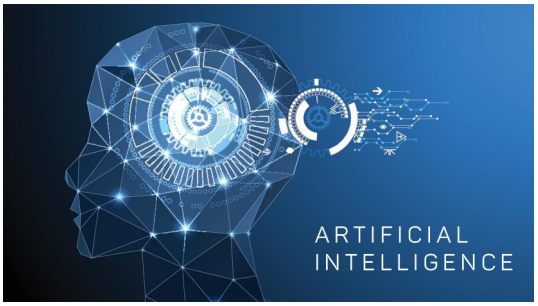
Intelligence Artificielle : Principes et Applications



Pr. Mustapha BOURAHLA, Département d'Informatique, Université de M'Sila, Contact : mustapha.bourahla@univ-msila.dz

Janvier 2020
v 1.0

Table des matières

Objectifs

Fiche d'enseignement

- Intitulé du Master: Intelligence Artificielle
- Semestre: 01
- Intitulé de l'UE: UEF1
- Intitulé de la matière: Intelligence Artificielle: Principes et Applications
- Crédits: 04
- Coefficients: 02

Ce module est destiné aux étudiants de la première année Master "Intelligence Artificielle" de la filière informatique dont les objectifs sont :

- Situer l'Intelligence Artificielle en tant que discipline scientifique
- Maîtriser les concepts de base de I 'intelligence artificielle
- Connaître les méthodes de représentation des connaissances
- Avoir un aperçu des approches pour la résolution de problèmes en IA

Connaissances préalables recommandées

- Les connaissances algorithmiques et éléments de logiques mathématiques

Contenu de la matière

Chapitre 1 : Introduction : une brève histoire de l'IA

Chapitre 2 : L'IA et la résolution des problèmes

1. Démarche IA pour résoudre un problème
2. Démarches de construction de la solution
3. Résolution par décomposition en sous problèmes
4. Résolution par satisfaction de contraintes

Chapitre 3 : Les systèmes à base de connaissances

1. Définition
2. Architecture
3. Ingénierie des connaissances
4. Domaines d'application: Santé et IA, Agriculture et IA, Énergiehydrocarbure, énergie solaire + IA, Environnement (catastrophes naturelles (inondation, séisme, etc),...) + IA, Administration et IA, Défense nationale et IA, Transport et IA et industrie et IA

Chapitre 4 : Prolog : un langage de l'IA

Mode d'évaluation

- Contrôle continu 50% et Examen écrit 50%

Chapitre 1 : Introduction

I

Ce premier chapitre vous introduit la notion de l'intelligence artificielle et un peu d'historique.

1. Définition de l'Intelligence Artificielle

Définition

- L'intelligence artificielle (IA) est l'ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence humaine
- Elle correspond donc à un ensemble de concepts et de technologies plus qu'à une discipline autonome constituée
- L'IA est le grand mythe de notre temps

L'intelligence Artificielle

- Souvent classée dans le groupe des sciences cognitives, elle fait appel à la neurobiologie computationnelle (particulièrement aux réseaux neuronaux), à la logique mathématique (partie des mathématiques et de la philosophie) et à l'informatique.
- Elle recherche des méthodes de résolution de problèmes à forte complexité logique ou algorithmique. Par extension elle désigne, dans le langage courant, les dispositifs imitant ou remplaçant l'homme dans certaines mises en œuvre de ses fonctions cognitives.
- Ses finalités et son développement suscitent, depuis toujours, de nombreuses interprétations, fantasmes ou inquiétudes s'exprimant tant dans les récits ou films de science-fiction que dans les essais philosophiques.
- La réalité semble encore tenir l'intelligence artificielle loin des performances du vivant ; ainsi, l'IA reste encore bien inférieure au chat dans toutes ses aptitudes naturelles.

2. Historique

Alan Turing

- Historiquement, l'idée d'intelligence artificielle semble émerger dans les années 1950 quand Alan Turing se demande si une machine peut « penser ».
- Turing explore ce problème et propose une expérience (maintenant dite test de Turing) visant à trouver à partir de quand une machine deviendrait consciente.

Warren Weaver

- Une autre origine probable est la publication, en 1949, par Warren Weaver d'un mémorandum sur la traduction automatique des langues qui suggère qu'une machine puisse faire une tâche qui relève typiquement de l'intelligence humaine.

Résultats du développement des techniques informatiques

- Augmentation de la puissance de calcul aboutit ensuite à plusieurs avancées
- dans les années 1980, l'apprentissage automatique se développe, notamment avec la renaissance du connexionnisme.
- L'ordinateur commence à déduire des « règles à suivre » en analysant seulement des données.
- Parallèlement, des algorithmes « apprenants » sont créés qui préfigurent les futurs réseaux de neurones, l'apprentissage par renforcement, les machines à vecteurs de support, etc.).
- Depuis les années 1960, l'intelligence artificielle devient un domaine de recherche international.
- dans les années 2000, le Web 2.0, le big data et de nouvelles puissances et infrastructures de calcul permettent à certains ordinateurs d'explorer des masses de données sans précédent ; c'est l'apprentissage profond (« deep learning »)

Les limites

- Les bornes de ce domaine varient, ainsi optimiser un itinéraire était considéré comme un problème d'intelligence artificielle dans les années 1950 et n'est plus considéré aujourd'hui que comme un simple problème d'algorithmie.
- Vers 2015, le secteur de l'intelligence artificielle cherche à relever quatre défis : la perception visuelle, la compréhension du langage naturel écrit ou parlé, l'analyse automatique du langage et la prise de décision autonome.
- Produire et organiser des données nombreuses et de qualité, c'est-à-dire corrélées, complètes, qualifiées (sourcées, datées, géo-référencées...), historisées est un autre enjeu.
- La capacité déductive et de généralisation pertinente d'un ordinateur, à partir de peu de données ou d'un faible nombre d'événements, est un autre objectif, plus lointain.
- Entre 2010 et 2016, les investissements auraient été décuplés, atteignant une dizaine de milliards en 2016

Travaux marquants de l'IA

- En 1950 , le mathématicien britannique Alan Turing publie, dans le journal philosophique Mind, un article intitulé Computing Machinery and Intelligence.
- Dans cet article, il décrit un test (Test de Turing) qui consiste à faire communiquer un individu avec un interlocuteur invisible à travers un terminal d'ordinateur.
- L'individu doit alors deviner si son interlocuteur est un être humain ou un système d'IA imitant un être humain.
- Le but du test est de déterminer si le système d'IA est intelligent.
- Considéré par beaucoup comme l'un des fondateurs de l'informatique, Turing est également un des précurseurs de l'IA.
- En 1950, "IPL-11", le premier langage d'IA, est créé.
- En 1950, Allen Newell, John Shaw, et Herbert Simon créent le "Logic Theorist", considéré comme le premier programme d'IA.

- En modélisant chaque problème par un arbre, il tente de le résoudre en sélectionnant la branche qui donnerait le résultat le plus proche de l'objectif final.
- Le programme fût capable de redémontrer des théorèmes déjà établis par les mathématiciens mais de manière plus "élégante".

- En 1956, John McCarthy, considéré comme le père de l'IA, organise "The Dartmouth summer research project on Artificial Intelligence".
- C'est à lui que l'on doit le terme "Intelligence Artificielle".
- Pour lui, "toute activité intellectuelle peut être décrite avec suffisamment de précision pour être simulée par une machine.
- "En 1958, il introduit le langage LISP.
- En 1959, avec Marvin Minsky, il fonde le "MIT AI Lab".

- En 1959, élaboration du GPR (General Problem Resolver), qui consiste à définir un état initial, un ou plusieurs états finaux, et des opérateurs de transition entre les états.

- Problèmes des premiers systèmes en IA : incapacité à imiter la capacité de l'homme à utiliser le contexte d'un problème pour déterminer le sens des mots et des phrases.
- L'échec du GPR, abandonné en 1967, marquera le début d'un période où l'IA se heurtera à de nombreux détracteurs.
- Parmi eux, Hubert Dreyfus, qui écrit "What Computers Can't Do" en 1972 dénoncera les sommes importantes dépensées par le gouvernement US pour le développement de l'IA.
- Les premiers succès auront vite montré leurs limites et beaucoup de biologistes et d'informaticiens jugeront les objectifs premiers de l'IA trop optimistes.
- En fait, le principal problème des ces systèmes était leur incapacité à imiter la capacité de l'homme à utiliser le contexte d'un problème pour déterminer le sens des mots et des phrases.
- On peut par exemple citer ce passage de l'ouvrage de Dreyfus : "In spite of journalistic claims at various moments that machine translation was at last operational, this research produced primarily a much deeper knowledge of the unsuspected complexity of syntax and semantics".
- Même si le "super-optimisme" du début semble disparaître, les tenants de l'IA n'en sont pas moins désespérés.

Années 70

- De nombreuses nouvelles méthodes de développement de l'IA sont testées.
- En 1971, le langage PROLOG est créé par Colmerauer.
- L'année 1974 verra l'arriver des premiers systèmes experts. dont le plus célèbre MYCIN (Edward H. Shortliffe), conçu pour l'aide au diagnostic et au traitement de maladies bactériennes du sang.
- En 1979, Mycin sera considéré par le "Journal of American Medical Assoc" comme aussi bon que les experts médicaux.
- La même année, le premier robot piloté par ordinateur est conçu.
- On notera également dans les années 70, l'abandon des subventions versées par les gouvernements (US pour la plupart) pour quelques programmes de recherche en IA.

Les années 80 et 90, Accélération du mouvement.

- Avec l'efficacité prouvée des systèmes experts, les ventes de matériels IA (hardware ou logiciel) grimpent en flèche.
- L'IA commence à intéresser les grandes firmes (Boeing, General Motors, etc).

- Face aux détracteurs, les "pro-IA" se défendent : en 1982, Minsky écrit "Why People Think Computers Can't", en réponse notamment aux critiques de Dreyfus.
- De nouvelles branches de l'IA font leur percée sur le marché, notamment celui de la vision de la machine. Par exemple, des travaux sont réalisés sur l'utilisation d'une caméra reliée à un ordinateur pour améliorer le contrôle qualité (Reconnaissance des formes, etc).
- Malgré quelques échecs, l'IA semble renaître : la logique floue est conçue, les réseaux de neurones sont réhabilités.
- Durant les années 80, l'IA est sortie des laboratoires pour montrer ses utilisations possibles dans la vie réelle.
- Mais la recherche continue, et de grands projets ont vu le jour au début des années 90.

3. Les précurseurs

Intelligence Artificielle d'autrefois

- Si les progrès de l'intelligence artificielle sont récents, ce thème de réflexion est tout à fait ancien, et il apparaît régulièrement au cours de l'histoire.
- Les premiers signes d'intérêt pour une intelligence artificielle et les principaux précurseurs de cette discipline sont les suivants.

Automates

Autrefois, plusieurs automates ont été conçus pour résoudre certains problème d'Intelligence Artificielle comme :

- A la fin du Moyen Âge, Roger Bacon a conçu automates doués de la parole; en fait, probablement de mécanismes simulant la prononciation de certains mots simples.
- Léonard de Vinci a construit en 1515 un automate en forme de lion pour amuser le roi de France.
- Gio Battista Aleotti et Salomon de Caus, eux, ont construit des oiseaux artificiels et chantants, des flûtiste mécaniques, des nymphes, des dragons et des satyres animés pour égayer des fêtes aristocratiques, des jardins et des grottes.
- Jacques de Vaucanson a construit en 1738 un « canard artificiel de cuivre doré, qui boit, mange, cancane, barbote et digère comme un vrai canard ». Il était possible de programmer les mouvements de cet automate, grâce à des pignons placés sur un cylindre gravé, qui contrôlaient des baguettes traversant les pattes du canard.

Pensée automatique

- Une des premières tentatives de formalisation de la pensée connue utilisaient les astrologues arabe pour générer des idées supposées logiques, dont l'invention est attribuée à Abu al-Abbas as-Sabti au XIIe siècle.
- Raymond Lulle s'en est probablement inspiré pour mettre au point son Ars Magna. Missionnaire, philosophe, et théologien espagnol du XIIIe siècle, il essaya lui aussi de générer des idées grâce à un système mécanique. Il combinait aléatoirement des concepts grâce à une sorte de règle à calcul, sur laquelle pivotaient des disques concentriques gravés de lettres et de symboles philosophiques. Il fondait sa méthode sur l'identification de concepts de base, puis leur combinaison mécanique soit entre eux, soit

avec des idées connexes. Raymond Lulle l'appliqua à la métaphysique, puis à la morale, à la médecine et à l'astrologie. Mais il n'utilisait que la logique déductive, ce qui ne permettait pas à son système d'acquérir un apprentissage, ni davantage de remettre en cause ses principes de départ : seule la logique inductive le permet.

- Gottfried Wilhelm Leibniz, au XVIIe siècle, a imaginé un calcul pensant (calculus rationator), en assignant un nombre à chaque concept. La manipulation de ces nombres aurait permis de résoudre les questions les plus difficiles, et même d'aboutir à un langage universel. Leibniz a toutefois démontré que l'une des principales difficultés de cette méthode, également rencontrée dans les travaux modernes sur l'intelligence artificielle, est l'interconnexion de tous les concepts, ce qui ne permet pas d'isoler une idée de toutes les autres pour simplifier les problèmes liés à la pensée.
- George Boole a inventé la formulation mathématique des processus fondamentaux du raisonnement, connue sous le nom d'algèbre de Boole. Il était conscient des liens de ses travaux avec les mécanismes de l'intelligence, comme le montre le titre de son principal ouvrage paru en 1854 : Les Lois de la pensée (The laws of thought), sur l'algèbre booléenne.
- Gottlob Frege perfectionna le système de Boole en formalisant le concept de prédicat, qui est une entité logique soit vraie, soit fausse (toute maison a un propriétaire), mais contenant des variables non logiques, n'ayant en soi aucun degré de vérité (maison, propriétaire). Cette formalisation eut une grande importance puisqu'elle permit de démontrer des théorèmes généraux, simplement en appliquant des règles typographiques à des ensembles de symboles. La réflexion en langage courant ne portait plus que sur le choix des règles à appliquer. Par ailleurs, l'utilisateur joue un rôle important puisqu'il connaît le sens des symboles qu'il a inventés et ce sens n'est pas toujours formalisé, ce qui ramène au problème de la signification en intelligence artificielle, et de la subjectivité des utilisateurs.
- Bertrand Russell et Alfred North Whitehead publièrent au début du XXe siècle un ouvrage intitulé Principia Mathematica, dans lequel ils résolvent des contradictions internes à la théorie de Gottlob Frege. Ces travaux laissaient espérer d'aboutir à une formalisation complète des mathématiques.
- Kurt Gödel démontre au contraire que les mathématiques resteront une construction ouverte, en publiant en 1931 un article intitulé « Des propositions formellement indécidables contenues dans les Principia mathematica et autres systèmes similaires ». Sa démonstration est qu'à partir d'une certaine complexité d'un système, on peut y créer plus de propositions logiques qu'on ne peut en démontrer vraies ou fausses. L'arithmétique, par exemple, ne peut trancher par ses axiomes si on doit accepter des nombres dont le carré soit -1. Ce choix reste arbitraire et n'est en rien lié aux axiomes de base. Le travail de Gödel suggère qu'on pourra créer ainsi un nombre arbitraire de nouveaux axiomes, compatibles avec les précédents, au fur et à mesure qu'on en aura besoin. Si l'arithmétique est démontrée incomplète, le calcul des prédicats (logique formelle) est au contraire démontré par Gödel comme complet.
- Alan Turing invente des machines abstraites et universelles (rebaptisées les machines de Turing), dont les ordinateurs modernes sont considérés comme des concrétisations. Il démontre l'existence de calculs qu'aucune machine ne peut faire (un humain pas davantage, dans les cas qu'il cite), sans pour autant que cela constitue pour Turing un motif pour douter de la faisabilité de machines pensantes répondant aux critères du test de Turing.
- Irving John Good33, Myron Tribus et E.T. Jaynes34 ont décrit de façon très claire les principes assez simples d'un robot à logique inductive utilisant les principes de l'inférence bayésienne pour enrichir sa base de connaissances sur la base du Théorème de Cox-Jaynes. Ils n'ont malheureusement pas traité la question de la façon dont on pourrait stocker ces connaissances sans que le mode de stockage entraîne un biais cognitif. Le projet est voisin de celui de Raymond Lulle, mais fondé cette fois-ci sur une logique inductive, et donc propre à résoudre quelques problèmes ouverts.

- Des chercheurs comme Alonzo Church ont posé des limites pratiques aux ambitions de la raison, en orientant la recherche (Herbert Simon, Michael Rabin, Stephen Cook) vers l'obtention des solutions en temps fini, ou avec des ressources limitées, ainsi que vers la catégorisation des problèmes selon des classes de difficulté (en rapport avec les travaux de Cantor sur l'infini).

Années 2000

- L'intelligence artificielle est un sujet d'actualité au XXIe siècle. En 2004, l'Institut Singularity a lancé une campagne Internet appelée « Trois lois dangereuses » : « Three Laws Unsafe » (en lien avec les trois lois d'Asimov) pour sensibiliser aux questions de la problématique de l'intelligence artificielle et l'insuffisance des lois d'Asimov en particulier. (Singularity Institute for Artificial Intelligence 2004).
- En 2005, le projet Blue Brain est lancé, il vise à simuler le cerveau des mammifères. Il s'agit d'une des méthodes envisagées pour réaliser une IA. Ils annoncent de plus comme objectif de fabriquer, dans dix ans, le premier « vrai » cerveau électronique. En mars 2007, le gouvernement sud-coréen annonce que plus tard dans l'année, il émettrait une charte sur l'éthique des robots, afin de fixer des normes pour les utilisateurs et les fabricants. Selon Park Hye-Young, du ministère de l'Information et de la communication, la Charte reflète les trois lois d'Asimov : la tentative de définition des règles de base pour le développement futur de la robotique. En juillet 2009, en Californie une conférence organisée par l'Association for the Advancement of Artificial Intelligence (AAAI), où un groupe d'informaticiens se demande s'il devrait y avoir des limites sur la recherche qui pourrait conduire à la perte de l'emprise humaine sur les systèmes informatiques, et où il est également question de l'explosion de l'intelligence (artificielle) et du danger de la singularité technologique conduisant à un changement d'ère, ou de paradigme totalement en dehors du contrôle humain.
- En 2009, le Massachusetts Institute of Technology (MIT) a lancé un projet visant à repenser la recherche en intelligence artificielle. Il réunira des scientifiques qui ont eu du succès dans des domaines distincts de l'IA. Neil Gershenfeld déclare « Nous voulons essentiellement revenir 30 ans en arrière, et de revoir quelques directions aujourd'hui gelées ».
- En novembre 2009, l'US Air Force cherche à acquérir 2 200 PlayStation 340[réf. obsolète] pour utiliser l processeur cell à 7 ou 8 cœurs qu'elle contient dans le but d'augmenter les capacités de leur superordinateur constitué de 336 PlayStation 3 (total théorique 52,8 petaFLOPS en double précision). Le nombre sera réduit à 1 700 unités le 22 décembre 200941. Le projet vise le traitement vidéo haute-définition, et l'« informatique neuromorphique », ou la création de calculateurs avec des propriétés /fonctions similaires au cerveau humain.

Années 2010

- Le 27 janvier 2010, l'US Air Force demande l'aide de l'industrie pour développer une intelligence avancé de collecte d'information et avec la capacité de décision rapide pour aider les forces américaines pour attaquer ses ennemis rapidement à leurs points les plus vulnérables. L'US Air Force utilisera une intelligence artificielle, le raisonnement ontologique, et les procédures informatique basées sur la connaissance, ainsi que d'autres traitements de données avancés afin de frapper l'ennemi au meilleur point. D'autre part, d'ici 2020, plus de mille bombardiers et chasseurs F-22 et F-35 de dernière génération, parmi plus de 2 500 avions militaires, commenceront à être équipés de sorte que, d'ici 2040, tous les avions de guerre américains soient pilotés par intelligence artificielle, en plus des 10 000 véhicules terrestres et des 7 000 dispositifs aériens commandés d'ores et déjà à distance.

- Le 16 février 2011, Watson, le superordinateur conçu par IBM, remporte deux des trois manches du jeu télévisé Jeopardy! en battant largement ses deux concurrents humains en gains cumulés. Pour cette IA, la performance a résidé dans le fait de répondre à des questions de culture générale (et non un domaine technique précis) dans des délais très courts. En février 2016, l'artiste et designer Aaron Siegel propose de faire de Watson un candidat à l'élection présidentielle américaine afin de lancer le débat sur « le potentiel de l'intelligence artificielle dans la politique ».
- En mai 2013, Google ouvre un laboratoire de recherches dans les locaux de la NASA. Grâce à un super calculateur quantique conçu par D-Wave Systems et qui serait d'après cette société 11 000 fois plus performant qu'un ordinateur actuel (de 2013), ils espèrent ainsi faire progresser l'intelligence artificielle, notamment l'apprentissage automatique. Raymond Kurzweil est engagé en décembre 2012 par Google afin de participer et d'améliorer l'apprentissage automatique des machines et des IA.
- Entre 2014 et 2015, à la suite du développement rapide du deep learning, et à l'encontre des penseurs transhumanistes, quelques scientifiques et membres de la communauté high tech craignent que l'intelligence artificielle ne vienne à terme dépasser les performances de l'intelligence humaine. Parmi eux, l'astrophysicien britannique Stephen Hawking, le fondateur de Microsoft Bill Gates et le PDG de Tesla Elon Musk.
- Les géants de l'Internet s'intéressent de plus en plus à l'IA. Le 3 janvier 2016, le patron de Facebook, Mark Zuckerberg, s'est donné pour objectif de l'année de « construire une intelligence artificielle simple pour piloter ma maison ou m'aider dans mon travail ». Il avait déjà créé en 2013 le laboratoire Facebook Artifical Intelligence Research (FAIR) dirigé par le chercheur français Yann Le Cun et ouvert un laboratoire de recherche permanente dans le domaine à Paris.
- Apple a de son côté récemment acquis plusieurs start-up du secteur (Perceptio, VocalIQ, Emotient et Turi).
- En janvier 2018, des modèles d'intelligence artificielle développés par Microsoft et Alibaba réussissent chacun de leur côté à battre les humains dans un test de lecture et de compréhension de l'université Stanford. Le traitement du langage naturel imite la compréhension humaine des mots et des phrases et permet maintenant aux modèles d'apprentissage automatique de traiter de grandes quantités d'informations avant de fournir des réponses précises aux questions qui leur sont posées.
- En février 2019, l'institut de recherche OpenAI annonce avoir créé un programme d'intelligence artificielle capable de générer des textes tellement réalistes que cette technologie pourrait être dangereuse. Si le logiciel est utilisé avec une intention malveillante, il peut générer facilement des fausses nouvelles très crédibles. Inquiet par l'utilisation qui pourrait en être faite, OpenAI préfère ne pas rendre public le code source du programme.

4. Qu'avez-vous retenu ?

Exercice *[solution* n°1 p.42*]*

Complétez les trous du texte suivant :

- L'intelligence artificielle (IA) est mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence.
- Souvent classée dans le groupe des , elle fait appel à la neurobiologie computationnelle (particulièrement aux), à la logique mathématique (partie des mathématiques et de la philosophie) et à l'informatique.
- Si les progrès de l'intelligence artificielle sont récents, ce thème de réflexion est tout à fait ancien, et il apparaît régulièrement au cours de l'histoire. Les premiers signes d'intérêt pour une intelligence artificielle et le principal précurseur de cette discipline est .

* *

*

Ce premier chapitre vous a présenté une introduction générale sur les principes et les applications de l'intelligence artificielle où une définition et un peu d'historique sont données.

Chapitre 2 : L'IA et la résolution des problèmes

II

- **La résolution de problèmes** est le processus qui consiste à sélectionner et ordonnancer des **actions élémentaires** en séquences afin d'atteindre des **buts donnés** en respectant les **contraintes** d'un environnement donné.
- **Composition du processus** : un état initial (le problème à résoudre), un état final (la solution au problème), un ensemble d'actions élémentaires permettant de passer d'un état à un autre.
- **Objectif du processus** : Trouver une séquence d'actions à exécuter pour passer de l'état initial à l'état final
- **Complexité du processus** : A partir d'un état donné, on peut effectuer plusieurs actions élémentaires différentes qui débouchent sur des états différents.
- La méthode qui consiste à essayer successivement toutes les possibilités jusqu'à trouver une solution est une méthode de recherche fortement **combinatoire**
- On est souvent amené à choisir entre plusieurs choix ou à choisir la meilleur solution : **problème d'optimisation combinatoire**
- Apport de l'IA : L'IA apporte une solution à ce problème :
    - les **heuristiques** : méthodes permettant de choisir en premier les actions ayant le plus de chance d'aboutir

1. Types de domaines

Types de domaines

- La démonstration de théorèmes
- Les jeux de stratégies (échec, dame)
- La planification :
    - robotique
    - allocations de ressources
    - emploi du temps

Exemple : Exemple de problème - Planification de chaîne de production

- Chaîne de production : Atelier A, Atelier B, Atelier C, Atelier D
- Capacité de production :

  - 5v/h 3v/h 2v/h 3v/h
  - Op1 : 8h-13h
  - Op2 : 8h-13h
  - Op3 : 9h-13h
  - Op4 : 10h-15h

- Ressources 30v en attente à 8h 0v en attente à 8h 0v en attente à 8h 0v en attente à 8h
- État initial
- Établir l'emploi du temps indiquant heure par heure quels opérateurs sont affectés à chaque atelier, de telle sorte que le maximum de voitures soient passées par les 4 ateliers à la fin de la journée

Exemple : Exemple de problème - Le voyageur de commerce

- Un voyageur de commerce partant d'une ville veut se rendre successivement dans un certain nombres de villes sans repasser deux fois dans la même ville et en revenant finalement à la ville de départ.
- On connaît les distances de ville à ville. Quel trajet doit-il effectuer de façon à minimiser la distance parcourue ?
- **EXPLOSION COMBINATOIRE** : $(n-1)\ !/2$ configurations
- configurations pour n=10 est 362880/2

2. Complexité

Complexité d'un algorithme

- Complexité est une mesure de temps (**complexité en temps**) ou d'espace mémoire (**complexité en espac**) utilisée par un algorithme en fonction de la taille des données
- La complexité en temps d'un algorithme est le nombre d'étapes de calcul qu 'il nécessite, et sa complexité en espace est la taille de mémoire nécessaire pour stocker les données intermédiaires au cours de son exécution
- Ces grandeurs sont indiquées en fonction de la taille des données du problème à résoudre, et généralemen à une constante de proportionnalité près
- Seul l'ordre de grandeur du nombre d'opérations, noté O, est utilisé pour exprimer la complexité : $O(f(n))$

Principales classes de complexité d'algorithmes

- **complexité logarithmique** : $O(log(n))$ exemple : recherche dichotomique dans un tableau trié
- **complexité linéaire** : $O(n)$ exemple : recherche du minimum d 'une liste de n éléments, calcul du produit scalaire de deux vecteurs de n éléments
- **complexité polynomiale** : $O(n^k)$ (k : une constante > 1) exemple : multiplication de deux matrices carrées d'ordre n : $(O(n^3))$
- **complexité exponentielle** :$O(a^n)$ (n : une constante > 1) exemple : voyageur de commerce

Complexité d'un problème

- La complexité (maximale) du meilleur algorithme connu pour le résoudre
- Classification des problèmes

  1. La **classe P** Classe des problèmes qu 'on peut résoudre en temps polynomial

2. La **classe E** Classe des problèmes qu 'on peut résoudre en temps exponentiel
3. La **classe NP** Classe des problèmes qu 'on peut vérifier en temps polynomial et résoudre en temps exponentiel (problèmes non déterministes polynomiaux)

3. Représentation des problèmes

Espace d'états

- Un problème est défini par un **espace d'états** qui est l'ensemble des états possibles de ce problème
- Exemples d 'états :
    - Une configuration de pièces dans un jeu d 'échecs
    - Un calendrier partiel ou complet des matches de tennis
    - Une partie de preuve ou une preuve complète d'un démonstrateur de théorèmes

Opérateurs

- Les états d'un problème sont reliés les uns aux autres par des **opérateurs** qui transforment un état dans un autre
- Exemples d'opérateurs :
    - Un coup légal aux échecs qui transforme la configuration des pièces dans une autre
    - L'ajout d'un match de tennis dans le problème de planification
    - L'application d'une règle d'inférence dans le démonstrateur de théorèmes

But

- Dans la résolution d'un problème, on a un **but** qui décrit ce que l'on cherche.
- Un but est un sous-ensemble d'états de l'espace d'états
- Exemples de buts :
    - Dans le jeu d'échecs, le but est l'ensemble des configurations de pièces qui mettent votre adversaire **échec et mat**
    - Pour le démonstrateur de théorèmes ce sont toutes les preuves dont **la dernière ligne** correspond à la formule à prouver

Graphe d'états et espace de recherche

- En IA, on utilise les **graphes** pour représenter l'espace d'états d'un problème (graphe d'états)
- Chaque **nœud** représente un état, chacun des **arcs** représente un opérateur (une transition) faisant passer d'un état à un autre
- Le but est un nœud particulier du graphe
- La résolution d'un problème revient à explorer le graphe dans la recherche du **nœud but** (l'espace de recherche)

Réduction de problèmes et graphe ET/OU

- Lorsque on a un problème principal qui peut se décomposer en sous problèmes plus faciles à résoudre, résoudre ce problème principal revient à résoudre tous les **sous problèmes** qu'ils le composent

- La technique qui consiste à résoudre un but en résolvant ses sous-buts est appelé **réduction de problème**
- Certains buts ne peuvent être atteints que si leurs sous-buts immédiats le sont tous (**nœud ET**)
- Les autres buts sont atteints si l'un au moins de leurs sous-buts immédiats est atteint (**nœud OU**)
- Le graphe formé de nœuds ET et OU est appelé **graphe ET/OU**

Composition d'un système de résolution

- Des structures de données organisés en arbre ou en graphe
- Des opérateurs définis par leurs conditions d'application et leur action
- Une structure de contrôle mettant en œuvre la stratégie de recherche dans l'arbre ou le graphe (méthodes de recherche ou de résolution)

4. Raisonnement

Caractéristiques

1. Le système doit-il trouver une solution à partir de ce qu'il connaît du problème ou peut-il y avoir une interaction avec l'utilisateur
2. Faut-il se contenter d'une solution sous-optimale mais plus facilement et rapidement accessible
3. Peut-on annuler une action décidée au cours de la réflexion une fois qu'elle a été exécutée
4. Est-ce qu'on peut prévoir exactement l'effet d'une action sur l'état courant du problème
5. Est-ce que le problème peut se décomposer en sous problèmes plus faciles à résoudre et est-ce que les sous problèmes sont indépendants ou non.
6. Quelles sont les connaissances minimales requises pour aboutir à la solution
7. Est-ce qu'une solution est garantie ?, est-ce que la recherche terminera ? quelle est la complexité de la recherche en temps et en espace ?

Méthodes de recherche

- **Méthodes aveugles** : énumération exhaustive de tous les états de l'espace de recherche
    1. Recherche en profondeur
    2. Recherche en largeur
    3. Recherche en profondeur limitée
    4. Recherche en approfondissement itératif
- **Méthodes heuristiques** : construction de **chemins minimaux** dans l 'exploration de l 'espace de recherche basée sur des critères, des principes, ou des méthodes permettant de faire **les choix les plus efficaces** pour atteindre le but fixé
    1. Recherche en profondeur ordonné
    2. Recherche du meilleur d 'abord

Critères d'évaluation des méthodes de recherche

- Complétude : solution garantie si elle existe
- Optimalisé : meilleur solution garantie
- Complexité en espace : espace mémoire nécessaire pour effectuer la recherche
- Complexité en temps : temps nécessaire pour effectuer la recherche

- La complexité est mesurée selon les 3 critères :
    1. b facteur de branchement (nombre de descendants/nœuds) maximum du graphe d'états
    2. d profondeur à la quelle se trouve le (meilleur) nœud solution
    3. m profondeur maximale du graphe

5. Méthodes de recherche aveugle

Profondeur d 'abord (LIFO)

Le principe est que la priorité est donnée aux nœuds de niveaux les plus profonds du graphe de recherche

```
Algorithme :
Début
   empiler la racine
   répéter Si sommet de pile <> nœud but Alors
        dépiler le premier élément
        empiler ses fils s'il en a du plus à droite au plus à gauche
     Sinon ne rien faire Jusqu'à ce que la pile soit vide ou sommet pile = nœud but
   la recherche aboutit si le nœud but a été atteint.
Fin
```

Propriétés :

- Complétude : Non si m tends vers l'infini
- Complexité en temps : $O(b^m)$
- Complexité en espace : $O(b * m)$
- Optimalité : Non

Largeur d'abord (FIFO)

Le principe est que la priorité est donnée aux nœuds de niveaux les moins profonds du graphe de recherche

```
Algorithme :
Début
     empiler la racine
     répéter Si sommet de pile <> nœud but Alors
          dépiler le premier élément
          empiler ses fils s'il en a du plus à gauche au plus à droite
        Sinon ne rien faire Jusqu'à ce que la pile soit vide ou sommet pile = nœud but
     la recherche aboutit si le nœud but a été atteint.
Fin
```

Propriétés :

1. Complétude : Non si b tends vers l'infini
2. Complexité en temps : $O(b^d)$
3. Complexité en espace : $O(b^d)$
4. Optimalité : Non

Exemple:

- Si $b = 10$
- production de nœuds : 1000 /s

- espace mémoire : 100 octet/noeud

Profondeur limitée

Principe : idem que profondeur d 'abord avec une limite de profondeur d 'exploration L

```
Algorithme :
Début
    empiler la racine
    répéter Si sommet de pile <> nœud but Alors
        dépiler le premier élément
        empiler ses fils s'il en a du plus à droite au plus à gauche
      Sinon ne rien faire Jusqu'à ce que la pile soit vide ou sommet pile = nœud but
            ou limite profondeur atteinte
    La recherche aboutit si le nœud but a été atteint.
Fin
```

Propriétés :

- Complétude : Oui si $L \geq d$
- Complexité en temps : $O(b^L)$
- Complexité en espace : $O(b * L)$
- Optimalité : Non

Approfondissement itératif

Le principe est la recherche en profondeur, limitée à la profondeur 1, puis 2, ... jusqu'à l'obtention d'une solution (ou l'échec de la recherche)

Propriétés :

- Complétude : Oui
- Complexité en temps : $O(b^d)$
- Complexité en espace : $O(b * d)$
- Optimalité : Non

Conclusions

- Les recherches en largeur garantissent de trouver la solution (si elle existe) et de trouver le chemin le plus court au but
- Les recherches en largeur demandent un espace mémoire qui augmente exponentiellement, car elles gardent tous les enfants au même niveau
- Les recherches en profondeur sont très efficaces dans les cas où le but est loin de la racine, et les branche sont arrangées de gauche à droite par probabilité de solution
- Les recherches en profondeur peuvent se perdre dans une branche sans fin et parfois se retrouve en boucl infinie
- Les recherches aveugles sont caractérisées par une taille souvent rédhibitoire des arborescences et une trè forte complexité des algorithmes

6. Méthodes de recherche heuristique

Fonction heuristique

$h : E \rightarrow Natural$ un état $e \in E$ (espace d'états) --> un nombre $h(e) \in Natural$:

- $h(e)$ est (généralement) une estimation du rapport coût/bénéfice qu'il y a à étendre le chemin courant en passant par e.
- contrainte : $h(solution) = 0$

Exemple de fonction heuristique :

- Problème des 8 reines
- Heuristique : laisser un plus grand nombre de cases non attaquables dans la partie restante de l'échiquier pour garder le plus de choix possibles pour les ajouts de reines futures

Profondeur ordonnée

Le principe est l'approfondissement des branches jusqu'à leur extrémité, dans l'ordre inverse de leur distance au but

```
1 Algorithme :
2 Début
3      empiler la racine
4      répéter Si sommet de pile <> nœud but Alors
5           dépiler le premier élément
6           empiler ses fils s'il en a dans l 'ordre inverse de leur distance au but

7         Sinon ne rien faire Jusqu'à ce que la pile soit vide ou sommet pile =
  nœud but
8      la recherche aboutit si le nœud but a été atteint
9 Fin
```

Meilleur premier

Le principe est dès que l'on rencontre un chemin meilleur qu'auparavant, c'est celui là que l 'on approfondit

Heuristique : proximité du but

```
1 Algorithme :
2 Début
3    empiler la racine
4    répéter Si sommet de pile <> nœud but Alors
5         dépiler le premier élément
6         empiler ses fils s'il en a dans la file et trier la pile entière par ordre
7            croissant des distances calculées au but
8        Sinon ne rien faire Jusqu'à ce que la pile soit vide ou sommet pile = nœud
  but
9    la recherche aboutit si le nœud but a été atteint
10 Fin
```

Propriétés :

Complétude : Non

Complexité en temps : $O(b^m)$

Complexité en espace : $O(b^m)$

Optimalité : Non

*Algorithme A**

Le principe est d'améliorer l'algorithme du **Meilleur premier** en calculant le plus court chemin pour passer d'un état initial à un état final. Soient :

- n_0 : la racine du graphe
- G : le nœud solution
- $c(n_i, n_j)$: le coût pour aller du nœud n_i au nœud n_j

On définit

- $g(n) = c(n_0, n)$: le coût pour aller du nœud racine n_0 au nœud n
- $h(n) = c(n, G)$: l 'estimation heuristique du coût pour aller du nœud n au nœud solution G

On obtient alors la fonction d 'évaluation du coût du chemin en passant par le nœud n : $f(n) = g(n) + h(n)$

Définitions :

Heuristique consistante : Pour tout x, pour tout y descendant de x : $h(x) \leq h(y) + c(x, y)$

Heuristique admissible : Pour tout $x, h(x) \leq h^*(x)$ où $h^*(x)$ est la valeur optimale de x à G

```
Algorithme A*:
Début
   Initialement, la file des chemins partiels contient le chemin d'ordre zéro,
   de longueur nulle et reliant la racine à nulle part
   Repeter
     Si le premier chemin de la file n'aboutit pas au but Alors le supprimer
      de la file, former les nouveaux chemins obtenus en prolongeant d'un niveau
      le chemin supprimé, insérer ces nouveaux chemins dans la file le coût
      de chaque chemin étant la somme de la distance déjà parcourue et
      d'une estimation minorante de la distance restant à parcourir.
      trier la file par coûts croissants si deux chemins ou plus atteignent
      le même noeud, ne conserver que celui ayant la longueur minimale
     Finsi
   Jusqu'à ce que la file soit vide ou que le noeud but soit atteint
```

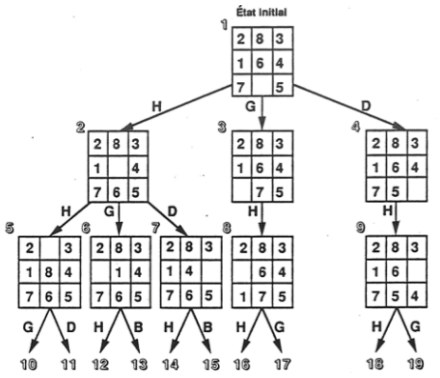
Propriétés :

- Complétude : Oui
- Complexité : linéaire si h est consistante
- Optimalité : Oui si h est admissible

Exemple : Prenons un exemple pour clarifier ces idées

- Le taquin à 8 tuiles est un casse-tête formé de tuiles numérotées de 1 à 8 que l'on peut déplacer dans une grille 3 x 3 comportant une case vide.
- Partant d'une situation initiale où les tuiles sont en désordre, le but du jeu est de les replacer en cercle dan l'ordre suivant : 1, 2 et 3 sur la première ligne, 8, case vide et 4 sur la deuxième et 7, 6 et 5 sur la troisième.
- Ayant décrit l'état final et l'état initial donnés dans l'énoncé du problème, il reste à constater à la figure qu'il y a quatre opérateurs possibles :

1. H : déplacer la tuile du haut dans la case vide;
2. G : déplacer la tuile de gauche dans la case vide;
3. D : déplacer la tuile de droite dans la case vide;
4. B : déplacer la tuile du bas dans la case vide



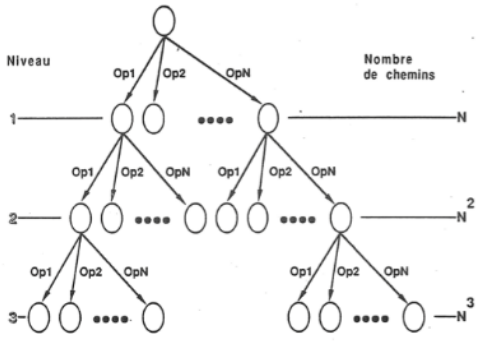
Une démarche de résolution du problème du taquin à 8 tuiles

Exemple : Une approche mécanique

Une façon classique de programmer la solution de ce problème consiste :

- à appliquer à l'état initial tous les opérateurs dans l'ordre H, G, D, B;
- à appliquer à chacun des quatre états à leur tour tous les opérateurs qui ne produisent pas une configuration déjà rencontrée;
- à procéder ainsi jusqu'à ce qu'une solution ait été atteinte ou que tous les chemins possibles aient été générés.

Cette approche mécanique a l'avantage d'être systématique. Toutefois, dès que le problème est quelque peu difficile, il faut abandonner l'idée d'une énumération exhaustive de tous les chemins.



L'explosion combinatoire du nombre de chemin

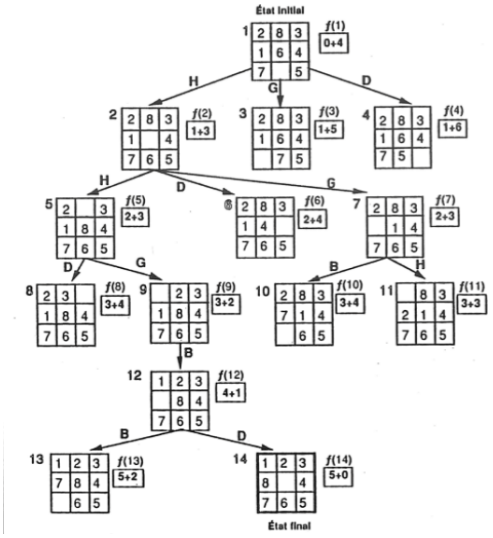
- Règle générale :
  - Si à un état, on peut appliquer N opérateurs, après a niveaux, on aura généré un nombre de chemins égal à N^a.
- Voilà très précisément le problème de l'explosion combinatoire ou exponentielle évoqué précédemment e comme le montre la figure.
- Par exemple, au début d'une partie d'échecs, s' il y a 30 coups possibles en moyenne dans chaque position après 10 coups joués, il y aura $30^{10} = 590490000000000$ chemins qui auront été générés!
- Voilà qui peut excéder rapidement la capacité même d'un ordinateur très puissant.

- Il faut alors trouver des moyens d'orienter la recherche dans des directions fructueuses, en augmentant « l'intelligence » du programme.

Exemple : Une approche heuristique

- Une façon d'augmenter l'intelligence d'un programme est de le doter d'une méthode d'évaluation des états qui associe à chacun une mesure de sa probabilité de conduire vers une solution, un peu comme le ferait un joueur humain.
- Pour chaque état, le programme examinera les valeurs des états successeurs obtenus en appliquant tous les opérateurs possibles et choisira de développer uniquement l'état ayant la meilleure valeur.
- Une telle méthode est basée sur le raisonnement suivant : « moins il y a de tuiles mal placées après l'application de l'opérateur, plus je suis sur la bonne voie; par contre, si j'ai joué beaucoup de coups depuis le début, je dois avoir fait fausse route ».
- Sur la figure, on a donné des numéros aux différents états E et on calcule pour chacun la fonction d'évaluation heuristique suivante :
  $f(E) =$ Nombre de coups depuis l'état initial + Nombre de tuiles mal placées.
- Voyons de plus près comment on utilise une telle fonction.
- Dans la figure nous avons utilisé l'algorithme A*.
- On considère cet algorithme comme une méthode générale pour faire une recherche dans un arbre des possibilités.
- Rendu à un état quelconque dans l'arbre, on examine tous les opérateurs applicables et on choisit de n'appliquer que celui qui produira la plus petite valeur de la fonction d'évaluation.

Un exemple de fonction heuristique



Cette méthode est efficace dans la mesure où la fonction d'évaluation mesure bien la valeur de chacun des états. Dans le problème du taquin, elle va presque droit au but et trouve une solution en développant seulement 12 états intermédiaires.

7. Qu'avez-vous retenu ?

Exercice *[solution* n°2 p.42*]*

Choisir par glissement de la réponse correcte

construction de chemins minimaux dans l 'exploration de l 'espace de recherche

énumération exhaustive de tous les états de l'espace de recherche

Méthodes aveugles	Méthodes heuristiques

Exercice *[solution* n°3 p.42*]*

Choisir par glissement de la bonne réponse

temps nécessaire pour effectuer la recherche

profondeur maximale du graphe

branchement maximum du graphe d'états

espace mémoire nécessaire pour effectuer la recherche

Meilleur solution garantie

profondeur à la quelle se trouve le nœud solution

Solution garantie si elle existe

Complétude	Optimalisé	Complexité en espace	Complexité en temps	Critère b	Critère d	Critère m

* *

*

Ce deuxième chapitre a présenté un aperçu sur l'apport de l'IA pour la résolution des problèmes

Chapitre 3 : Les systèmes à base de connaissances

III

L'IA est la branche de l'informatique qui consiste à concevoir des systèmes intelligents, c'est-à-dire qui soient capables de produire un raisonnement proche de celui de l'être humain.

1. Définitions

Différence entre donnée, information et connaissance

- Une donnée transporte l'information (un signal non interprété).
- L'information est une interprétation de la donnée (par exemple, la donnée "!" transporte l'information (interprétation) "point d'exclamation"),
- La connaissance utilise l'information dans le cadre d'actions, dans un but précis (par exemple, écrire un “!” pour marquer une exclamation en fin de phrase).
- Les actions peuvent être la prise de décisions, la création de nouvelles informations, etc.

Base de connaissances

- C'est un ensemble de connaissances dédié à un domaine particulier (c'est un ensemble de faits et de règles

Système à base de connaissances

- Un système à base de connaissances (SBC) est un programme construit pour modéliser les compétences de résolution de problèmes des humains.
- C'est à dire, un système informatique fonctionnant avec une base de connaissances explicite sur un sujet donné.

Système expert

- Système informatique permettant de résoudre les problèmes dans un domaine d'application déterminé à l'aide d'une base de connaissances établie à partir de l'expertise humaine.
- C'est à dire un système expert classique est un cas particulier d'un système à base de connaissances (tout système informatique utilise de la connaissance, mais dans un système à base de connaissances, celle-ci est représentée de façon explicite.).

C'est un logiciel jouant le même rôle qu'un expert humain, c' est-à- dire capable, à partir de ses connaissances spécialisées, en appliquant des méthodes d'inférence, de fournir des conseils relatifs à son domaine d'expertise. Un expert doit :

1. Posséder un savoir spécialisé qui regroupe de vastes connaissances dans un domaine donné. Ce savoir ne se limite pas à des connaissances factuelles, mais il comprend un réseau de concepts, de règles, de méthodes qui forment l'expertise.
2. Savoir utiliser les connaissances efficacement.Il ne suffit pas à l'expert de posséder de vastes connaissances, il doit pouvoir les utiliser rapidement et avec succès. Pour cela,il doit savoir poser les bonnes questions au client de façon à repérer les connaissances qui s' appliquent dans son cas.
3. Connaître ses limites de façon à ne pas s' occuper de questions hors du cadre de sa compétence.
4. Fournir des explications appropriées sur les informations qu'il donne ou les solutions qu'il propose. Généralement, ce n'est pas l'expert qui décide, mais celui qui demande conseil. Il est donc un conseiller qui doit être capable de convaincre le client que sa solution est bonne.

La consultation d'un expert est donc essentiellement un dialogue où celui-ci pose des questions de plus en plus précises pour identifier les connaissances dont il a besoin et où il explique les raisons de ses recommandations. Dans un tel dialogue, le client fournit des faits et pose des questions pour bien comprendre le sens de la solution proposée par l'expert. À la fin, il lui revient de prendre la décision finale

Les inférences

- Avec un moteur d'inférence on peut déduire des faits à partir de faits initiaux et des règles.
- Deux approches de base :
    1. A partir de ce qu'on veut trouver et remonter vers les faits (chaînage arrière) ;
    2. A partir des faits et aller vers ce qu'on veut trouver (chaînage avant).
- La structure classique des systèmes experts utilisaient toujours un ensemble de règles de production.

Mécanismes de raisonnement

- Pour les inférences, la plupart des systèmes à base de connaissances (systèmes experts) existants reposent sur des mécanismes de logique formelle et utilisent le raisonnement déductif.
- Pour l'essentiel, ils utilisent le syllogisme : si P est vrai (fait ou prémisse) et si on sait que $P \rightarrow Q$ (règle) alors Q est vrai (nouveau fait ou conclusion).
- Les plus simples des systèmes experts s'appuient sur la logique des propositions (dite aussi « logique d'ordre 0 »).
- Dans cette logique, on n'utilise que des propositions, qui sont vraies, ou fausses.
- D'autres systèmes s'appuient sur la logique des prédicats du premier ordre (dite aussi « logique d'ordre 1 »), que des algorithmes permettent de manipuler aisément.

2. Architecture des systèmes à base de connaissances

Architecture

- Un système expert est un outil capable de reproduire les mécanismes cognitifs d'un expert, dans un domaine particulier.

- Plus précisément, un système expert est un logiciel capable de répondre à des questions, en effectuant un raisonnement à partir de faits et de règles connus.
- Il peut servir notamment comme outil d'aide à la décision.
- Dans tout système à base de connaissances ou système expert, on retrouvera à la base les composantes suivantes :

    1. Une base de connaissances composée d'une base de faits et une base de règles ;
    2. Un moteur d'inférence : Le moteur d'inférence est capable d'utiliser faits et règles pour produire de nouveaux faits, jusqu'à parvenir à la réponse à la question posée.
    3. En plus à une interface

1. Base de connaissances

- La base de connaissances est propre au domaine traité et, contrairement aux bases de données, n'est pas limitée aux connaissances de type factuel, aux données.
- Elle rassemble tous les types de connaissances pertinentes pour le domaine considéré : description des objets et de leurs relations, règles à appliquer pour résoudre un problème, méta-connaissances permettant de choisir quelles règles appliquer, etc.
- D'autre part, le système conserve à tout moment dans sa mémoire de travail, une base des faits connus.
- Grâce à elle, le moteur d'inférence peut choisir les éléments de sa base de connaissances; par exemple, les règles à utiliser en fonction des faits disponible, de manière efficace et adaptée au problème posé.
- La base de faits s'enrichit au fur et à mesure que le moteur d'inférence déduit de nouveaux faits en appliquant des connaissances aux faits déjà connus.

2. Moteur d'inférence

- Le moteur d'inférence est le programme qui construit les raisonnements en puisant ses matériaux dans la base de connaissances et la base de faits.
- En examinant la base de faits, il détecte les connaissances intéressantes pouvant s'appliquer à certains faits, les enchaîne et construit un plan de résolution.
- Il déduit de nouveaux faits de ceux fournis au départ ou en cours d'interaction par l'usager.
- Indépendant du domaine, le moteur d'inférence rassemble les mécanismes de raisonnement qui vont exploiter la base de connaissances.

3. Interface

- À ces deux composantes essentielles s' ajoutent des modules d'interface eux aussi indépendants du domaine de connaissance.
- Grâce à eux, l'expert peut accéder facilement à la base de connaissances, la modifier en rectifiant une information inutile ou erronée, ou en ajoutant une précision.
- L'utilisateur peut quant à lui suivre le raisonnement du système dans un langage qui lui est naturel, poser des questions, demander des explications, sans avoir besoin d'acquérir une connaissance approfondie des systèmes experts ou de l'informatique.

Séparation entre les connaissances spécialisées et les règles

- Séparation de la connaissance et du raisonnement : les systèmes à base de connaissances, tout comme les systèmes experts, sont fondés sur une séparation nette entre les connaissances et les méthodes d'inférence.
- Ainsi les connaissances, au lieu d'être imbriquées dans la structure du programme sont considérées comm des données interchangeables, susceptibles d'être corrigées, mises à jour et exploitées par des programmes (les moteurs d'inférence) qui, eux, simulent les mécanismes de raisonnement et assurent l'interface avec les utilisateurs.
- L'idée de distinguer les connaissances spécialisées des règles agissant sur cet ensemble a permis plusieurs progrès importants dans certains domaines de l'intelligence artificielle.
- La technologie des systèmes experts a notamment profité de cette distinction en permettant le développement de moteurs d'inférence indépendants d'un domaine.
- Ces logiciels contiennent une représentation des connaissances générales nécessaires au traitement de règles et de faits et permettent à un utilisateur de se concentrer uniquement sur la construction d'une base de connaissances spécialisées.

Inférence et résolution de problèmes

- Résoudre des problèmes est une des fonctions fondamentales de l'intelligence artificielle qui exige une bonne représentation des connaissances et des méthodes d'inférence dont l'application permettra de trouver la solution.
- On construit ainsi une base de connaissances qui, au départ, contiendra une description de l'état initial. La base de connaissances contiendra également des opérations exprimées par des règles permettant de transformer cet état initial en d'autres états qui, petit à petit, nous rapprocherons de l'état final ou solution.
- **Heuristique** : Technique consistant à apprendre petit à petit, en tenant compte de ce que l'on a fait précédemment pour tendre vers la solution d'un problème.

Exemple : Le problème des seaux à remplir

- Prenons le problème suivant : deux seaux A et B contiennent respectivement 9 et 4 litres.
- Une personne a un tonneau de douze litres d'huiles ; elle veut en donner 3 litres à un ami. Pour les mesurer, elle n'a que ces deux seaux A et B, A contient 9 litres et B contient 4 litres. Comment doit-elle opérer pour avoir les 3 litres dans le seau A?
- Peut-on obtenir exactement 3 litres dans le contenant A, uniquement en remplissant ou en vidant les seau ou en les versant l'un dans l'autre?

On peut représenter les connaissances nécessaires pour résoudre ce problème à l'aide d'états et d'opérateurs comme suit :

- les états sont de la forme [A] [B] où A et B sont le nombre de litres de chaque récipient; par exemple, [4] [5] signifie que le récipient A contient 4 litres et le récipient B, 5 litres.
- les opérateurs sont représentés par les règles SI... ALORS de transformation entre états permises par l'énoncé du problème tel qu'indiqué au dessous.

```
Remplir(A) = Si [A][B] Alors [9][B]
Remplir(B) = Si [A][B] Alors [A][4]
Vider(A) = Si [A][B] Alors [0][B]
```

```
Vider(B) = Si [A][B] Alors [A][0]
Verser(B dans A) = Si [A][B] et A+B <= 9 Alors [A+B][0]
Verser(B dans A) = Si [A][B] et A+B > 9 Alors [9][A+B-9]
Verser(A dans B) = Si [A][B] et A+B<=4 Alors [0][A+B]
Verser(A dans B) = Si [A][B] et A+B>4 Alors [A+B-4][4]
```

Signification des opérateurs

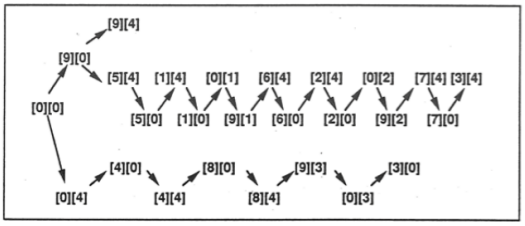
- La présence de deux opérateurs pour le versement tient compte du fait que le calcul sera différent si on peut verser complètement un récipient dans un autre ou seulement une partie.
- Par exemple, l'avant-dernier opérateur signifie que si le total des deux récipients est inférieur ou égal à 4 litres, on peut verser tout le contenu de A dans B, lequel contient alors A + B litres, alors que A est devenu vide.
- Dans le cas contraire, l'autre opérateur affirme que l'on pourra remplir B jusqu'à 4 litres et que dans A, on ajoutera B - 4 litres à A qui contiendra alors A + B - 4 litres.

Inférence

- Pour utiliser ces connaissances, le programme devra contenir un moteur d'inférence qui « instancie » les variables A et B c'est-à-dire leur donne des valeurs particulières correspondant à un état déjà obtenu.
- Après cette opération, le moteur peut appliquer la règle de modus ponens et utiliser ces opérateurs comme des énoncés en logique propositionnelle.

Résolution du problème

- Le problème se présente donc comme une suite d'états obtenue en appliquant une suite d'opérateurs (instanciés correctement) à partir de l'état initial jusqu'à l'état final soit [0][0]; puis remplir A [9][0]; puis vider A dans B [5][4].
- Le problème « obtenir 3 litres dans A » n'est pas aussi simple et nécessite d'essayer divers chemins en appliquant tous les opérateurs possibles à une situation avant de trouver un chemin-solution
- La résolution de ce problème est un graphe (arbre) : les nœuds représentent les états et les arcs représentent les opérations comme en témoigne la figure.



Démarche de résolution

- Cette démarche peut être faite par un moteur d'inférence capable d'instanciation de variables.
- À partir d'un fait initial [0][0], on applique systématiquement les règles permises (les opérateurs) en déduisant toujours des faits nouveaux jusqu'à déduire la solution, un fait du type [3][B].
- On s' aperçoit qu'il y a plusieurs façons de parcourir l'arbre des possibilités. Notamment,l'ordre dans lequel on applique les opérateurs est critique.

- Ici, on perd beaucoup de temps à développer la « branche » [5][4], alors que la « branche » [0][4] conduit deux fois plus rapidement à une solution.
- L'étude des méthodes les plus efficaces de recherche dans un arbre des possibilités sert à la fois à trouver de bonnes méthodes de déduction ou d'inférence et de bonnes méthodes de résolution de problèmes.
- L'inférence, tout comme la résolution d'un problème, sont tous deux des mécanismes de recherche dans un arbre des possibilités qui peuvent être décrits ainsi :
    - En partant d'une situation initiale connue (les faits), atteindre une des situations finales (fait-conclusion, fait-solution) désirées en appliquant une suite finie d'opérateurs à la situation initiale ou aux situations déjà obtenues de celle-ci.

3. Ingénierie des connaissances

Définition

- L'ingénierie des connaissances est la gestion des connaissances au sein d'une organisation.
- Elle fait référence à l'ingénierie de systèmes complexes intelligents incorporant beaucoup de connaissances tels les systèmes à bases de connaissances et systèmes experts.

Dimensions

- L'exploitation des connaissances passe par cinq opérations : **identification**, **création**, **stockage**, **partage** et **utilisation**.
- L'ingénierie des connaissances se concentre sur l'identification, la création, le stockage et la mise à disposition des connaissances afin de rester neutre face aux outils de partage et d'utilisation.

Identification

- L'identification des connaissances consiste en l'identification des connaissances critiques pour une organisation.
- Elle peut s'effectuer de manière individuelle sur base de questionnaires, d'interviews, etc., ou de manière collective en identifiant les départements, services, équipes.

Collecte

La collecte peut consister en un transfert de connaissances ou en un partage de connaissances, le tout en structurant correctement l'information :

- Transfert de connaissances :
    - le tutorat ;
    - le parrainage.
- Partage de connaissances :
    - les communautés de pratique ;
    - l'inter-vision (partant d'un problème) ;
    - le focus groupe (petits groupes);
    - le retour d'expérience.

Structuration

- La structuration consiste à organiser les connaissances collectées en catégories, éventuellement en plusieurs niveaux hiérarchiques, pour en faciliter l'accès et la consultation.
- La structuration est une étape nécessaire pour pouvoir arriver à l'exploitation des données.

Exploitation

- La collecte et la structuration des données sont bien distinctes dans l'approche de la gestion des connaissances.
- Néanmoins, la collecte et la structuration ne suffisent pas pour parler d'une approche complète de gestion de connaissances.
- En effet, cette approche ne s'achève que lorsqu'on pourrait utiliser les données collectées et stockées pour atteindre les objectifs de l'organisme concerné.
- Ainsi, l'exploitation des données, connue aussi sous l'expression « Extraction des Connaissances à partir des Données » (ECD) (en anglais Knowledge Data Discovery, KDD) consiste à relier et à interpréter les faits pour en déduire des résultats et des conséquences utiles.
- L'extraction des connaissances à partir des données peut être réalisée suivant plusieurs méthodes, selon le domaine d'application et la nature des données brutes. Dans ce cadre, on distingue diverses méthodes

Méthode : Exploration de données

- L'exploration de données (aussi appelé fouille de données ou encore datamining pour les anglophones), es la pratique (par des moyens automatiques ou semi-automatiques) de la recherche et de l'exploration de grands ensembles de données ayant pour résultat la découverte de motifs significatifs et de règles.
- Pour faire cela, le data mining utilise des techniques informatiques empruntées à la statistique, et la reconnaissance de motifs récurrents dans de grandes masses de données récolté par un système d'information.
- Les résultats des analyses Data Mining ont pour but de connaître le comportement d'un usager, inférer puis prévoir son comportement.
- Ainsi, ce sont les résultats du datamining qui génèrent de la connaissance pour l'organisation (Knowledge Discovery). En entreprise, le data mining s'utilise principalement en:
    - Marketing: la connaissance du client, son comportement plus précisément est essentiel pour: mieu cibler les clients, les fidéliser, améliorer les forces de vente, améliorer la relation client.
    - Gestion des risques: principalement dans le secteur bancaire et assurantiel. Il s'agit de connaître des clients ou partenaires à risque et mesurer ce risque. Cette information est indispensable pour la banque ou l'assurance car constitue le cœur de son métier.

Méthode : intelligence artificielle

- L'intelligence artificielle peut se définir comme un traitement automatique des données basé sur des règle universelles pour la prise de décisions satisfaisantes au cas traité.
- Les méthodes d'ECD qui se basent sur l'intelligence artificielle ont évolué considérablement depuis l'apparition du web 2.0 et depuis le progrès de l'informatique notamment face à la multiplication des sources de données (Web, ERP, gestion de la relation client, etc.)

4. Domaines d'application

Pourquoi utiliser un système à base de connaissances

- Pour remplacer un expert par un programme
- Automatiser une tâche routinière nécessitant un expert
- Besoin d'une expertise dans un environnement hostile
- Assister un expert
- Améliorer la productivité
- Gérer la complexité

Les domaines d'applications

- Santé et IA,
- Agriculture et IA,
- Énergie hydrocarbure,
- énergie solaire + IA,
- Environnement (catastrophes naturelles (inondation, séisme, etc),...) + IA,
- Administration et IA,
- Défense nationale et IA,
- Transport et IA
- industrie et IA

Santé et IA : MYCIN

- Il s' attaque au problème du diagnostic et du traitement des maladies infectieuses du sang.
- Sa base de connaissances comprend à peu près 400 règles qui mettent en relations des états possibles et des interprétations correspondantes.
- Lorsqu'il résout un problème, MYCIN compare les conditions d'application d'une règle aux données disponibles ou encore, il demande des données additionnelles au médecin.
- Le cas échéant, il essaie de déduire la vérité ou la fausseté d'une condition à partir d'autres règles.

5. Exercices sur le troisième chapitre

Exercice : Donnée, Information et Connaissance *[solution n°4 p.43]*

Donner la classe de

si ahmed est une personne alors il est humain | ahmed est une personne | ahmed

donnée	information	connaissance

Exercice : L'objectif du langage de données *[solution n°5 p.43]*

Complétez les trous du texte suivant :

Un système à base de connaissances est composé de trois composants :

1. Une
2. Un
3. Une

Exercice : Base de connaissances *[solution n°6 p.43]*

Soit la base de connaissances suivante :

$\{pere(ahmed, mohamed), pere(mohamed, farid), pere(X, Z) \wedge pere(Z, Y) \rightarrow grandPere(X, Y)\}$

Donner le graphe ET/OU pour le but (question) $grandPere(X, Y)$

Exercice : Ingénierie de connaissances *[solution n°7 p.43]*

Remplir les trous suivants

L'exploitation des connaissances passe par cinq opérations : , , , et .

* *

*

Ce troisième chapitre a présenté un aperçu sur les systèmes à base de connaissances (SBC)

Chapitre 4 : Prolog, langage de l'IA

IV

Ce quatrième chapitre présente le langage Prolog comme langage de l'IA.

1. Le langage PROLOG

- Langage d'expression des connaissances fondé sur le langage des prédicats du premier ordre
- Programmation déclarative :
    - L'utilisateur définit une base de connaissances
    - L'interpréteur Prolog utilise cette base de connaissances pour répondre à des questions

Constantes et variables

- Constantes
    - Nombres : 12, 3.5
    - Atomes
        - Chaînes de caractères commençant par une minuscule
        - Chaînes de caractères entre " "
        - Liste vide []
- Variables
    - Chaînes de caractères commençant par une majuscule
    - Chaînes de caractères commençant par _
    - La variable « indéterminée » : _

TROIS SORTES DE CONNAISSANCES : FAITS , RÈGLES , QUESTIONS

- Faits : P(...). avec P un prédicat
    - pere(jean, paul).
    - pere(albert, jean).
    - Clause de Horn réduite à un littéral positif
- Règles : P(...) :- Q(...), ..., R(...).
    - papy(X,Y) :- pere(X,Z), pere(Z,Y).

    - Clause de Horn complète
- Questions : S(...), ..., T(...).
    - pere(jean,X), mere(annie,X).
    - Clause de Horn sans littéral positif

2. Graphe de résolution

programme.pl

```
pere(charlie, david).
pere(henri, charlie).
papy(X,Y) :- pere(X,Z), pere(Z,Y).
```

```
?- papy(X,Y).
```

```
X=henri, Y=david
```

```
$swiprolog
Welcome to SWI-Prolog (Version 3.3.0)
Copyright (c) 1993-1999 University of Amsterdam.
All rights reserved.
For help, use ?- help(Topic). or ?-apropos(Word).
?- [programme].
% programme compiled 0.00 sec, 824 bytes
true.
?- listing.
pere(charlie, david).
pere(henri, charlie).
papy(X, Y) :-
     pere(X, Z),
     pere(Z, Y).
true.
```

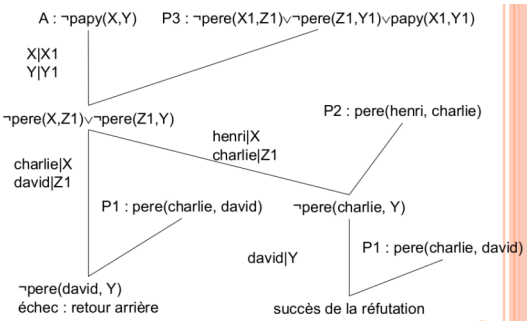
```
?- pere(charlie,david).
true.
?- pere(charlie,henri).
false.
?- pere(X,Y).
X = charlie
Y = david
true.
?- pere(X,Y).
X = charlie
Y = david ;
X = henri
Y = charlie
?- papy(x,y).
false.
?- papy(X,Y).
X = henri
Y = david
?- papy(henri,X).
X = david
true.
?- halt.
```

```
23 $
```

Remarque : Ordre des réponses

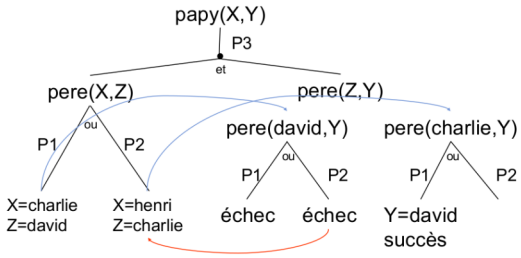
Prolog parcourt le paquet de clauses de haut en bas, chaque clause étant parcourue de gauche à droite

Graphe de résolution



Résolution par réfutation

Graphe de résolution



Graphe de résolution ET/OU

3. Les arithmétiques en Prolog

Symboles fonctionnels

Dans le fait age(homme(ahmed), 25), le symbole homme(ahmed) est un fonctionnel, ce n'est pas un prédicat.

Programme récursif

- Un programme récursif est un programme qui s'appelle lui-même
- Pour écrire un programme récursif, Il faut :

    1. Choisir sur quoi faire l'appel récursif
    2. Choisir comment passer du résultat de l'appel récursif au résultat que l'on cherche
    3. Choisir le(s) cas d'arrêt

```
1 fact(N,R) :- fact(N,1,R).
2 fact(1,R,R).
3 fact(N,I,R) :- N > 1,
4     Nm1 is N-1,
5     NewI is N*I,
```

```
    fact(Nm1,NewI,R).
```

```
?- trace, fact(3,N).
Call: (7) fact(3, _G234) ? creep
Call: (8) fact(3, 1, _G234) ? creep
Call: (9) 3>1 ? creep
Exit: (9) 3>1 ? creep
^ Call: (9) _G305 is 1*3 ? creep
^ Exit: (9) 3 is 1*3 ? creep
^ Call: (9) _G308 is 3-1 ? creep
^ Exit: (9) 2 is 3-1 ? creep
Call: (9) fact(2, 3, _G234) ? creep
Call: (10) 2>1 ? creep
Exit: (10) 2>1 ? creep
^ Call: (10) _G311 is 3*2 ? creep
^ Exit: (10) 6 is 3*2 ? creep
^ Call: (10) _G314 is 2-1 ? creep
^ Exit: (10) 1 is 2-1 ? creep
Call: (10) fact(1, 6, _G234) ? creep
Call: (11) 1>1 ? creep
Fail: (11) 1>1 ? creep
Redo: (10) fact(1, 6, _G234) ? creep
Exit: (10) fact(1, 6, 6) ? creep
N = 6
```

Bouclage : Cas 1

```
maries(jean, sophie).
maries(philippe, stephanie).
maries(A, B) :- maries(B, A).
```

```
?- maries(jean,sophie).
true.
?- maries(sophie,jean).
true.
?- maries(X,Y).
X = jean
Y = sophie ;
X = philippe
Y = stephanie ;
X = sophie
Y = jean ;
X = stephanie
Y = philippe ;
X = jean
Y = sophie ;
...
```

Bouclage : Cas 2

```
maries(jean, sophie).
maries(philippe, stephanie).
sont_maries(A, B) :- maries(A, B).
sont_maries(A, B) :- maries(B, A).
```

```
?- sont_maries(X,Y).
```

```
X = jean
Y = sophie ;
X = philippe
Y = stephanie ;
X = sophie
Y = jean ;
X = stephanie
Y = philippe ;
false.
?-
```

Les arithmétiques

- Comparaisons:
  - =:=, =\=, >, <, >=, =<
- Affectaction : is
  - ?- X is 3+2.
  - X=5
- Fonctions prédéfinies :
  - -, +, *, /, ^, mod, abs, min, max, sign, random, sqrt, sin, cos, tan, log, exp, ...

Comparaison et unification de termes

- Vérifications de type :
  - var, nonvar, integer, float, number, atom, string, ...
- Comparer deux termes :
  - T1==T2 réussit si T1 est identique à T2
  - T1\==T2 réussit si T1 n'est pas identique à T2
  - T1=T2 unifie T1 avec T2
  - T1\=T2 réussit si T1 n'est pas unifiable à T2

4. Les listes en Prolog

- Liste vide : [ ]
- Cas général : [Tete|Queue]
  - [a,b,c] ≡ [a|[b|[c|[ ]]]]

Exemple

- [X|L] = [a,b,c] ⟶ X = a, L = [b,c]
- [X|L] = [a] ⟶ X = a, L = []
- [X|L] = [] ⟶ échec
- [X,Y] = [a,b,c] ⟶ échec
- [X,Y|L] = [a,b,c] ⟶ X = a, Y = b, L = [c]
- [X|L] = [X,Y|L2] ⟶ L = [Y|L2]

Somme des éléments

```
SOMME DES ÉLÉMENTS D'UNE LISTE DE NOMBRES
/* somme(L, S) L liste de nb donnée, S nb résultat */
somme([],0).
somme([X|L],N) :- somme(L,R), N is R+X.

?- somme([1,2,3,5],N).
N = 11
```

Variable indéterminée

```
/* ieme(L,I,X) L liste donnée, I entier donné, X elt res */
ieme([X|_],1,X).
ieme([_|L],I,R) :- I>1, Im1 is I-1, ieme(L,Im1,R).

?- ieme([a,b,c,d],2,N).
N = b ;
false.
```

Test ou génération

```
/* appart(X,L) X elt donné, L liste donnée */
appart(X,[X|_]).
appart(X,[_|L]) :- appart(X,L).

?- appart(a,[b,a,c]).
true.
?- appart(d,[b,a,c]).
false.
?- appart(X,[b,a,c]).
X = b ;
X = a ;
X = c ;
false.
?- trace, appart(X,[b,a,c]).
Call: (7) appart(_G284, [b, a, c]) ? creep
Exit: (7) appart(b, [b, a, c]) ? creep
X = b ;
Redo: (7) appart(_G284, [b, a, c]) ? creep
Call: (8) appart(_G284, [a,
Exit: (8) appart(a, [a, c]) ? creep
X = a ;
Redo: (8) appart(_G284, [a, c]) ? creep
Call: (9) appart(_G284, [c]) ? creep
Exit: (9) appart(c, [c]) ? creep
X = c ;
Redo: (9) appart(_G284, [c]) ? creep
Call: (10) appart(_G284, []) ? creep
Fail: (10) appart(_G284, []) ? creep
false.
```

LE PRÉDICAT MEMBER

- Le prédicat appart est prédéfini en Prolog

- Il est très utile

```
?- member(c,[a,z,e,c,r,t]).
true
?- member(X,[a,z,e,r,t]).
X = a ; X = z ; X = e ; X = r ; X = t.
?- member([3,V],[[4,a],[2,n],[3,f],[7,g]]).
V = f .
```

Utilisation du prédicat append

```
Append est le prédicat prédéfini pour la concaténation de listes
?- append([a,b,c],[d,e],L).
L = [a, b, c, d, e]
Il est complètement symétrique et peut donc être utilisé pour
Trouver le dernier élément d'une liste :
?- append(_,[X],[a,b,c,d]).
X = d
Couper une liste en sous-listes :
?- append(L1,[a|L2],[b,c,d,a,e,t]).
L1 = [b, c, d],
L2 = [e, t]
```

5. Exercices sur Prolog

Exercice : Programmation avec Prolog *[solution* n°8 p.43*]*

Écrire un programme Prolog pour le problème de remplissage des seaux.

Écrire un programme Prolog pour le problème de jeu de taquin à 9 tuiles.

* *

*

Un aperçu sur le langage Prolog est présenté dans ce chapitre.

Conclusion

Les principes et les applications de l'intelligence artificielle sont présentés dans ce module afin de donner une idée générale à l'étudiant et de l'initier à cette science qui est devenue importante pour l'économie.

Solutions des exercices

> **Solution** n°1

Exercice p. 13

Complétez les trous du texte suivant :

- L'intelligence artificielle (IA) est `l'ensemble des théories et des techniques` mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence.
- Souvent classée dans le groupe des `sciences cognitives`, elle fait appel à la neurobiologie computationnelle (particulièrement aux `réseaux neuronaux`), à la logique mathématique (partie des mathématiques et de la philosophie) et à l'informatique.
- Si les progrès de l'intelligence artificielle sont récents, ce thème de réflexion est tout à fait ancien, et il apparaît régulièrement au cours de l'histoire. Les premiers signes d'intérêt pour une intelligence artificielle et le principal précurseur de cette discipline est `l'automate` .

> **Solution** n°2

Exercice p. 24

Choisir par glissement de la réponse correcte

Méthodes aveugles	**Méthodes heuristiques**
énumération exhaustive de tous les états de l'espace de recherche	construction de chemins minimaux dans l 'exploration de l 'espace de recherche

> **Solution** n°3

Exercice p. 24

Choisir par glissement de la bonne réponse

Complétude	**Optimalisé**	**Complexité en espace**	**Complexité en temps**	**Critère b**	**Critère d**	**Critère m**
Solution garantie si elle existe	Meilleur solution garantie	espace mémoire nécessaire pour effectuer la recherche	temps nécessaire pour effectuer la recherche	branchement maximum du graphe d'états	profondeur à la quelle se trouve le nœud solution	profondeur maximale du graphe

> **Solution** n°4

Exercice p. 33

Donner la classe de

donnée	**information**	**connaissance**
ahmed	ahmed est une personne	si ahmed est une personne alors il est humain

> **Solution** n°5

Exercice p. 33

Complétez les trous du texte suivant :

Un système à base de connaissances est composé de trois composants :

1. Une `base de de connaissances`
2. Un `moteur d'inférence`
3. Une `interface`

> **Solution** n°6

Exercice p. 33

Soit la base de connaissances suivante :

$\{pere(ahmed, mohamed), pere(mohamed, farid), pere(X, Z) \wedge pere(Z, Y) \rightarrow grandPere(X, Y)\}$

Donner le graphe ET/OU pour le but (question) $grandPere(X, Y)$

le noeud initial : grandPere(X, Y), le noeud suivant pere(X, Z) ET pere(Z, Y), le noeud suivant pere(ahmed, mohamed) OU pere(mohamed, farid) à continuer

> **Solution** n°7

Exercice p. 33

Remplir les trous suivants

L'exploitation des connaissances passe par cinq opérations : `identification`, `création`, `stockage`, `partage` et `utilisation`.

> **Solution** n°8

Exercice p. 40

Écrire un programme Prolog pour le problème de remplissage des seaux.

Écrire un programme Prolog pour le problème de jeu de taquin à 9 tuiles.

Les programmes Prolog