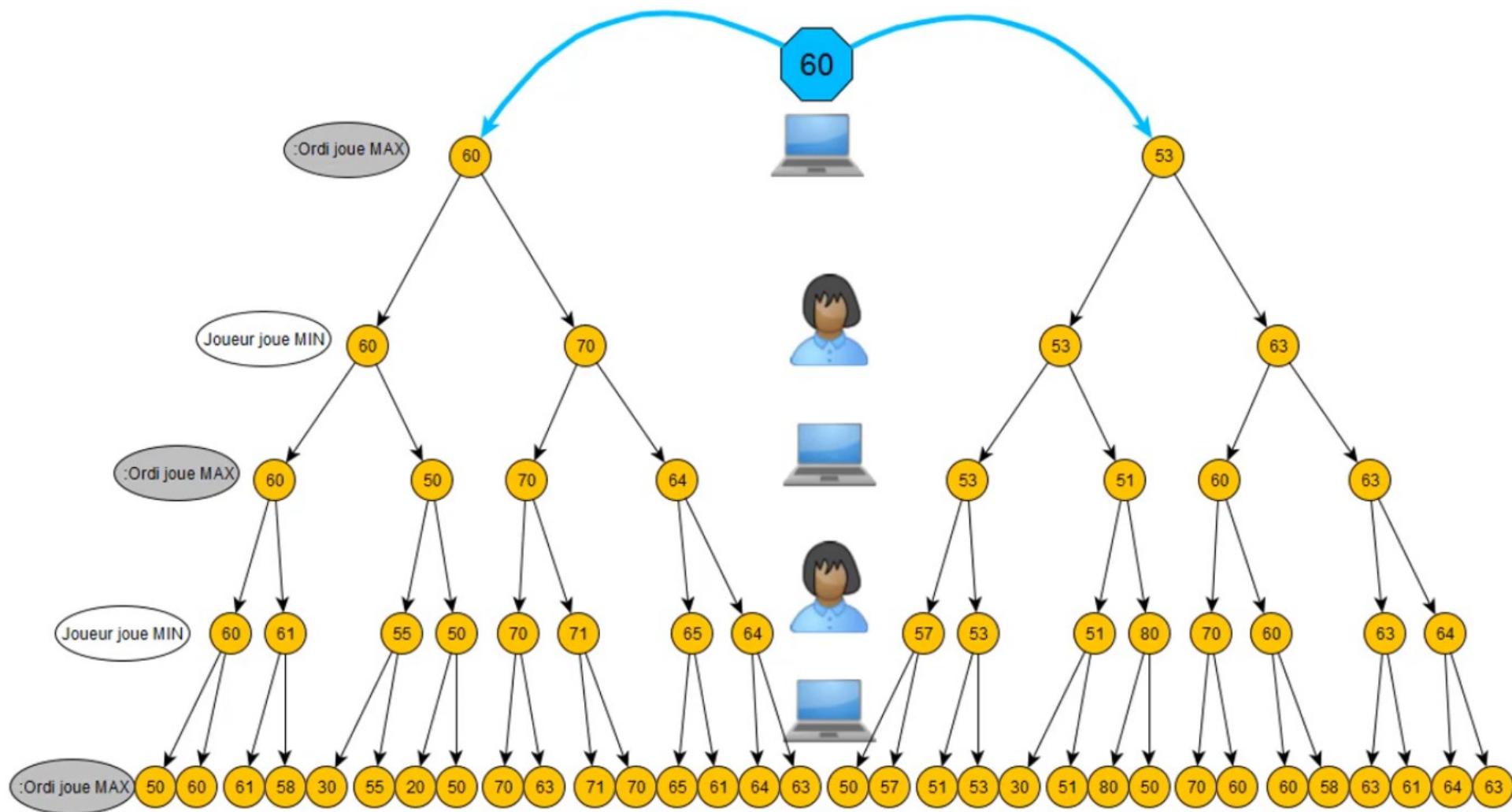


Algorithme MINI MAX: Exemple 3



Fondement de l'IA

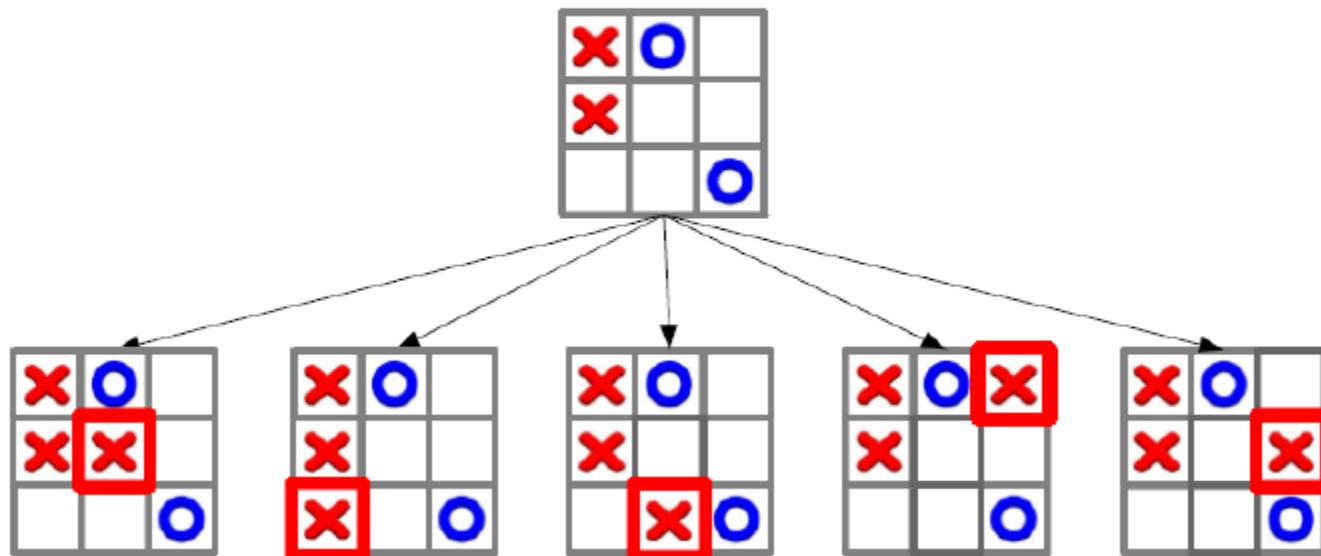
2021-2022



Algorithme MINI MAX: Exemple 4



Algorithme Mini Max pour le TicTacToe



Algorithme MINI MAX: Exemple 4



Fondement de l'IA

2021-2022

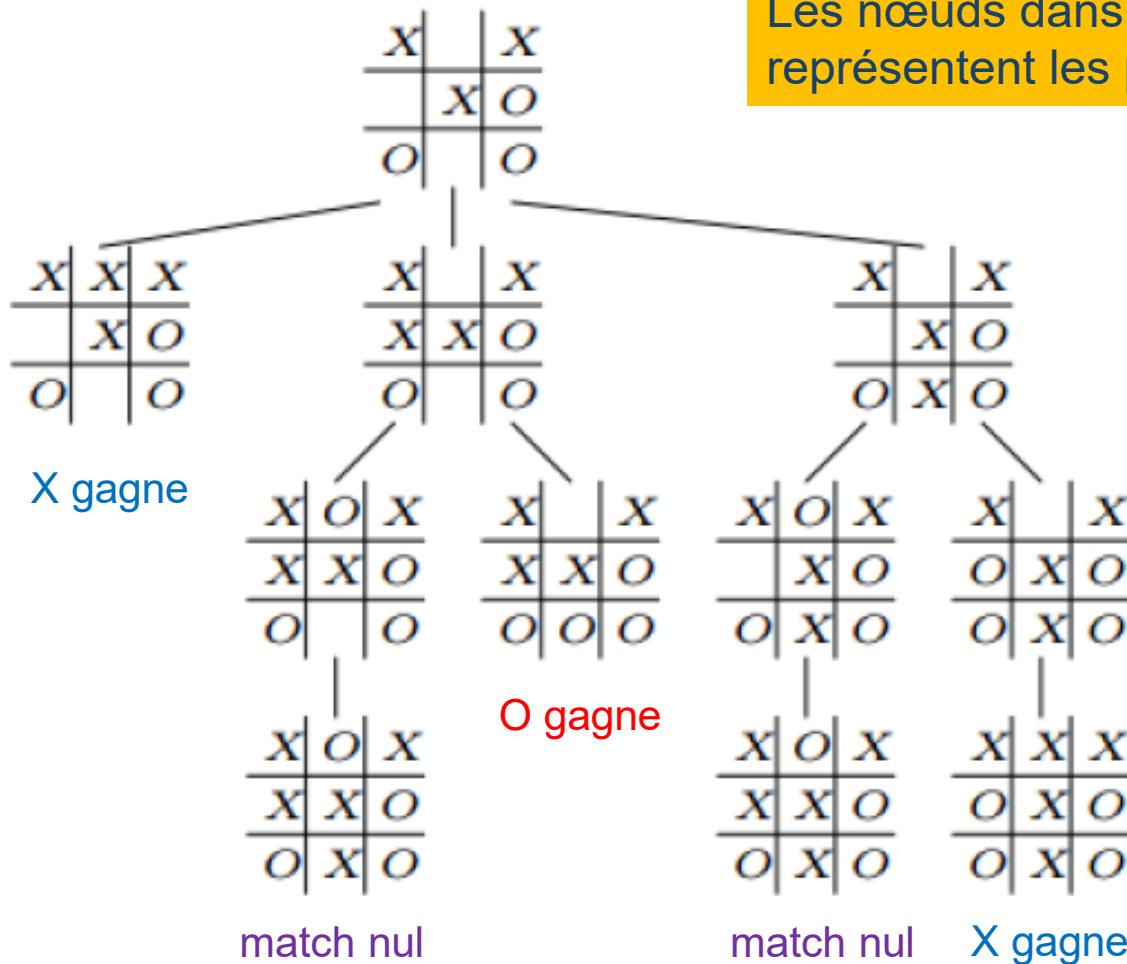
Algorithme Mini Max pour le TicTacToe

J1 joue X

J2 joue O

J1 joue X

J2 joue O



Les nœuds dans l'arbre
représentent les positions du jeu

Algorithme MINI MAX: Exemple 4



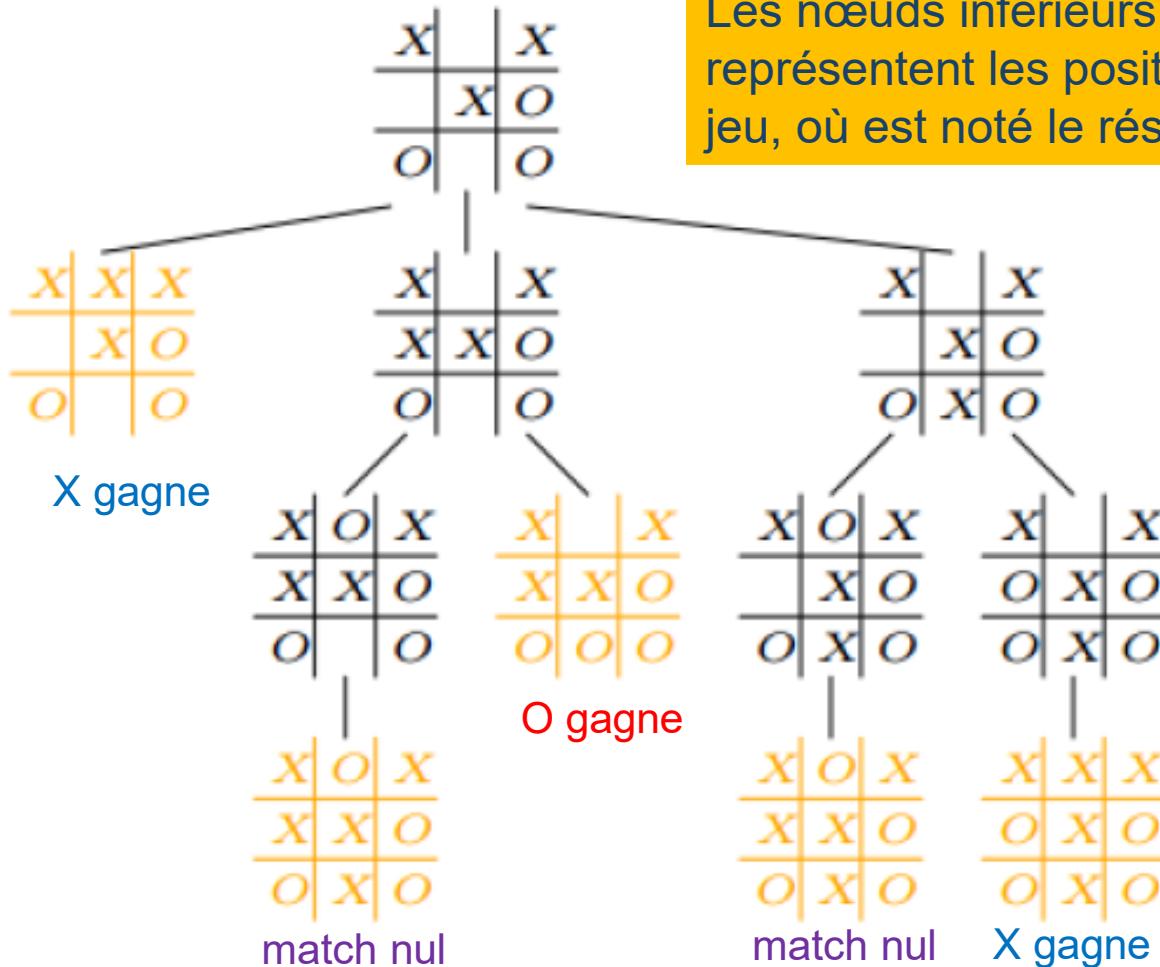
Algorithme Mini Max pour le TicTacToe

J1 joue X

J2 joue O

J1 joue X

J2 joue O

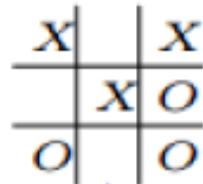


Algorithme MINI MAX: Exemple 4



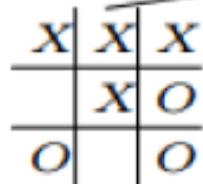
Algorithme Mini Max pour le TicTacToe

J1 joue X

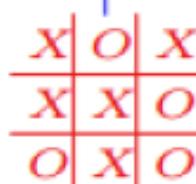
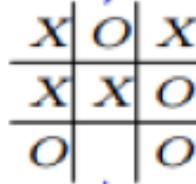
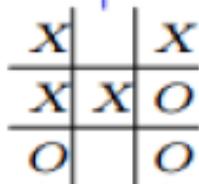


Conclure le meilleur coup
pour que J1 gagne

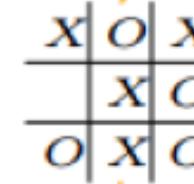
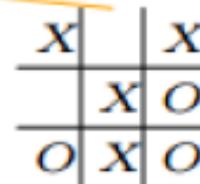
J2 joue O



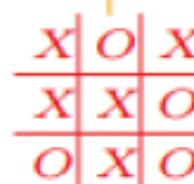
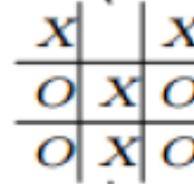
X gagne



match nul



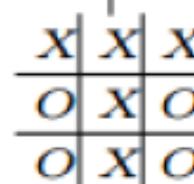
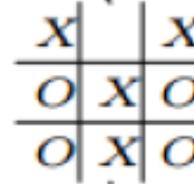
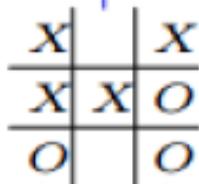
O gagne



match nul

J1 joue X

J2 joue O



X gagne

Algorithme MINI MAX: Exemple 4



Algorithme Mini Max pour le TicTacToe

- Pour le tic-tac-toe, supposons que Max joue avec les X

$\text{EVAL}(n) = (\text{nb. de rangées, colonnes et diagonales disponibles pour Max}) - (\text{nb. de rangées, colonnes et diagonales disponibles pour Min})$

•

X	O	

$$\text{EVAL}(n) = 6 - 4 = 2$$

O	X	X
O		

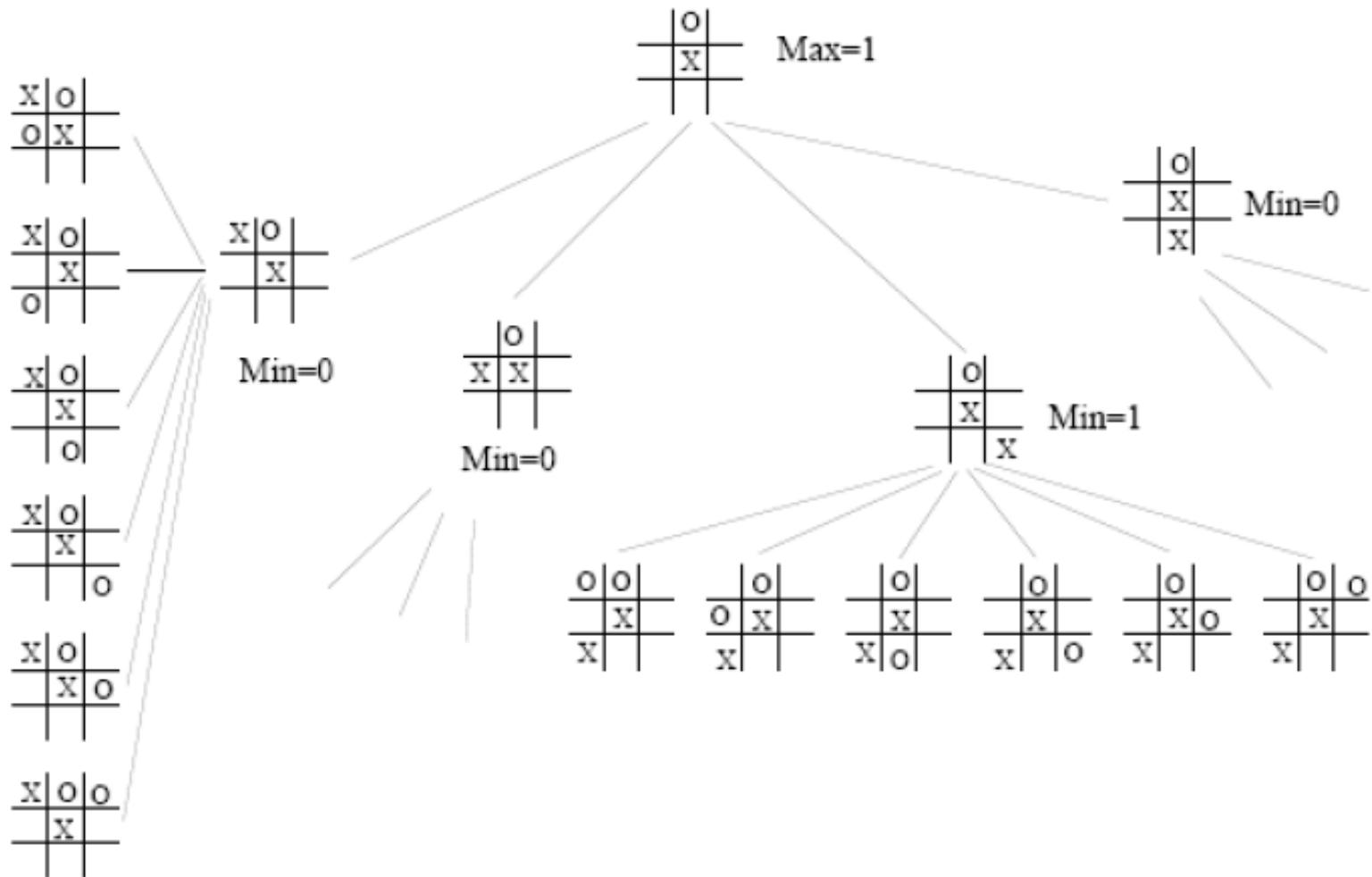
$$\text{EVAL}(n) = 4 - 3 = 1$$

Algorithme MINI MAX: Exemple 4



Algorithme Mini Max pour le TicTacToe

Exemple d'évaluation des nœuds terminaux:

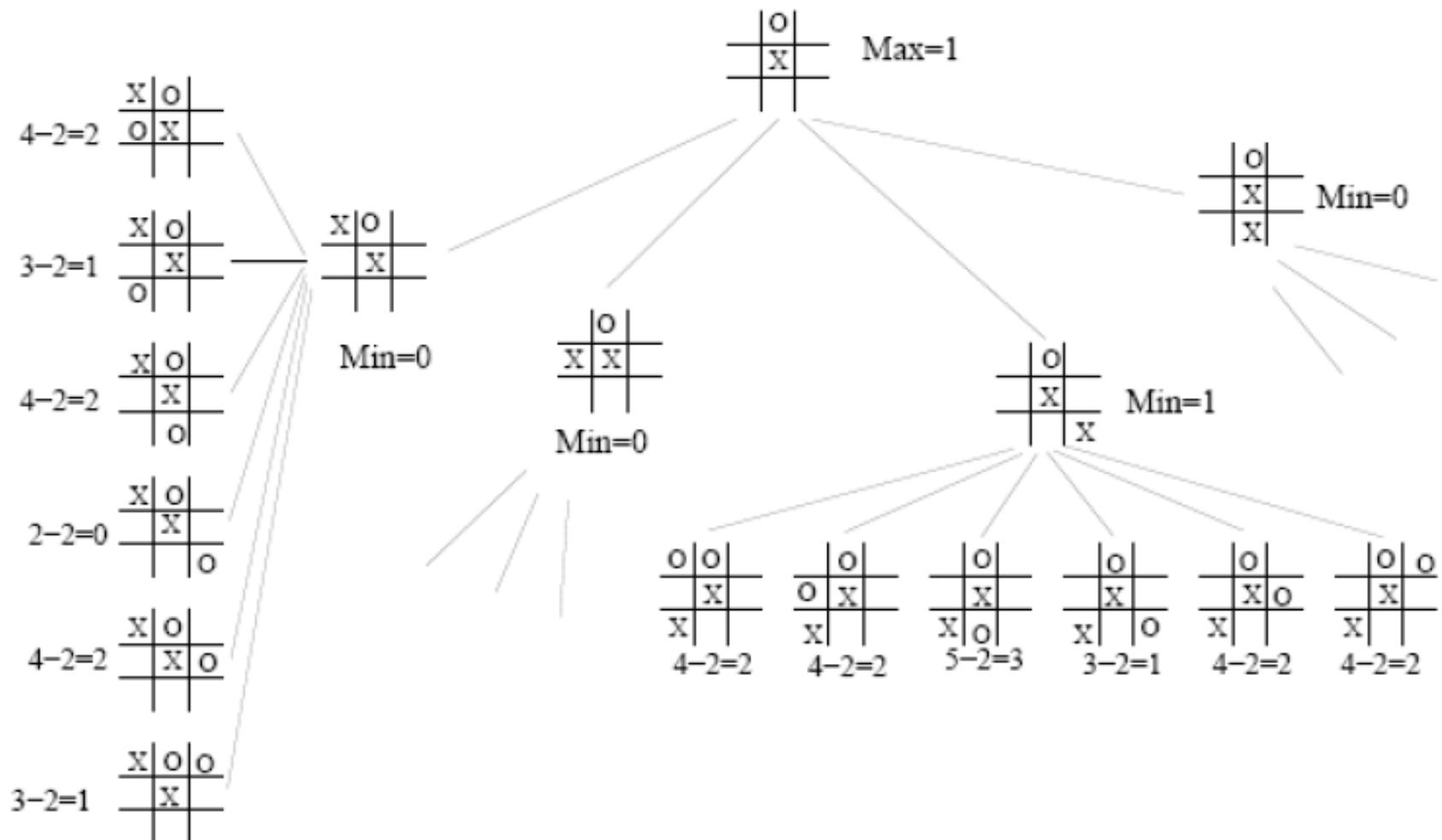


Algorithme MINI MAX: Exemple 4



Algorithme Mini Max pour le TicTacToe

Exemple d'évaluation des nœuds terminaux:



L'algorithme d'élagage α - β

2021-2022

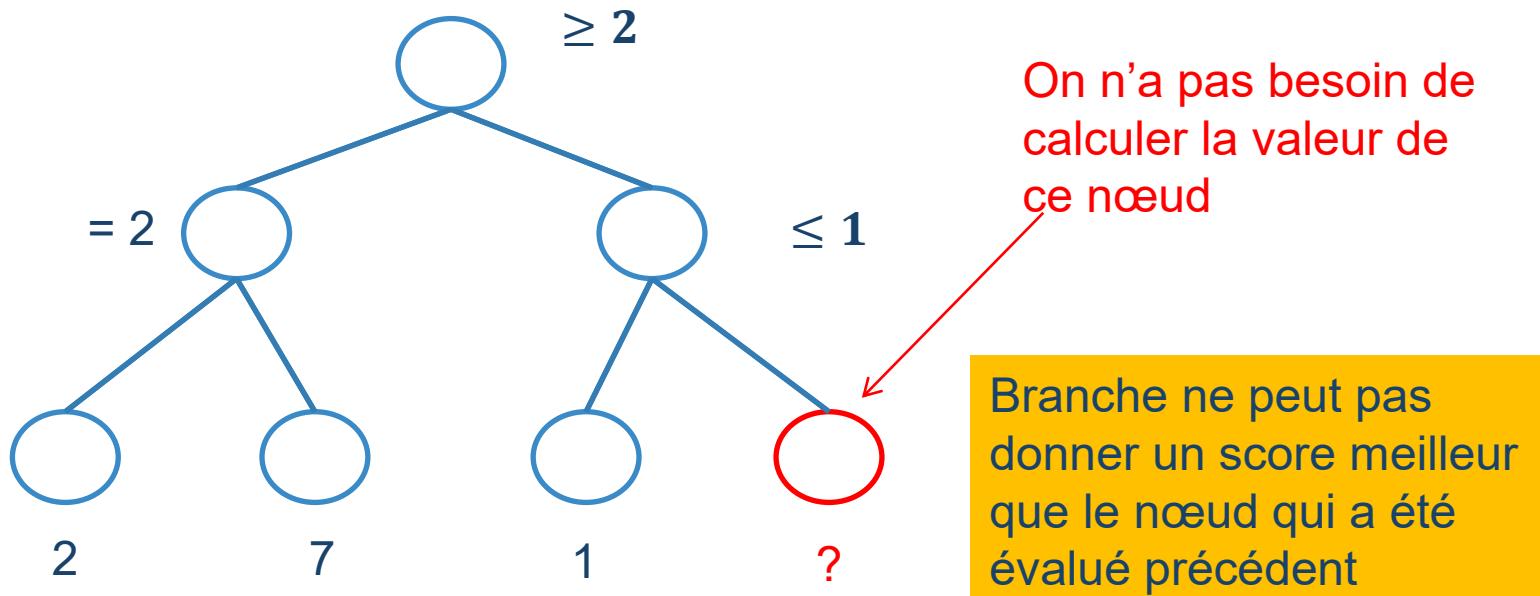


- L'algorithme d'élagage $\alpha - \beta$ est un algorithme de recherche qui cherche à réduire le nombre de nœuds évalués par l'algorithme minimax dans son arbre de recherche
- C'est un algorithme de recherche à adversaire utilisé couramment pour le jeu à deux joueurs (Tic tac toe, Echecs, etc)
- Il arrête complètement l'évaluation d'un coup lorsqu'il est possible que le coup actuel est pire que le coup précédemment examiné
- Appliqué à un arbre minimax standard, il retourne le même coup que minimax, mais élimine les branches qui ne vont pas influencer la décision finale

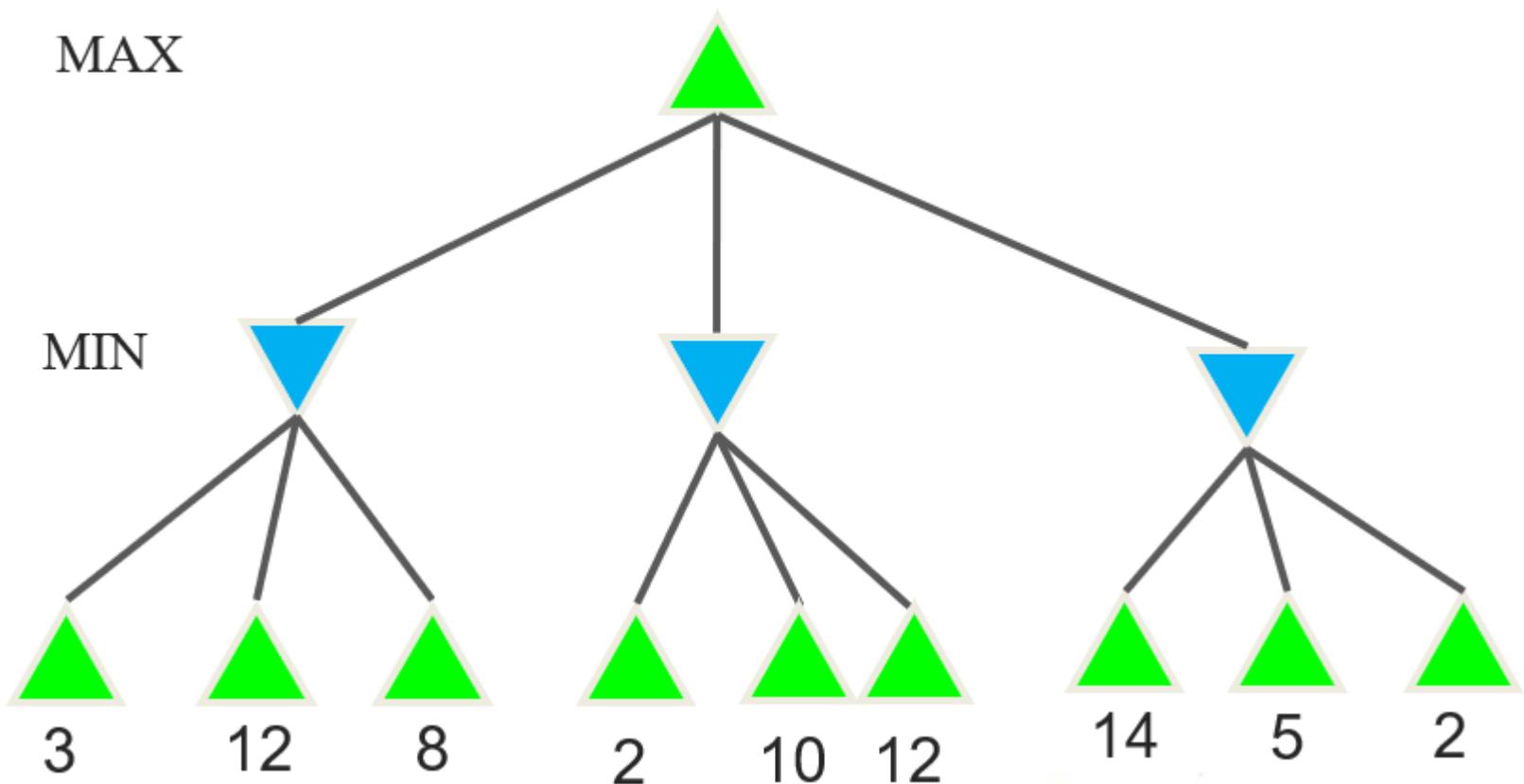
L'algorithme d'élagage α - β



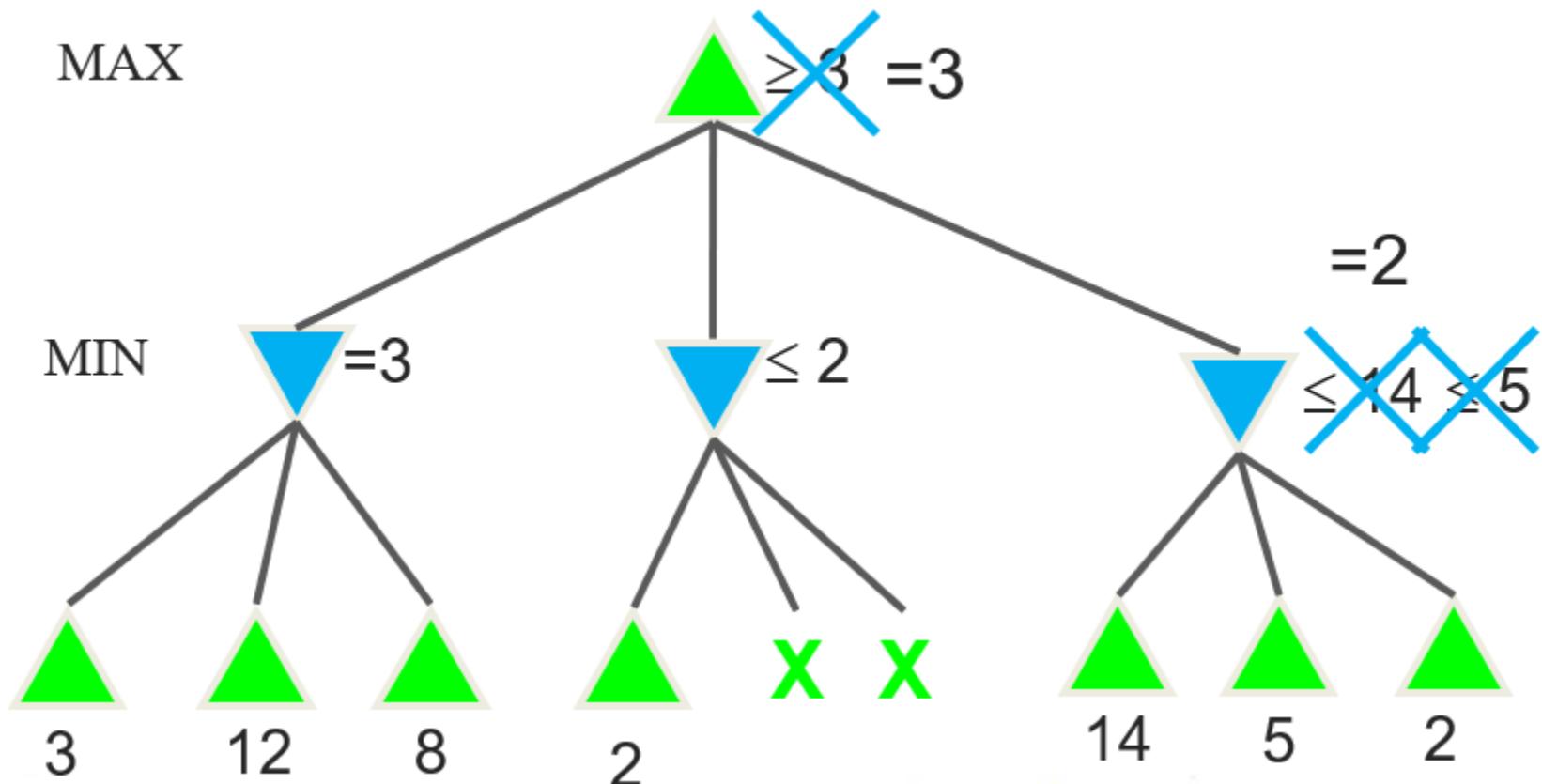
- Initialiser $\alpha = -\infty$ et $\beta = +\infty$
- Parcourez l'arbre en profondeur d'abord
- Arrêter la recherche du nœud MAX si $\alpha \geq \beta$ de son nœud parent
- Arrêter la recherche du nœud MIN si $\beta \leq \alpha$ de son nœud parent



Exemple d'élagage α - β



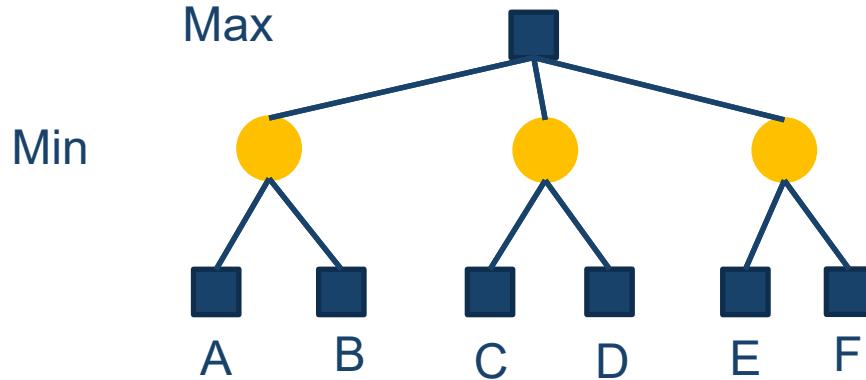
Exemple d'élagage α - β



Algorithme d'élagage α - β



Exercice: Considérez l'arbre de jeux suivant.



- 1) Soit $A=3$; $B=5$; $C=1$; $D=8$; $E=2$; $F=1$
 - a) Appliquez l'algorithme MINIMAX sur cet arbre
 - b) Appliquez l'algorithme α - β sur cet arbre
- 2) Donnez des valeurs (toutes différentes les unes aux autre, et identiques pour les deux parcours) aux feuilles de façon à ce que l'algorithme α - β :
 - a) coupe une seule feuille avec un parcours de gauche à droite
 - b) coupe au moins une feuille avec un parcours de droite à gauche



Fondement de l'IA

Cours: Intélligence Artificielle

2ème année informatique de gestion

Enseignante: Dr. Saoussen Mathlouthi

Saoussen.mathlouthi@ensi-uma.tn



Chapitre 5:

Satisfaction de contraintes

Plan



Rappel

Problème de satisfaction de contraintes CSP

Définition d'un problème CSP

Exemples

Algorithme de DFS pour CSP

Algorithme backtracking-search

Rappel: recherche dans un espace d'états



- Nous avons vu qu'on peut résoudre certains problèmes en les formulant comme des problèmes de recherche dans un graphe d'états:
 - Chaque nœud est une configuration (**état**) de l'environnement
 - La **fonction de transition** reflète les propriétés
 - On définit une **heuristique (h)** pour guider efficacement l'exploration
- Par contre, les nœuds du graphe sont **opaque** vis-à-vis de l'algorithme de recherche:
 - Un état est une **boite noire**; N'importe quelle structure de données qui contient un test pour le but, une fonction d'évaluation, une fonction successeur
 - L'algorithme de recherche ne sait pas comment le choix des successeurs est fait par la fonction de transition

Problème de satisfaction de contraintes

Fondement de l'IA

2021-2022



- La résolution de problèmes de satisfaction de contraintes peut être vue comme un cas particulier de la recherche heuristique
- La structure interne des nœuds est un ensemble de **variables** avec des **valeurs** correspondantes
- Les transitions entre les nœuds tiennent de **contraintes** sur les valeurs possibles des variables
- Sachant cela, on va pouvoir utiliser des heuristiques générales, plutôt que des heuristiques spécifiques à une application
- En traduisant un problème sous forme de satisfaction de contraintes, on élimine la difficulté de définir l'heuristique $h(n)$ pour notre application

Définition d'un problème de satisfaction de contraintes



- Formellement, un problème de satisfaction de contraintes (ou CSP pour Constraint Satisfaction Problem) est défini par:
 - Un ensemble fini de **variables** $V=\{X_1, \dots, X_N\}$
 - Chaque **variable** X_i a un domaine D_i de **valeurs** possibles
 - Un ensemble fini de **contraintes** C_1, \dots, C_M sur les variables
 - Une contrainte restreint les valeurs pour un sous-ensemble de variables
- un **état (nœud)** d'un problème CSP est défini par une assignation de valeurs $\{X_i=v_i, X_j=v_j, \dots\}$ à certaines ou à toutes les variables
 - Assignation **compatible** ou **légale**: une assignation qui ne viole aucune contrainte
 - Assignation **complète**: Assignation qui concerne toutes les variables
 - Une solution à un problème CSP est une assignation complète et compatible



Exemple 1

- Soit le problème CSP défini comme suit:
 - Ensemble de variables $V=\{X_1, X_2, X_3\}$
 - Un domaine pour chaque variable $D_1=D_2=D_3=\{1,2,3\}$
 - Une contrainte spécifiée par l'équation linéaire $X_1+X_2=X_3$
- Il y a trois solution possibles
 - $\{X_1=1, X_2=1, X_3=2\}$
 - $\{X_1=1, X_2=2, X_3=3\}$
 - $\{X_1=2, X_2=1, X_3=3\}$

Exemples

Fondement de l'IA

2021-2022



Exemple 2: Coloriage de carte



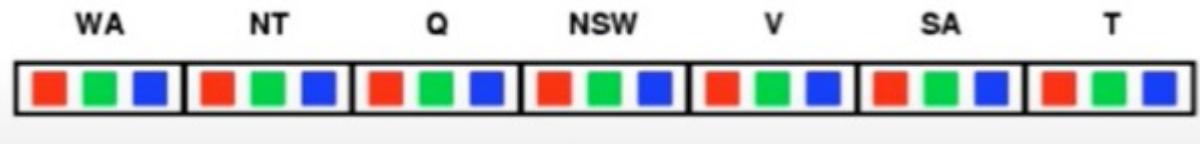
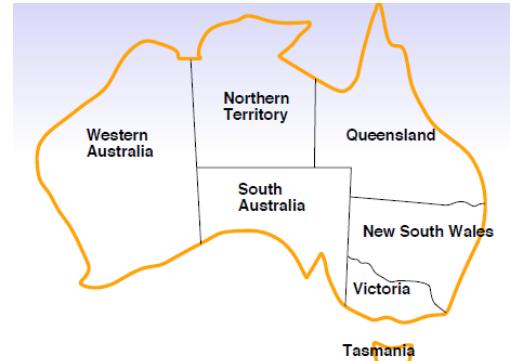
- ❖ Soit une carte de l'Australie
- ❖ On doit utiliser seulement trois couleurs (rouge, vert et bleu) de sorte que deux états frontaliers n'aient jamais les mêmes couleurs

Exemples



Exemple 2: Coloriage de carte

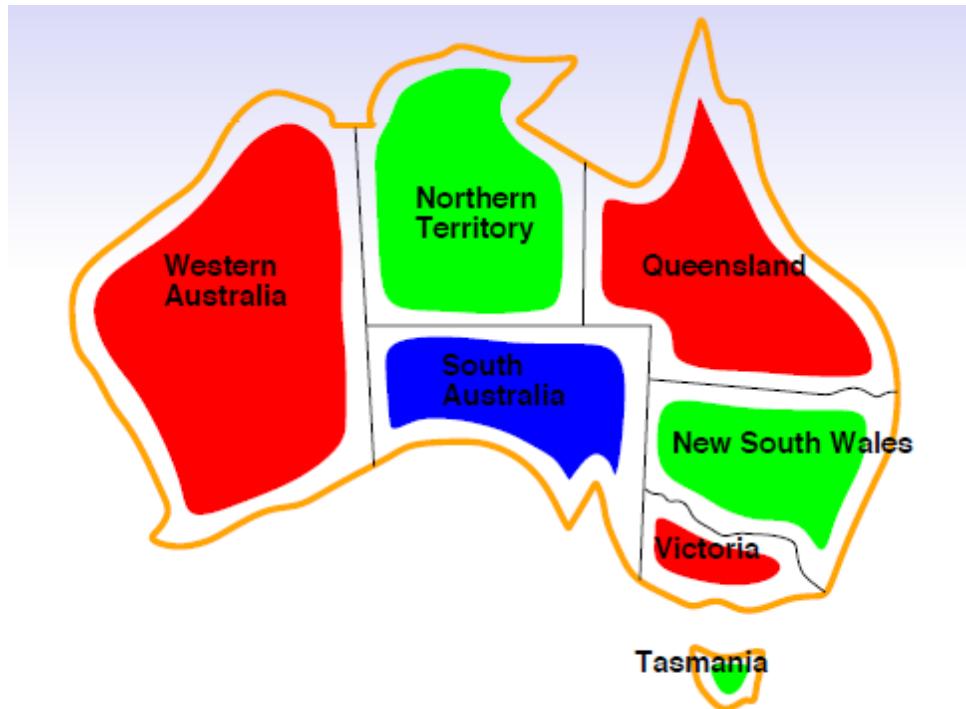
- Variables: les variables sont les états de l'Australie
 - V: WA, NT, SA, Q, NSW, V, T
- Domaines: le domaine de chaque variable est l'ensemble des trois couleurs rouge, vert et bleu
 - $D_i = \{R, G, B\}$
- Contraintes : les régions adjacentes doivent être de couleurs différentes
 - $WA \neq NT, \dots, NT \neq Q, \dots$



Exemples



Exemple 2: Coloriage de carte



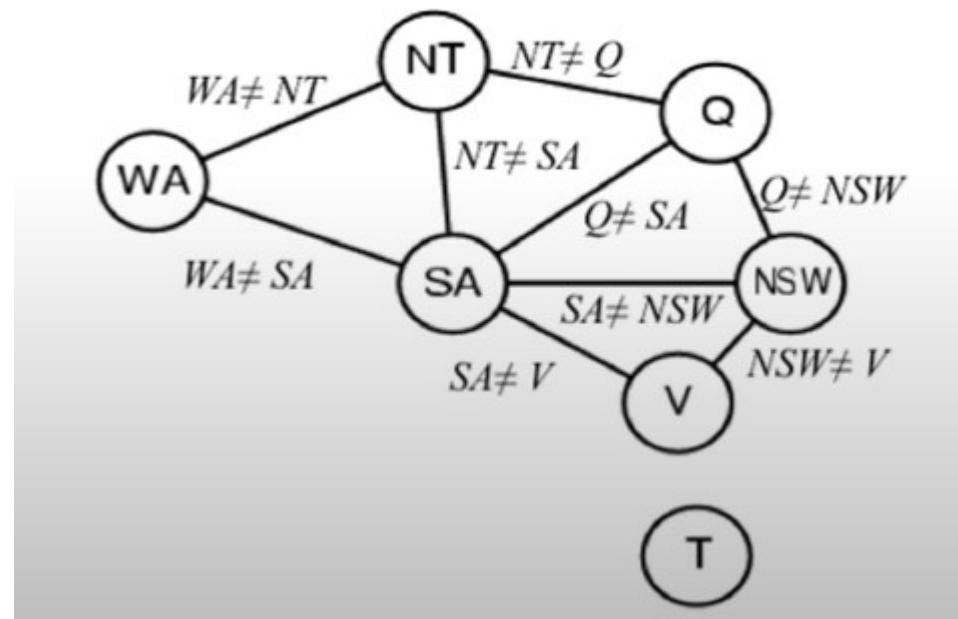
Solution:

- Les solutions sont des affectations qui satisfont toutes les contraintes
- Par exemple, {WA = R, NT = G, Q = R, NSW = G, V = R, SA = B, T = G}

Graphe de contraintes



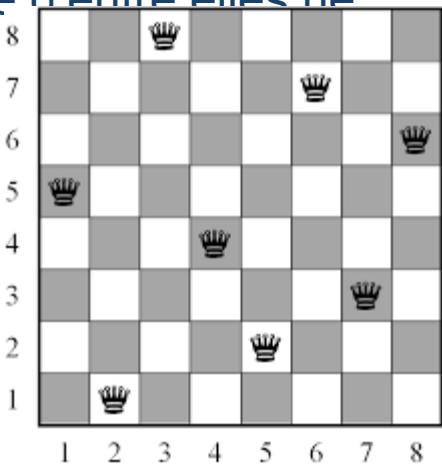
- ❖ Pour des problèmes avec des contraintes binaires (c-à-d, entre deux variables), on peut visualiser le problème CSP par un **graphe de contraintes**
- ❖ Un **graphe de contrainte** est un graphe dont les **nœuds** sont des **variables** et les **arcs** sont des **contraintes** entre les deux variables





Exemple 3: N-Queens

- ❖ Positionner N reines sur un échiquier de sorte qu'aucune d'entre elles ne soit en position d'attaquer une autre
 - ❖ Exemple avec 8 reines
- ❖ Nombre de configurations:
 - ❖ N=4: $4^4=256$ configurations
 - ❖ N=8: $8^8=16\ 777\ 216$ configurations
 - ❖ N=16: $16^{16}=18\ 446\ 744\ 073\ 709\ 551\ 616$ configurations

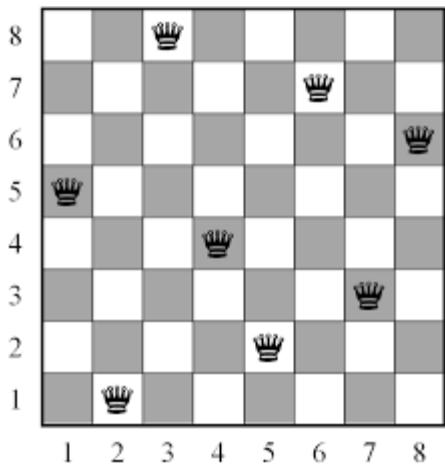




Exemple 3: N-Queens

Modélisation comme problème CSP:

- ❖ **Variables**: colonnes 1, ..., N
- ❖ **Domaines**: rangées 1, ..., N
- ❖ La colonne i a la valeur k si la reine dans la colonne i est dans la rangée k
- ❖ **Contraintes**: pas deux reines sur même rangée ou diagonale (par définition, il y a toujours une reine par colonne)



Problèmes CSPs du monde réel

Fondement de l'IA

2021-2022



- ❖ Problèmes d'affectation (eg. qui enseigne quel cours?)
- ❖ Problèmes d'emploi du temps
- ❖ Configuration de matériels
- ❖ Planification pour les transports
- ❖ Planification dans les usines
- ❖ Allocation de salles
- ❖ ...
- ❖ **Note:** beaucoup de problèmes du monde réel impliquent des variables à valeurs réelles

Algorithme de recherche en profondeur d'abord pour CSP

2021-2022



Fondement de l'IA

On peut utiliser la recherche dans un graphe avec les paramètres suivants:

- ❖ Un **état** est une assignation
- ❖ **État initial**: assignation vide { }
- ❖ **Fonction de transition**: assigne une valeur à une variable non encore assignée
- ❖ **Fonction but**: retourne vrai si l'assignation est complète et compatible

L'algorithme est général et s'applique à tous les problèmes CSP

Comme la solution doit être **complète**, elle paraît à une **profondeur N**

Algorithme de recherche en profondeur d'abord pour CSP

Fondement de l'IA

2021-2022



Limitation de l'approche recherche en profondeur d'abord (DFS)

- ❖ Taille de l'arbre de recherche: (D est la taille du domaine, N est le nombre des variables)
 - Premier niveau: $N*D$ branches
 - Niveau suivant: $(N-1)*D$ successeurs pour chaque nœud
 - Ainsi de suite jusqu'à le niveau N
 - Donc, taille des nœuds= $N!D^N$
- ❖ L'algorithme ignore la commutativité des transitions:
 - $SA=R$ suivi de $WA=B \Leftrightarrow WA=B$ suivi de $SA=R$
 - en tenant compte de la commutativité, taille des nœuds= D^N

Idée 1: considérer une seule variable à assigner à chaque niveau

Algorithme de recherche en profondeur d'abord pour CSP

Fondement de l'IA

2021-2022



Limitation de l'approche recherche en profondeur d'abord (DFS)

- ❖ S'il y a des contraintes violées alors inutile de continuer à assigner des variables

Idée 2: reculer (**backtrack**) lorsqu'aucune nouvelle assignation compatible est possible

- ❖ Le **backtracking-search** est la combinaison des deux idées
 - C'est l'algorithme de base pour résoudre les problèmes CSP

Algorithme backtracking-search



Algorithme Backtracking-search (CSP)

1. Retourner **BACK-TRCAK** ($\{\}$, CSP)

information sur les variables, domaines, contraintes du problème CSP

Algorithme **BACK-TRCAK** (assignation, CSP)

- ← assignation de variables à des valeurs
1. Si assignation complète, retourner assignation ;
 2. $X = \text{VAR-NON-ASSIGNEE}$ (assignation, CSP) ← choix de prochaine variable
 3. Pour chaque v dans **VALEUR-ORDONNEES** (X , assignation, CSP)
 1. Si **COMPATIBLE** ($(X=v)$, assignation, CSP)
 2. Ajouter ($X=v$) à assignation
 3. $CSP^* = CSP$ mais où **DOMAINE** (X , CSP) est $\{v\}$ ← ordre des valeurs à essayer
 4. CSP^* , ok = **INFERENCE** (CSP^*) ← si détecte conflit, ok = faux
 5. Si ok = vrai
 1. Résultat = **BACK-TRACK** (assignation, CSP^*)
 2. Si résultat \neq faux, retourner résultat
 6. Enlever ($X = v$) de assignation
 7. Retourner faux.

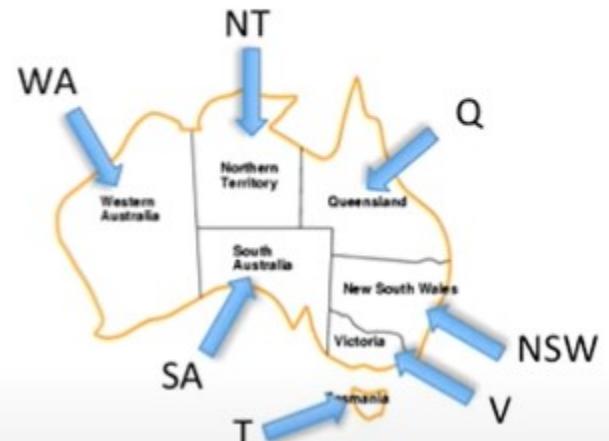
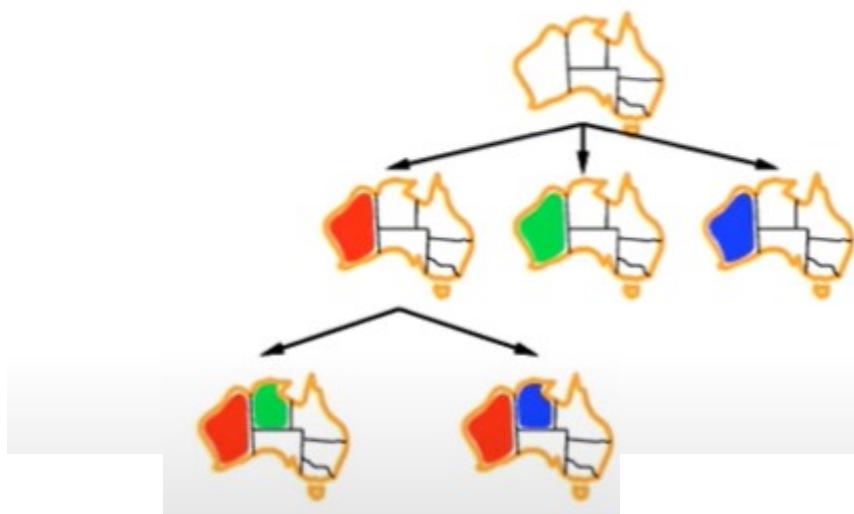
Algorithme backtracking-search



Fondement de l'IA

2021-2022

Illustration du Backtracking-search





Exemple de problème CSP: Résoudre le problème cryptarithmétique avec l'exemple SEND + MORE = MONEY

- ❖ Où chaque lettre représente un chiffre différent (compris entre 0 et 9)
- ❖ On souhaite connaître la valeur de chaque lettre, sachant que la première lettre de chaque mot représente un chiffre différent de 0

$$\begin{array}{r} \text{S} \quad \text{E} \quad \text{N} \quad \text{D} \\ + \quad \text{M} \quad \text{O} \quad \text{R} \quad \text{E} \\ \hline \text{M} \quad \text{O} \quad \text{N} \quad \text{E} \quad \text{Y} \end{array}$$

Exemple de problème CSP



Fondement de l'IA

2021-2022

Exercice: Résoudre le problème cryptarithmétique avec l'exemple SEND + MORE = MONEY

- Déterminer les:

- Variables:
- Domaines:
- Contraintes:

$$\begin{array}{r} \text{S} \quad \text{E} \quad \text{N} \quad \text{D} \\ + \quad \text{M} \quad \text{O} \quad \text{R} \quad \text{E} \\ \hline \text{M} \quad \text{O} \quad \text{N} \quad \text{E} \quad \text{Y} \end{array}$$

- Connaître la valeur de chaque lettre, sachant que la première lettre de chaque mot représente un chiffre différent de 0

$$\begin{array}{r} \boxed{} \quad \boxed{} \quad \boxed{} \quad \boxed{} \\ + \quad \boxed{} \quad \boxed{} \quad \boxed{} \quad \boxed{} \\ \hline \boxed{} \quad \boxed{} \quad \boxed{} \quad \boxed{} \quad \boxed{} \end{array}$$

Character	Code
S	
E	
N	
D	
M	
O	
R	
Y	