REPUBLIQUE ALGERIENNE DEMOCRATIQUE
ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
Université des Sciences et de la Technologie
d'Oran Mohamed BOUDIAF
Faculté des Mathématiques et de l'Informatique
Département de l'Informatique

Polycopié de cours

Fondements Logiques pour l'Informatique

Dr. YEDJOUR Hayat

Année Universitaire 2022/2023

Avant-Propos

Ce cours introduit les fondements et les bases de la logique formelle pour l'intelligence artificielle. Il s'adresse aux étudiants de master 1 informatique du département Informatique de l'USTO-MB, spécialités : SID, RSID et IAA. Le but de ce cours est d'étudier en détails les fondements de la logique qui peuvent être très utile pour automatiser les tâches de raisonnement rencontrées lors de la construction ou analyse de modèles et de programmes.

Avec l'article de Turing, l'intelligence artificielle a vu le jour comme une finalité de la formalisation de la logique humaine afin de la simuler sur les machines. La machine de Turing, les systèmes formels, les systèmes experts, et toute autre machine abstraite sont basés sur des principes liés aux logiques de différents ordres.

Dans ce cours, on présentera les notions et les outils logiques de base qui, dans l'univers concret de l'informatique, permettent de comprendre en quoi la logique est nécessaire à la pratique de l'informatique et par suite leurs domaines d'application tels que les fondements des langages de programmation et des bases de données, l'intelligence artificielle ou le génie logiciel.

Ce cours est divisé en six chapitres à savoir :

— Introduction et rappels : Ce chapitre introduit une courte histoire de l'Intelligence Artificielle (IA), les différentes façons de voir l'(IA) avec des exemples illustrant son état de l'art aujourd'hui. L'IA face au test de Turing, permet de vérifier la capacité d'une machine à faire preuve de signes d'intelligence humaine.

— Récursivité et décidabilité : Ce chapitre traite la notion de calculabilité qui cherche d'une part à identifier la classe des fonctions qui peuvent être calculées à l'aide d'un algorithme et d'autre part à appliquer ces concepts à des questions fondamentales des mathématiques. La notion de décidabilité et celle de calculabilité sont semblables. Toutes les fonctions ne sont pas calculables, le problème de l'arrêt en est un exemple.

— Les systèmes formels : Dans ce chapitre on définira un système formel comme étant une modélisation mathématique d'un langage en général spécialisé, pour fournir une description générique à partir d'un noyau minimal et de règles d'extension. En effet, les systèmes formels sont apparus en logique mathématique afin de représenter mathématiquement le langage et le raisonnement mathématique.
— Le calcul propositionnel : Est un premier modèle logique limité à des raisonnements sans variables. Les principes de la démonstration automatique sont fournis dans ce chapitre.
— Le calcul des prédicats : Etend le chapitre précédent à des individus et à des propriétés sur ces individus.
— Applications et résolution : Dans ce chapitre, nous nous intéressons à la preuve par résolution. Nous montrerons aussi que la méthode de résolution étendue au calcul des prédicats est plus complexe que sa version en calcul propositionnel puisqu'elle doit prendre en compte l'existence des variables. Elle sert de base au langage Prolog.

Les chapitres sont composés de définitions, de propriétés et d'illustrations par des exemples. Leur connaissance est indispensable pour la réalisation des exercices. Les chapitres sont systématiquement accompagnés d'exercices. Quelques démonstrations sont proposées comme exercices qui, ensuite corrigées. Les corrigées sont assez détaillés pour que l'étudiant puisse acquérir les notions essentielles par la pratique.

Table des matières

Chapitre 1

Introductions et rappels

Sommaire

1.1 L'intelligence artificielle et ses applications

L'intelligence artificielle fait référence aux technologies logicielles qui permettent à un robot ou à un ordinateur d'agir et de penser comme un humain. Certains ingénieurs en logiciel disent que l'intelligence artificielle n'est valable que si elle fonctionne aussi bien ou mieux qu'un être humain. Cependant depuis une soixantaine d'années des hommes de science s'évertuent à rendre ce défi possible ! Créer une Intelligence Artificielle ! L'intelligence artificielle est donc un concept assez difficile à définir, elle englobe plusieurs disciplines. On peut la définir comme étant une théorie de développement de systèmes informatiques capables d'effectuer des tâches nécessitant normalement une intelligence humaine. Ce qui est certain, c'est que quand on parle des progrès récents en intelligence artificielle, on parle principalement des avancées extraordinaires réalisées dans un des domaines de l'intelligence artificielle, à savoir La reconnaissance vocale, la prise de décision, la perception visuelle, par exemple, sont des caractéristiques de l'intelligence humaine que peut posséder l'intelligence artificielle. La traduction entre les langues est une autre caractéristique.

1.2 Définitions

L'Intelligence Artificielle (IA) regroupe les sciences et technologies qui permettent d'imiter, d'étendre et/ou d'augmenter l'intelligence humaine avec des machines. L'IA fait partie de ce que l'on appelle aussi les sciences cognitives. Depuis toujours, l'homme cherche à concevoir des machines pouvant exécuter des tâches manuelles puis intellectuelles semblables à celles de l'être humain. On peut dire que les automates, sont une toute première forme d'intelligence artificielle. Grâce aux progrès techniques et aux nombreuses expositions universelles, les automates deviennent accessibles au public. Partant d'une simple calculatrice et en allant jusqu'à la robotique et l'informatique. Ces inventions témoignent de la volonté de l'homme de créer des objets capables de reproduire l'action de l'Homme. Aujourd'hui, l'intelligence artificielle fait donc partie intégrante de notre culture comme en témoigne l'existence de nombreux articles, livres, ou films sur ce sujet ! L'intelligence artificielle (IA) fait référence à la simulation de l'intelligence humaine dans des machines programmées pour penser comme les humains et imiter leurs actions. Le (IA) terme peut également s'appliquer à toute machine présentant des caractéristiques associées à l'esprit humain, telles que l'apprentissage et la résolution de problèmes. La caractéristique idéale de l'intelligence artificielle est sa capacité à rationaliser et à prendre des mesures offrant les meilleures chances d'atteindre un objectif spécifique.

Il n'est pas aisé de donner une définition de l'intelligence artificielle. Intéressons nous d'abord à la l'intelligence : l'intelligence est la faculté de comprendre, de connaître, de penser, de raisonner, de s'adapter et d'être conscient de ses facultés. Définissons ensuite le mot artificiel : ce terme fait référence à ce qui a été produit par la technique, par l'activité humaine et non par la nature. . On peut donc définir l'intelligence artificielle comme une discipline scientifique relative au traitement des connaissances et au raisonnement, dans le but de permettre à une machine d'exécuter des fonctions normalement associées à l'intelligence humaine ; compréhension, raisonnement, dialogue, adaptation, apprentissage...

Définition (*Marvin Lee Minsky 1956*) L'intelligence artificielle est la construction de programmes informatiques qui s'adonnerait 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.

Définition(*John McCarthy 1955*) Le but de intelligence artificielle est l'étude de la structure de l'information et de la structure des processus de résolution de problèmes, indépendamment des applications et indépendamment d'une réalisation.

L'intelligence artificielle n'a pas d'objet de recherche académique bien défini à part l'intérêt porté au "mental" et aux représentations de connaissances. Elle s'est divisée en de nombreuses sous-disciplines focalisant sur des problèmes bien distincts (tel que la vision, la résolution de problèmes, la compréhension du langage, l'apprentissage,...). Il n'existe pas de paradigme unifié de recherche et certaines branches de l'IA sont devenues des terrains d'échanges multidisciplinaires où se côtoient philosophes, psychologues, informaticiens et autres qui s'intéressent aux divers problématiques de l'intelligence. Ainsi les branches fondamentales de l'IA font partie des sciences cognitives. Forcément, il existe une multitude de façons de concevoir recherche fondamentale et recherche appliquée.

Le but de l'intelligence artificielle étant la modélisation de l'intelligence humaine, afin de concevoir des systèmes capables de reproduire le comportement de l'humain dans l'activité du raisonnement, tout comme d'autres domaines tels que la physique, la chimie ou la biologie qui ont pour but la modélisation d'autres phénomènes.

1.3 Histoire de l'intelligence artificielle

L'histoire de l'intelligence artificielle (IA) a commencé dans l'antiquité, avec des mythes, des histoires et des rumeurs d'êtres artificiels dotés d'intelligence ou de conscience par des maîtres artisans. Les graines de l'IA moderne ont été plantées par des philosophes classiques qui ont tenté de décrire le processus de la pensée humaine comme une manipulation mécanique des symboles. Ce travail a abouti à l'invention du calculateur numérique programmable dans les années 1940, une machine basée sur l'essence abstraite du raisonnement mathématique. Cet appareil et les idées qu'il a inspirées ont incité une poignée de scientifiques à commencer à discuter sérieusement de la possibilité de créer un cerveau électronique. Le domaine de la recherche sur l'IA a été créé lors d'un atelier qui s'est tenu sur le campus du Dartmouth College au cours de l'été 1956. Ceux qui y ont participé deviendraient les chefs de file de la recherche sur l'IA pendant des décennies. Nombre d'entre eux ont prédit qu'une machine aussi intelligente que l'être humain n'existerait pas avant une génération et on leur a donné des millions de dollars pour concrétiser cette vision. Finalement, il devint évident qu'ils avaient largement sous-estimé la difficulté du projet. En 1973, en réponse aux critiques de James Lighthill (pionnier dans le domaine) et aux pressions incessantes du Congrès, les gouvernements américain et britannique arrêtèrent de financer des recherches non dirigées sur l'intelligence artificielle. Les années difficiles qui suivirent furent connues plus tard sous le nom d'« hiver de l'IA ». Sept ans plus tard, une initiative visionnaire du gouvernement japonais a incité les gouvernements et l'industrie à fournir à l'IA des milliards de dollars, mais à la fin des années 80, les investisseurs ont été déçus par l'absence de la puissance informatique nécessaire (matériel informatique) et ont à nouveau retiré des fonds.

Quelques dates :

• Jusqu'à la première moitié du XXième siècle : il s'agit avant tout de construire les bases philosophiques et mathématiques sur lesquelles pourra se développer l'IA. A noter, quelques projets de « machines à penser » , qui sont essentiellement des « machines à calculer » tels que le projet de machine à calculer de Pascal au XVIIème siècle, le projet « d'automate rationnel » de Leibnitz au début du XVIII ème et la machine de Babbage au XXième.

•Années 1940 / 1950 : Naissance de l'IA.

— Modèles mathématiques de Pitts et MacColloch en 1943 ; machine de Türing en 1950 ;

théorie de l'information de Shannon ;

— problème de représentation et d'extraction de connaissances et problème de rédaction générique ;

— Intelligence artificielle dans la litérature et le cinéma de la science fiction (Film : Odysée 2001 de Kubrick, IA de Spielberg)

— Apparition du mor robot pour la première fois en 1923 dans la pièce de théâtre « R.U.R » (Rossum's Universal Robots) écrite par Karel Capek.

•Années 1950 - 1960 : Essor de l'IA : conférence de Darmouth en 1956 consacrant le terme d'Intelligence Artificielle ; développement des ordinateurs, des langages informatiques de type Lisp, ...

— En 1950, Isaac Asimov (auteur de science fiction avec un background scientifique) propose ses trois lois de la robotique (Film I.Robot avec Will Smith 2004) : - Un robot ne doit pas attenter à la vie d'un humain. - Un robot doit obéir aux ordres d'un humain sauf si cela contredit la première loi. - Un robot doit préserver sa propre existence sauf si cela contredit les deux premières lois.

— Le terme intelligence artificielle est apparu en 1956 à la rencontre de Minsky, McCarthy, Newell et Simon au collège de Darmouth (New Hampshire, USA). C'était l'époque de l'enthousiasme absolu (Simon en 1958 : Simon reçoit le prix nobel en économie en 1978), et développement de travaux de jeux d'échec, et démonstration de théorèmes en géométrie : - Un programme d'échec arrivera au niveau d'un champion du monde. - Un programme de démonstration automatique de théorème découvrira un théorème mathématique. - Apparition du premier programme le LOGIC THEORIST (démonstration automatique de théorème) en 1956 et du langage IPL1.

— Apparition du langage LISP en 1960 par MarCarthy, et PROLOG en 1971 par Alan Colmerauer

— ELIZA est construit au MIT en 1965, un système intelligent qui dialogue en anglais et qui joue au psychothérapiste.

•Années 70 : Apparition de nouveaux modèles de représentation et de traitement des connaissances. Systèmes experts, création du langage Prolog orienté vers le traitement de la langue naturelle en France au début des années 70, Les systèmes experts parmi lesquels :

— 1969 : DENDRAL : analyse des résultats d'une spectrographie de masse ;

— 1967 : MACSYMA (logiciel de calcul formel) ;

— 1977 : MYCIN (maladies infectieurses) ;

— PROSPECTOR : en géologie et - HEARSAY-II en compréhension de la parole.

•Années 80 : Phase d'industrialisation et diffusion d'applications industrielles, commercialisation de progiciels.

— L'IA est le carrefour de plusieurs disciplines : informatique, logique, linguistique, neurologie et psychologie ;

— Naissance du langage Smalltalk en 1980 ;

— Des techniques spécifiques à l'informatique ont été développées à partir des années 1980, les plus connues sont les réseaux de neurones qui simulent l'architecture du cerveau humain, les algorithmes génétiques qui simulent le processus de sélection naturel des individus, la programmation logique inductive et le processu de déduction, les réseaux bayésiens qui se fondent sur la théorie de probabilités pour choisir parmi plusieurs hypothèses la plus satisfaisante ;

— Apparition du champ l'IA distribué ou « Systèmes multiagents » qui simule l'intelligence collective trouvée dans la nature, comme les êtres multicellulaires simples, les colonies d'insectes sociaux, les sociétés humaines. Ces sources d'inspirations montrent que l'activité corrélée d'entités plus simples peut donner une forme d'intelligence supérieure.

• 1990 - 2000 : retour à des projets plus réalistes.

— Nouveaux concepts (ex. Agents intelligents), développement des applications liées à la recherche de l'information (parallèlement au développement d'Internet), résurgence des concepts de réseaux neuronaux et d'algorithmes génétiques ;

— L'avènement d'Internet a ouvert la voie au partage et à la communication de connaissances ;

— Organisation et traitement de grandes masses d'information ;

— Extraction de connaissances pertinentes (DATAMINING) ;

— Intégration de moteurs de recherche comme Google ;

— En 1995 : Le système automatique de vision ALVINN de Carnegie Mellon University a permis la conduite automatique d'un véhicule appelé Navlab5 de Pittsburgh à San Diego ;

— En 1997 : à Philadelphie, le champion du monde aux échecs, GarryKasparov a été battu par Deep Blue (un ordinateur IBM) ;

— En 1997 : RoboCup, a eu le championnat des robot qui jouent au football, lors d'une

conférence en IJCAI-97 à Nag.

1.4 Différents types d'Intelligence Artificielle

Les experts en informatique distinguent plusieurs types d'intelligence artificielle :

— Systèmes qui pensent comme des humains : automatisent des activités telles que la prise de décision, la résolution de problèmes et l'apprentissage. Un exemple est les réseaux de neurones artificiels.
— Systèmes agissant comme des humains : ce sont des ordinateurs qui effectuent des tâches similaires à celles des utilisateurs. C'est le cas des robots.
— Les systèmes qui pensent de manière rationnelle : ils essaient d'imiter la pensée logique rationnelle de l'homme, c'est-à-dire étudient comment faire en sorte que les machines puissent percevoir, raisonner et agir en conséquence. Les systèmes experts sont inclus dans ce groupe.
— Les systèmes qui agissent de manière rationnelle : idéalement, ce sont ceux qui tentent de rationaliser le comportement humain, tels que les agents intelligents.

Les progrès de l'IA stimulent déjà l'utilisation du big data en raison de sa capacité à traiter d'énormes quantités de données et à fournir des avantages en matière de communication, commerciaux et commerciaux, qui lui ont permis de se positionner comme la technologie essentielle des prochaines décennies. Transports, éducation, santé, culture... aucun secteur ne résistera à ses charmes.

1.5 De l'I.A Faible à l'I.A Forte

L'IA s'inscrit dans une longue tradition humaine d'innovations s'appuyant d'abord sur la force mécanique puis sur la force intellectuelle toutes deux artificielles. Certains scientifiques visent à atteindre dans un premier temps l'intelligence humaine.

Avec le développement de l'ordinateur et des réseaux informatiques, on constate que le rêve de construire des machines «intelligentes» se concrétise lentement. L'ordinateur est largement supérieur à l'être humain pour calculer, mémoriser, classer et recouvrer des informations. Ce-

pendant, pour des opérations cérébrales comme raisonner, analyser son environnement (voir) et communiquer, l'être humain le déclasse complètement. L'intelligence artificielle est un domaine très vaste, dont les utilisations sont aussi nombreuses que différentes. Il faut distinguer deux principales catégories d'intelligence artificielle :

1.5.1 Intelligence artificielle forte

L'intelligence artificielle forte, ou ascendante, est l'approche la plus similaire du comportement humain, puisqu'elle fait commencer le programme avec des choses simples pour arriver au final à des choses compliquées. nous fait tout de suite penser à l'apprentissage logique d'un enfant, qui s'adapte et apprend seul face à des situations, tout comme le programme qui assemble plusieurs éléments simples entre eux pour avoir une idée complexe. L'automate serait alors doté d'une réelle conscience de lui même et pourrait éprouverait des sentiments ; il serait capable de penser et son raisonnement serait donc le même que celui d'un être humain. Le grand avantage de l'intelligence artificielle ascendante, c'est qu'elle peut apprendre ou s'adapter très facilement et enfin d'agir pour répondre au mieux aux besoins pour lesquels elle a été conçue.

1.5.2 Intelligence artificielle faible

L'IA faible, ou descendante, vise principalement à reproduire le plus fidèlement possible, à l'aide d'un programme informatique, le résultat d'un comportement spécifique observé à l'avance et ne donnant pas d'application non prévue. Les programmes de type descendants ne visent pas à évoluer, ils effectuent leurs tâches et uniquement ce pour quoi ils ont été programmés. On peut citer les ordinateurs ou les calculatrices comme exemples de figures d'intelligence artificielle faible.

1.6 Domaines d'applications de l'I.A

La diversité des domaines d'application de l'IA est un principe qui évolue sans cesse puisque c'est un sujet de recherche scientifique et de fiction exploité dans le monde entier et tous les secteurs seront concernés par l'émergence de l'Intelligence Artificielle dans les décennies qui viennent. Les nombreux domaines d'applications en sont la preuve incluants la robotique, les programmes informatiques, le diagnostic médical, la vision par ordinateur, la classification naturelle, la planification des tâches pour des prédictions financières, l'architecture (CAO),l'éducation, les centrales nucléaires, et les jeux vidéos.

1.6.1 Application de l'I.A dans le traitement d'images et de vidéos

Les programmes de traitement de l'image et de la vidéo sont les technologies de l'Intelligence Artificielle les plus connues et seront peut-être les plus utilisées à très court terme. Partant du principe que l'IA tente d'imiter l'homme, il est normal que les chercheurs se soient rapidement concentrés sur la notion de perception (du monde qui nous entoure). La vue et la reconnaissance de ce qui est vu par la machine (reconnaissance d'images) sont donc des fondamentaux de l'IA. De plus ou moins jeunes sociétés en ont d'ailleurs fait leur cœur de métier ou leur principal axe stratégique de développement : Cortica, Deepomatic, Facebook, etc. Le système de reconnaissance d'images fonctionne généralement avec le machine learning, une technique d'apprentissage automatique. Toutefois d'autres techniques basées sur de la statistique bayésienne existent comme celle de la société Gamalon.

1.6.2 Applications liées au langage

Les programmes de reconnaissance vocale ou de parole, les agents conversationnels (ou chat bots) et les autres applications liées au langage naturel sont des technologies du domaine appelé « traitement du langage naturel langage », une branche de l'Intelligence Artificielle. Ces outils utilisent généralement des méthodes innovantes comme le machine learning et les réseaux de neurones convolutifs pour accélérer le traitement du langage dans différents domaines (défense militaire, renseignements généraux, justice, communication entre particuliers à travers le téléphone portable, ... bref, tout ce qui peut nécessiter l'usage de la parole et des mots !). De nombreuses applications permettent cependant aujourd'hui, grâce à l'IA, d'effectuer aisément et rapidement des taches jadis fastidieuses, telles que résumer des textes, structurer ou ordonner un document volumineux, traduire un texte une conversation en plusieurs langues et en temps réel, rédiger des contenus ou répondre aux questions « classiques » ou récurrentes de sa clientèle.

1.6.3 Application dans la vie quotidienne

Reconnaissance vocale sur Apple : SIRI

SIRI est un assistant personnel intelligent lancé en 2011 et compatible avec tous les téléphones d'Apple à partir de l'iPhone 4S. C'est un système de reconnaissance vocale couplé à une intelligence artificielle, qui permet à son utilisateur de parler à son mobile afin d'exécuter des tâches (Envoyer un message, S'informer de la météo, etc...) ou d'accéder à des informations (Restaurant le plus proche, séance de cinéma, etc...) plus efficacement et rapidement, seulement

en utilisant la voix. Siri se classe dans la catégorie des IA forte, car ce logiciel a la capacité d'évoluer au cours du temps. Il est capable de s'habituer à votre articulation, votre accent et même à un cheveu sur la langue ! Néanmoins, Il n'est pas disponible dans toutes les langues ni dans tous les pays.

Les Jeux Vidéos

Depuis peu, les développeurs de jeux vidéo ont réussi à créer une IA dîtes plus « flexible » capable d'apprentissage et d'interactions au fil du temps, afin d'améliorer l'expérience de jeu. Fréquemment utilisé dans les jeux de sport, cette IA a été retravaillé et est maintenant plus abouti , le résultat est constaté dans le jeu FIFA 12 où l'IA prend des décisions diverses en fonction de la situation et s'adapte plus aisément à votre style de jeu. (Offensives, stratégies, etc...). L'intelligence artificielle est également très présente dans les jeux de combat. Les machines sont aujourd'hui capables d'affronter et de battre les humains sur le terrain des jeux. Dotés d'algorithmes performants, elles se comportent comme de véritables compétitrices à l'approche à la fois très stratégique et parfois surprenante d'après les experts. Récemment, des champions du monde du jeu de Go se sont inclinés devant AlfaGo, un logiciel d'Intelligence Artificielle conçu par Google.

L'IA Depuis quelques années l'intelligence artificielle est de plus en plus présente dans le domaine militaire, notamment dans les robots. Ce robot est une drone d'espionnage et de destruction de cibles militaires. Il est principalement utilisé par l'armée américaine en Afghanistan. Cet avion sans pilote ramène, de ses vols au-dessus des territoires ennemis, des informations précieuses comme par exemple des photos montrant la configuration de terrain et de la position des troupes adverses. De plus, il est capable de diffuser un rayonnement électromagnétique qui brouille les radars, les instruments de communication... Enfin, le Predator peut aussi tirer des missiles sur les cibles désignées par ses pilotes au sol.

L'Automatisation intelligente

Le premier niveau d'usage du cerveau par l'homme est lié à son aptitude à la répétition. Or, ce qui est répétitif a été rapidement considéré comme automatisable industriellement. Demain l'Intelligence Artificielle rendra l'automatisation encore plus intelligente. Cette tendance va encore plus propulser l'économie mondiale dans l'ère de l'automate, bien que l'on craigne de nombreuses pertes d'emplois dans tous les secteurs d'activité. Aujourd'hui L'automatisation

intelligente des tâches gagne le transport, les médias, la finance, la santé, la restauration et bien d'autres activités. Les ordinateurs sont désormais capables de planifier et d'ordonnancer des travaux vraiment complexes, qui ne pouvaient être réalisés jusqu'à présent que par l'homme.

BiGDoG Evolution

Le Big Dog est un robot spécialisé dans le transport de matériel. Le créateur et fabricant, Boston Dynamics (Cambridge, USA). Ce projet est financé par la Drapa, l'agence de recherche de la défense des Etats-Unis. Pour l'instant, personne n'utilise ce robot car il est encore en développement. Il équipera prochainement les militaires américains. Ce robot de 75 kg et dont le nom signifie « grand chien » en anglais, est garanti tout-terrain. Il avance aussi bien sur sol humide que sec, sur un terrain boueux ou enneigé, même sur des rochers. En cas de choc, il conserve son équilibre. Il peut soulever 150 kg et est utilisé pour transporter le paquetage des troupes (nourriture, médicaments, etc...).

BiGDoG

L'apparition de réalisations cinématographiques mettant en scène des robots intelligents (Ex : Star Wars 1977).

La santé, la Bio-Informatique

Les géants de l'informatique et du Web (IBM, Google et autres) ont investi le juteux domaine de la santé pour proposer des solutions d'Intelligence Artificielle qui permettent de répondre à certains besoins médicaux. Diagnostiquer (avec un taux de réussite supérieur à l'homme) et traiter efficacement des pathologies graves comme le cancer devient maintenant plus aisé. Des entreprises proposent également des applications mobiles pour le diagnostic et pour l'auto-traitement de certaines pathologies par les patients eux-mêmes.

1.7 Les grands acteurs de l'intelligence artificielle

Google, Apple, Facebook et Amazon ne pouvaient pas ne pas s'investir fortement dans l'intelligence artificielle alors qu'ils possèdent des données en énorme quantité. En voici quelques exemples.

a) **Google :**

Google, très impliqué dans l'intelligence artificielle, procède habituellement par rachat. En 2014, Google a acheté la société anglaise DeepMind, qui avait développé des réseaux de neurones pour

jouer aux jeux vidéo. Mais l'objectif avoué de DeepMind est actuellement de « comprendre ce qu'est l'intelligence ». DeepMind est célèbre pour son programme AlphaGo, qui a battu le champion du monde de go. En octobre 2017, le programme a franchi une étape supplémentaire : en jouant contre lui-même, non seulement son apprentissage a été plus court, mais surtout il est devenu plus fort que la version précédente 22. Nous avons ici un premier exemple d'apprentissage non supervisé, facilité par le fait que le contexte, à savoir les règles du jeu de go, est parfaitement mathématisable. Google a aussi son propre moteur de recommandation nommé Google Home, un haut-parleur et un assistant vocal disponible en trois versions différentes.

b) **Amazon :**

Amazon utilise de l'intelligence artificielle dans son moteur de recommandation, nommé Echo, et dans ses assistants basés sur son système de reconnaissance (Voir « AlphaGoZero : Learning from scratch »,)

deepmind.com/blog/alphago-zero-learning-scratch

Via son offre de services dans le cloud, Amazon propose également des services fondés sur l'intelligence artificielle, comme la reconnaissance de la parole ou des robots de discussion, les fameux chatbot.

c) **Facebook :**

Facebook est un énorme utilisateur d'intelligence artificielle. Il choisit les messages qu'il affiche en utilisant un moteur de type moteur de recommandation. Récemment, Facebook a mis en place un moteur d'intelligence artificielle pour détecter les tendances suicidaires. Comme le dit Joaquin Candela, directeur du département d'intelligence artificielle appliquée, « Facebook n'existerait pas sans intelligence artificielle».

d) **Apple :**

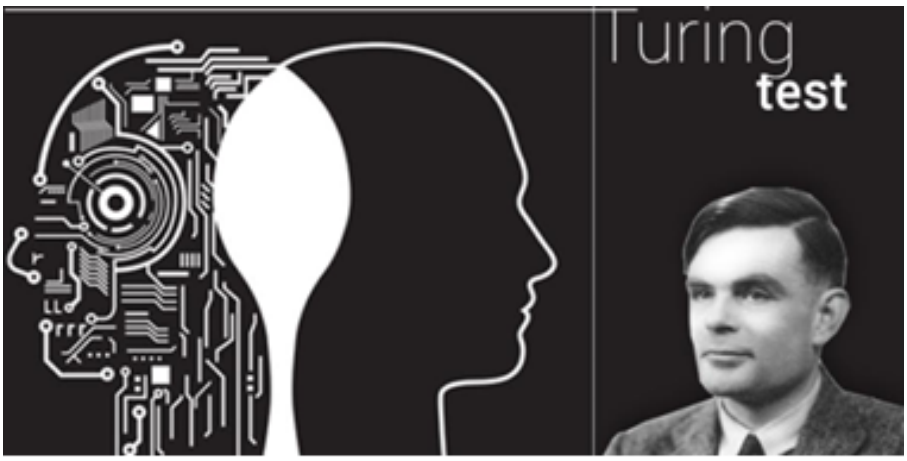
Apple investit énormément dans l'intelligence artificielle et propose même un blog où sont expliquées ses recherches. Siri en est l'expression la plus évidente : ce système de reconnaissance de la parole très performant utilise des réseaux de neurones de dernière génération qui permettent de comprendre des commandes même dans des environnements très bruyants, comme dans la rue. Apple va également bientôt lancer son propre moteur de recommandation, nommé HomePod, un haut-parleur connecté avec Apple Music.

e) **IBM :**

IBM a créé Watson, un système d'intelligence artificielle pour jouer, à ses débuts, au jeu Jeopardy, l'équivalent américain de l'émission française Questions pour un champion. Watson a analysé 200 millions de pages pour finalement battre les anciens champions et ainsi gagner le premier prix. Maintenant, Watson est proposé par IBM dans d'autres domaines, comme la médecine ou le droit. IBM présente aussi un blog sur le sujet de l'intelligence artificielle. La question majeure est le degré d'ouverture de ces systèmes. IBM propose des interfaces de programmation à Watson qui sont ouvertes, tout comme Amazon propose des interfaces sur son système d'apprentissage profond. Donc tout le monde peut, à peu de frais, se connecter à ces systèmes et en bénéficier. Facebook, en revanche, ne partage pas la plupart de ses recherches, ne propose pas encore d'interfaces de programmation permettant de les utiliser, mais a récemment mis en accès libre son environnement d'apprentissage, Caffe2.

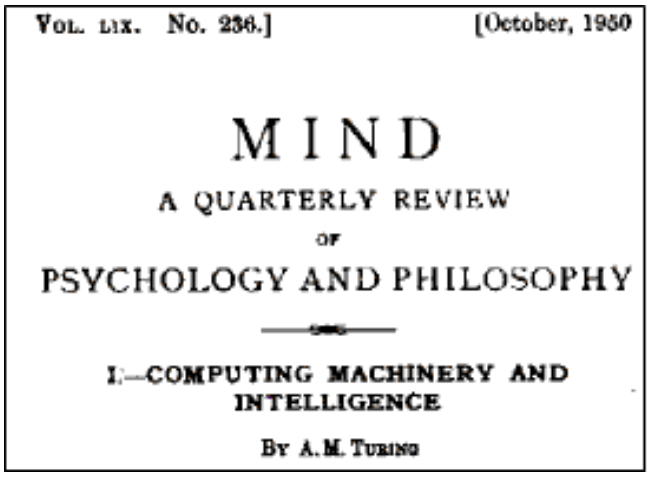
1.8 Le Test de Turing et L'Intelligence Artificielle

Comment reconnaître une machine intelligente d'une machine moins intelligente ?
Le premier qui s'est posé la question ou qui a sérieusement essayé de répondre est le mathématicien britannique nommé « Alain Turing ». Le test de Turing est un test permettant de vérifier la capacité d'une machine à faire preuve de signes d'intelligence humaine.



En 1950, le mathématicien britannique Alan Turing publie, dans le journal philosophique Mind, un article intitulé Computing Machinery and Intelligence. Considéré par beaucoup comme l'un des fondateurs de l'informatique, Turing est également un des pionniers de l'Intelligence Artificielle. Dans cet article, il explore le problème de l'intelligence artificielle peu défini jusqu'à lors. Il propose également une expérience connue sous le nom de test de Turing dans une

tentative de qualifier une machine de « consciente ». La tâche est loin d'être évidente !



Vol. lix. No. 236.] [October, 1950

MIND

A QUARTERLY REVIEW

OF

PSYCHOLOGY AND PHILOSOPHY

I.—COMPUTING MACHINERY AND INTELLIGENCE

By A. M. Turing

FIGURE 1.1 – Extrait du journal Mind

L'article de Turing commence par cette phrase : « Je propose de réfléchir à la question : Les machines peuvent-elles penser ? »

Et comme la tâche est impossible (pas claire), Turing propose de remplacer la question par un procédé expérimental moins ambigu. Turing écrit un jeu qu'il appelle « jeu de limitation », ce jeu simple comporte 3 joueurs. 2 joueurs dans une pièce, et un interrogateur dans une autre pièce.

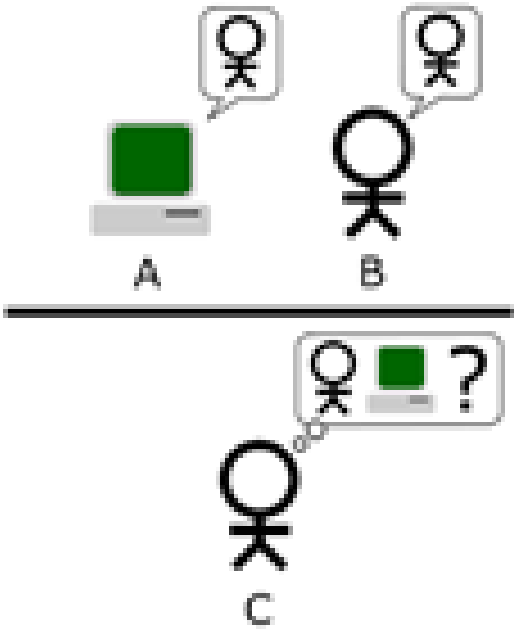


FIGURE 1.2 – Test Turing

— Le joueur A est un homme.
— Le joueur B est une femme.
— Le joueur C communique avec A et B par écrit et A et B lui répondent par écrit aussi.

En gros, des échanges e-mails 30 ans qu'on les invente. Le but de ce jeu pour l'interrogateur c'est de réussir de déterminer dans un temps (5 mns que Turing à décrit dans son article), qui est l'homme et qui est la femme ?

Turing propose alors un autre jeu en remplaçant l'homme par une machine. Est-ce que le juge (interrogateur que ça soit homme ou femme) parviendra-t-il plus facilement de distinguer l'ordinateur de l'humain (que l'homme ou la femme).

Autrement dit, telle est la question : Si le comportement de la machine est indispensable de celui d'un être humain alors pourra- on conclure que la machine est intelligente ?

L'article se poursuit par un dialogue imaginaire entre l'interrogateur et la machine essayant de passer pour un humain sous/forme (questions-réponse), et pour des questions mathématiques de somme ou produit de grand chiffres, l'ordinateur fait volontairement des erreurs pour tromper l'interrogateur.

C'était la période où sont apparus les premiers ordinateurs, qu'on appelait les « cerveaux électroniques ». les ordinateurs étaient à la mode, et ils étaient le futur (remarque, il n'y avait pas d'informaticien à l'époque, il existait juste des mathématiciens et des électroniciens). Et tout le monde voulaient savoir si ces ordinateurs étaient capable de penser et atteignent le niveau du cerveau. Donc ils ont créé des logiciels qu'on parlera par la suite, et si l'interrogateur échoue au test cela implique que la machine a gagné le test. Mais passer ou gagner le test n'est pas la même chose que penser (disait Turing).

Les chercheurs après Turing ont essayé de construire réellement des machines intelligentes et vont se répartir en 2 groupes :
— IA forte -> humain et machine identiques (science fictions existait depuis longtemps).
— IA faible->point de vue limité (ie analogie entre procédure mécanique humain et machine).

Dans une émission de radio en 1952, Turing a dit qu'il faut au moins un siècle pour que la machine réussit au test (ie 2052), car il était conscient de la difficulté du problème. Dans son article, on voyait 2 niveaux de l'IA, un niveau où il défend sa théorie en IA et 2éme niveau où il réclame que la machine et le cerveau humain ne peuvent pas être la même chose pour des raisons théoriques profondes, du fait qu'une machine est déterministe, mais le cerveau ne l'es pas. (une simulation informatique arrive toujours au même résultat, alors qu'une expérience dans le monde réel ne peut jamais être reproduite parfaitement à l'identique).

Il n'existe aujourd'hui aucun ordinateur capable de passer le test de Turing et il n'y aura pas dans le futur proche car la tâche est si difficile.

En juin 2014 les journaux du monde entier déclarait qu'une créature baptisée par « Eugene Goostman » pour passer le test de Turing à l'université Readind en Angleterre. Et 33‰ de juges ont cru discuter avec un adolescent Ukranier de 13 ans alors qu'Eugéne est un programme développé en Russie.

Mais Eugéne n'est que le dernier Avantar d'une longue liste de programmes qui commencent en 1966 avec « ELIZA » (un logiciel basique simple) d'une centaine de ligne qui réussissait pourtant à faire illusion en quelque minute et se fait passer pour un psychothérapeute.

En Septemble 2011, à Guawahati en Inde, le programme Cleverbot parvient à convaincre la majorité des participants qu'il est humain et il gagne le test à 59‰.

1.9 Machine de Turing

1.9.1 Introduction

Alain Turing est un mathématicien britannique né en 1912, après ses études à l'université à Cambridge, il a publié en 1936 un article intitulé :

Sur les nombres calculables
avec application au problèmes de décision
- Alain Turing - 1936

Cet article se veut une réponse à un problème posé quelques années auparavant par « David Hilbert » : « Existe-t-il une méthode effectivement calculable pour décider si une proposition mathématique est valide ou non ? »

Pour répondre à ces questions, Alain Turing va imaginer une machine très simple, mais qui est totalement programmable et qui permet de tester tout sorte d'algorithme.

En informatique théorique, une machine de Turing est un modèle abstrait du fonctionnement des appareils mécaniques de calcul, tel un ordinateur. Ce modèle a été imaginé par Alan Turing en 1936, en vue de donner une définition précise au concept d'algorithme ou de « procédure mécanique ». Il est toujours largement utilisé en informatique théorique, en particulier dans les domaines de la complexité algorithmique et de la calculabilité.

Et ce qu'on appelle la machine universelle de Turing, c'est ce modèle de machine de Turing auquel on a rajouté un système permettant d'enregistrer les programmes de la même façon que les données.

Aujourd'hui, on considère que tous les ordinateurs actuels grands, petits, visibles ou invisibles ont tous le même ancêtre : Machine de Turing.

1.9.2 Définition

Une machine de Turing est un concept abstrait, c'est-à-dire un objet mathématique. Par sa forme canonique simplifiée, Une machine de Turing comporte les éléments suivants :

— 1. Un ruban infini divisé en cases consécutives. Chaque case contient un symbole parmi un alphabet fini (comme des lettres, des chiffres , des ponctuations.... Mais une chose importante, c'est que cette table contient un symbole spécial appelé « symbole blanc » (marqué '0' ou un 'B' ou rien tout dépend des exemples que l'on rencontre dans la littérature : pour dire qu'un moment donnée sur le ruban il n'y a rien).
— 2. Une tête de lecture/écriture qui peut lire et écrire les symboles sur le ruban, et se déplacer sur le ruban à gauche ou à droite.
— 3. Un registre d'état qui mémorise l'état courant de la machine de Turing. Le nombre d'états possibles est toujours fini, et il existe un état spécial appelé « état de départ » qui est l'état initial de la machine avant son exécution.
— 4. Une table d'actions (ou table de transitions) qui va permettre de dire à cette machine si elle doit lire/écrire/se déplacer/changer d'état. Si aucune action n'existe pour une combinaison donnée d'un symbole lu et d'un état courant, la machine s'arrête.

1.9.3 Principe

Dans la table d'actions ou de transitions, la procédure est formulée en termes d'étapes élémentaires du type. Chacune de ces transitions est en fait une association de la forme :

(e,a) $\longrightarrow$ (e',a') : La signification est la suivante : « si la tête de lecture à un instant donné est dans l'état e et que le symbole contenu sur la case que vous regardez est a, alors remplacer ce symbole par un a', passer dans l'état e', et la tête de lecture va se déplacer sur une autre case. Cette table de transition représente en fait le programme de la machine de Turing.

1.9.4 Exemple 1 : Incrémentation d'un nombre de 1 : n⟶ n+1

J'ai n fois le nombre 1 et je veux ajouter un 1. Une manière de représenter le nombre n dans le contexte de cet exemple : 11 ⟶ 111

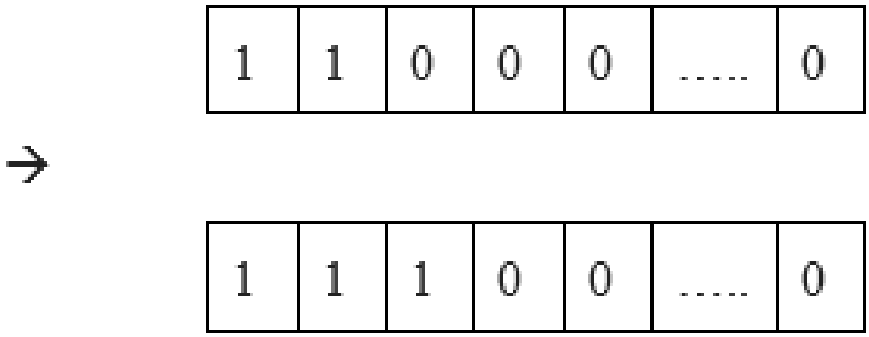


FIGURE 1.3 – Incrémentation de 1

La table de transition ici sera composée exactement de deux transitions ou deux règles suivantes :

- La première association est : (0,1)-> (0,1).
- Initialement la tête de lecture se pointera sur la valeur la plus à gauche et sera dans l'état 0.
- La seconde règle est : (0,0)-> (1,1).

R1 : L'idée dans ce cas est : si la tête de lecture à un instant donné est à l'état 0 et devant elle il y'a l'élément 1, alors elle ne change pas son état

R2 : si la tête de lecture arrive en face d'une case qui contient le 0 et si elle-même elle est dans l'état 0, et bien alors, elle va transformer son état à 1 et elle transforme le contenu de la case qu'elle vient de lire à 1.

En résumé :
On avait initialement :

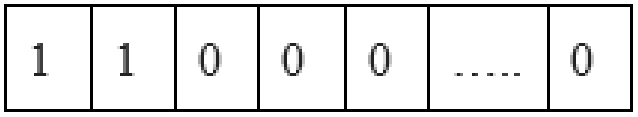


FIGURE 1.4 – Représentation du nombre à incrémenter

On a 2 règles :

R1 : (0,1) $\longrightarrow$ (0,1)

R2 : (0,0) $\longrightarrow$ (1,1)

On passe de la case 1 à la case 2 en exécutant R1. Ensuite de la case 2 à la case 3 en exécutant R2. Et quand on arrivera à la 3éme case qui contient le 0, on va transformer ce 0 en 1 et la machine va passer à l'état1 et elle va s'arrêter parcequ'il n'y a aucune règle qui prend comme état initial 1.

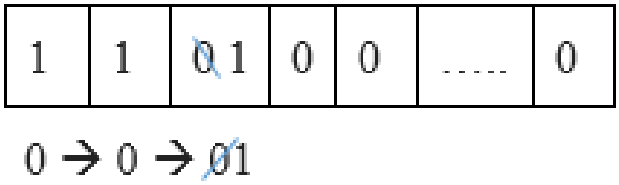


FIGURE 1.5 – Déroulement de l'incrémentation

Représentation graphique

On peut représentation les transitions sous forme d'une représentation graphique suivante :

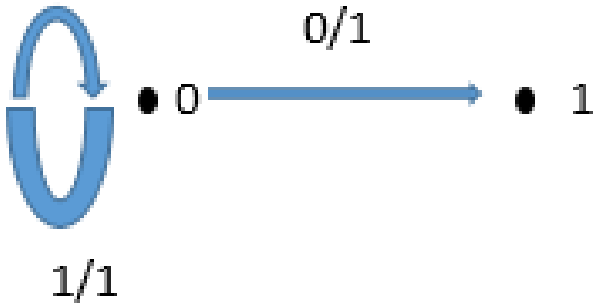


FIGURE 1.6 – Représentation graphique

1.9.5 Exemple 2 : Doubler le nombre de '1' en intercalant un '0' entre deux séries

La machine de Turing qui suit possède un alphabet '0', '1', '0' étant le « blanc ». On suppose que le ruban contient une série de '1', et que la tête de lecture/écriture se trouve initialement au-dessus du '1' le plus à gauche. Cette machine a pour effet de doubler le nombre de '1', en intercalant un '0' entre les deux séries. Par exemple, « 111 » devient « 1110111 ». L'ensemble d'états possibles de la machine est e1, e2, e3, e4, e5 et l'état initial est e1. La table d'actions est la suivante :

Ancien état	Symbole lu	Symbole écrit	Mouvement	Nouvel état
e1	**0**	*(Arrêt)*		
	1	0	Droite	e2
e2	**1**	1	Droite	e2
	0	0	Droite	e3
e3	**1**	1	Droite	e3
	0	1	Gauche	e4
e4	**1**	1	Gauche	e4
	0	0	Gauche	e5
e5	**1**	1	Gauche	e5
	0	1	Droite	e1

FIGURE 1.7 – Table de transition pour l'exemple 2

L'exécution de cette machine pour une série de deux '1' serait (la position de la tête de lecture/écriture sur le ruban est inscrite en caractères gras et rouges) :

Étape	État	Ruban
1	e1	**1**1
2	e2	0**1**
3	e2	01**0**
4	e3	010**0**

Étape	État	Ruban
5	e4	010**1**
6	e5	0101
7	e5	**0**101
8	e1	**1**101

Étape	État	Ruban
9	e2	100**1**
10	e3	100**1**
11	e3	1001**0**
12	e4	1001**1**

Étape	État	Ruban
13	e4	1001**1**
14	e5	1001**1**
15	e1	1**1**011
	(Arrêt)	

FIGURE 1.8 – Déroulement de l'exemple 2

Le comportement de cette machine peut être décrit comme une boucle :

— Elle démarre son exécution dans l'état e1, remplace le premier 1 par un 0.

— Puis elle utilise l'état e2 pour se déplacer vers la droite, en sautant les 1 (un seul dans cet exemple) jusqu'à rencontrer un 0 (ou un blanc), et passer dans l'état e3.

— L'état e3 est alors utilisé pour sauter la séquence suivante de 1 (initialement aucun) et remplacer le premier 0 rencontré par un 1.

— L'état e4 permet de revenir vers la gauche jusqu'à trouver un 0, et passer dans l'état e5.

— L'état e5 permet ensuite à nouveau de se déplacer vers la gauche jusqu'à trouver un 0, écrit au départ par l'état e1.

— La machine remplace alors ce 0 par un 1, se déplace d'une case vers la droite et passe à nouveau dans l'état e1 pour une nouvelle itération de la boucle.

Ce processus se répète jusqu'à ce que e1 tombe sur un 0 (c'est le 0 du milieu entre les deux séquences de 1) ; à ce moment, la machine s'arrête.

1.9.6 Exemple 3 : Machine à additionner

Dans cet exemple, tout nombre sera codé sur le ruban par une série de '1' (3 sera codé '111' , 4 par '1111', etc... chacun de ces nombres étant délimité par un séparateur #. Une addition tel que « 2+4=6 » pour cette machine peut donc être représentée par :

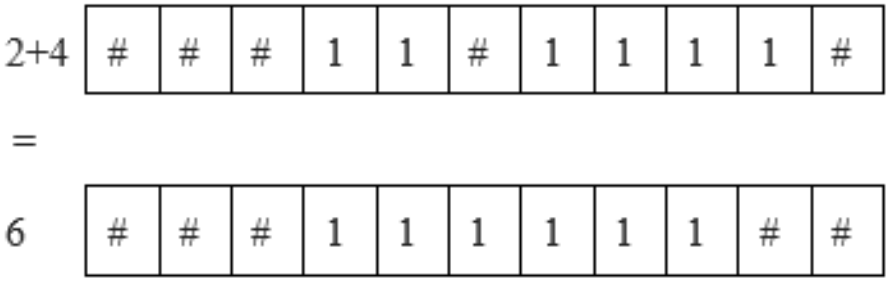


FIGURE 1.9 – Additionner deux nombre

On supposant que la tête de lecture/écriture se trouve initialement au-dessus du '1' le plus à gauche. Pour effectuer cette opération, la machine de Turing doit posséder au moins trois états q0,q1 et q2, et appliquer les règles suivantes :

	Situation de départ		Situation de l'étape suivante		
	Etats interne	Symbole lu	Symbole écrit	Décalage du ruban	Nouvel état
1	q0	1	1	Droite	q0
2	q0	#	1	Droite	q1
3	q1	1	1	Droite	q1
4	q1	#	#	Gauche	q2
5	q2	1	#	Stop	q2
6	q2	#	#	Stop	q2

FIGURE 1.10 – Table de transition pour l'exemple 3

La programmation avec des machines de Turing est extrêmement bas niveau. On peut toutefois programmer réellement beaucoup de choses avec ce modèle. La première étape est de se convaincre que plein de problèmes simples peuvent se programmer. A vrai dire, la seule façon de s'en convaincre est d'essayer soit même de programmer avec des machines de Turing, c'est-à-dire de faire les exercices qui suivent.

1.10 Exercices

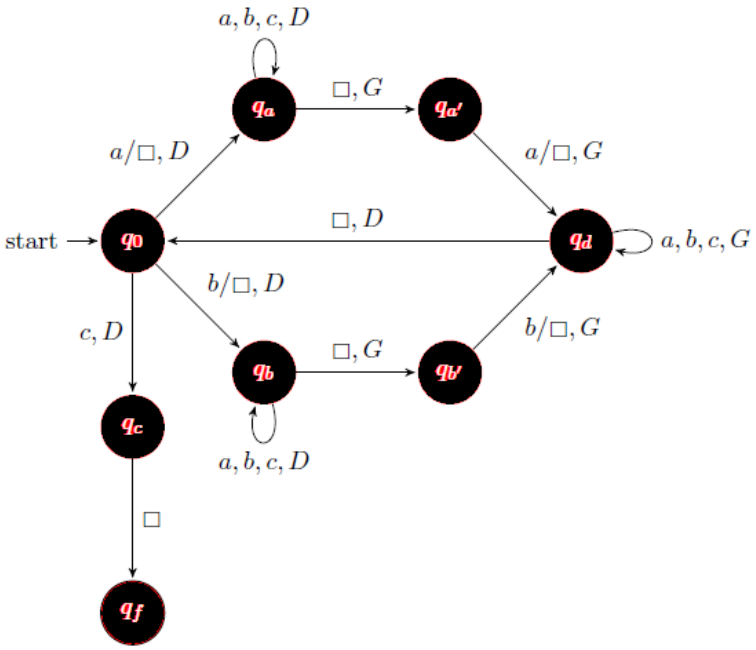
Exercice 1 :

Construire une machine de Turing acceptant le langage $\{ uc\overline{u} \mid u \in \{ab\}^*\}$

Solution :

Le mot miroir contenant des caractères a et b dont le séparateur est le caractère c. Le principe est le suivant :

– Lire le premier caractère a ou b du ruban l'effacer (avec un blanc) et conserver sa valeur dans l'état q_a ou q_b.
– Aller à la dernière case non vide (blanc) du ruban.
– vérifier que le caractère qui s'y trouve est bien le même et l'effacer.
– Revenir (état q_d) au premier caractère non blanc du ruban et recommencer tant que ce caractère est a ou b.
– Lorsque le premier caractére est c (état q_c) vérifier que la case suivante est bien vide et à cette condition entrer dans l'état final.



Exercice 2 :

Soit $\Sigma = \{a, b\}$.

a) Construire une machine de Turing qui prend un mot u sur son ruban d'entrée et écrit uu sur un ruban de sortie.

b) Construire une machine de Turing qui décide le langage L $= \{w \in \Sigma^* \mid w = w^R\}$.

Exercice 3 :

a) Construire une machine de Turing qui :

1- permet de Concaténer deux chaînes de caractères.

2- Permet d'inverser une chaine de caractères.

b) Construire une machine de Turing qui convertit en unaire un entier entré en binaire.

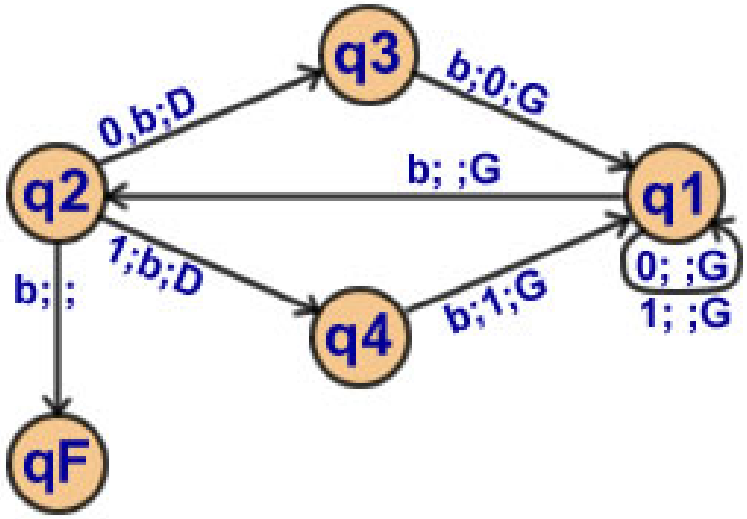
c) Construire une machine de Turing qui convertit en binaire un entier entré en unaire.

Solutions :

a)

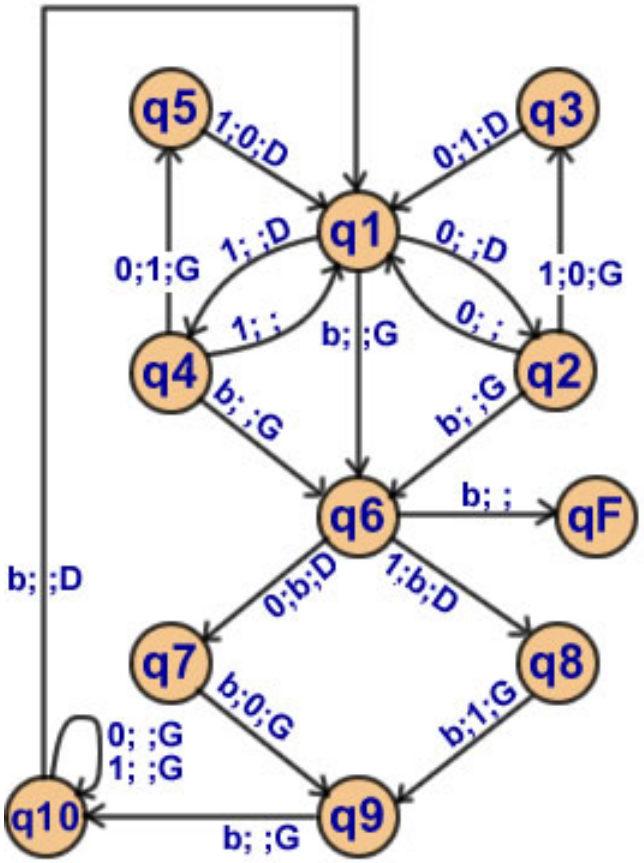
1- Concaténer deux chaînes de caractères :

Concaténer deux chaînes de caractères				
État	Lecture	Écriture	Déplacement	Nouvel état
1	b		Gauche	2
	0		Gauche	
	1		Gauche	
2	b			F
	0	b	Droite	3
	1	b	Droite	4
3	b	0	Gauche	1
	0			
	1			
4	b	1	Gauche	1
	0			
	1			



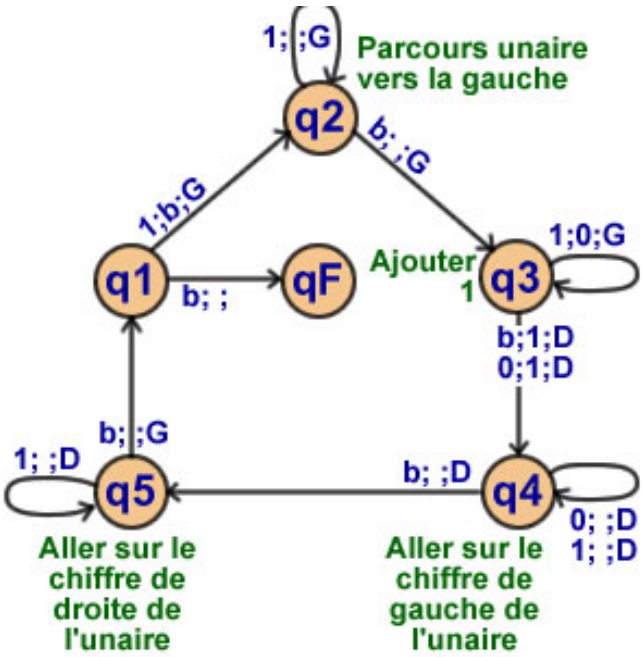
2- Inverser une chaine de caractères :

État	Lecture	Écriture	Déplacement	Nouvel état
1	b		Gauche	6
	0		Droite	2
	1		Droite	4
2	b		Gauche	6
	0			1
	1	0	Gauche	3
3	b			
	0	1	Droite	1
	1			
4	b		Gauche	6
	0	1	Gauche	5
	1			1
5	b			
	0			
	1	0	Droite	1
6	b			F
	0	b	Droite	7
	1	b	Droite	8
7	b	0	Gauche	9
	0			
	1			
8	b	1	Gauche	9
	0			
	1			
9	b		Gauche	10
	0			
	1			
10	b		Droite	1
	0		Gauche	
	1		Gauche	



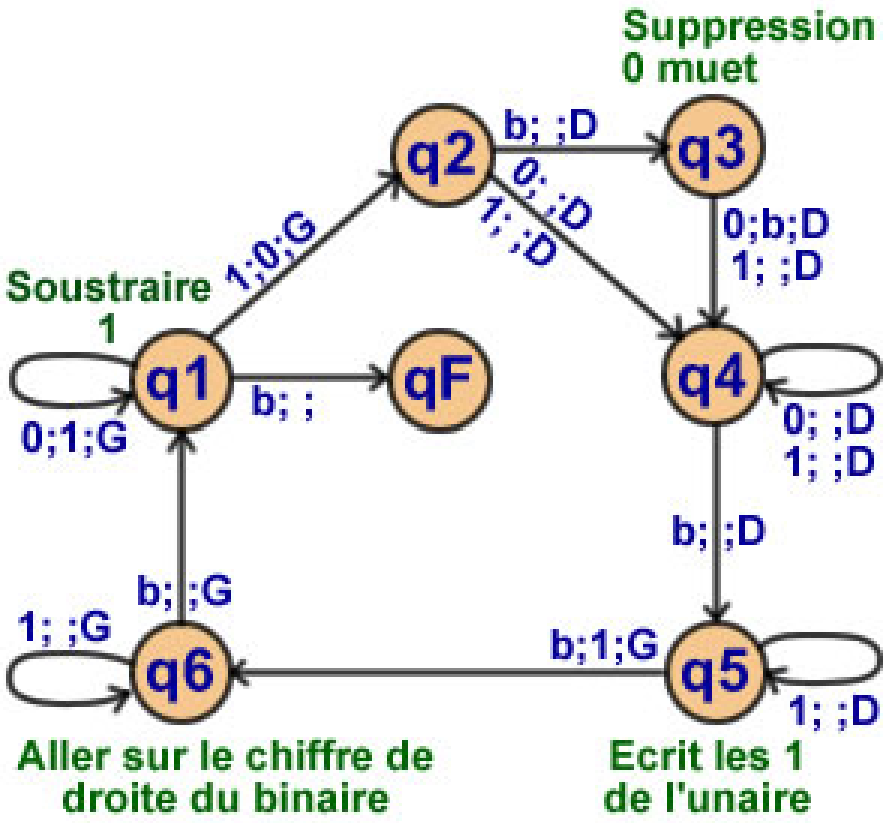
b) Ecriture d'un unaire en binaire :

Ecriture d'un unaire en binaire				
État	Lecture	Écriture	Déplacement	Nouvel état
1	b			F
	0			
	1	b	Gauche	2
2	b		Gauche	3
	0			
	1		Gauche	
3	b	1	Droite	4
	0	1	Droite	4
	1	0	Gauche	
4	b		Droite	5
	0		Droite	
	1		Droite	
5	b		Gauche	1
	0			
	1		Droite	



c) Ecriture d'un binaire en unaire :

Binaire en unaire				
État	Lecture	Écriture	Déplacement	Nouvel état
1	b			F
	0	1	Gauche	
	1	0	Gauche	2
2	b		Droite	3
	0		Droite	4
	1		Droite	4
3	b			
	0	b	Droite	4
	1		Droite	4
4	b		Droite	5
	0		Droite	
	1		Droite	
5	b	1	Gauche	6
	0			
	1		Droite	
6	b		Gauche	1
	0			
	1		Gauche	



Exercice 4 :

Concevez une machine de Turing (MT) qui calcule la somme binaire de deux entiers positifs.

Solution :

Faire la somme de deux nombres binaires M et N est généralisation simple du cas du successeur binaire et du prédécesseur binaire déjà corrigés. Par exemple, l'addition de M = 110 à N = 100 peut s'effectuer en supprimant 1 de N et en ajoutant 1 à M jusqu'à ce que N devient 0 et la somme est N. L'exemple suivant montre le principe :

M + N = 110 + 100
= 111 + 011
= 1000 + 010
= 1001 + 001
= 1010 + 000

On suppose que la MT à lire la bande ci-dessus à l'extrême droite. L'algorithme suivant explique le déroulement de la machine.

Tant que N contient des 1.

Supprimer 1 de N

Aller sous le chiffre de droite de M

Ajouter 1 à M

Aller sous le chiffre de gauche de N

Parcourir N vers la droite

Si la tête trouve 1

Passage à l'état q5 et poursuivre

Sinon

Sortie de tant que et aller à l'état final qF

Aller sous le chiffre de droite de N et repasser à l'état q0

Fin tant que

La liste des configurations de la machine est donnée par le tableau suivant :

Etat	Lecture	Ecriture	Déplacement	Nouvel Etat
q0	#			
	0	1	gauche	
	1	0	gauche	q1
q1	#		gauche	q2
	0		gauche	
	1		gauche	
q2	#	1	droite	q3
	0	1	droite	q3
	1	0	gauche	
q3	#		droite	q4
	0		droite	
	1		droite	
q4	#			qF
	0		droite	
	1		droite	q5
q5	#		gauche	q0
	0		droite	
	1		droite	

FIGURE 1.11

Cependant on pourra penser à supprimer les 0 restants de N.

Chapitre 2

Récursivité et décidabilité

Sommaire

2.1 Introduction

Le calcul mathématique peut se réduire à une succession d'opérations élémentaires (songez à la multiplication entière comme une série d'additions). Les nombres calculables sont les nombres qui sont générables en un nombre fini d'opérations élémentaires. De la même manière, une fonction mathématique sera dite calculable s'il existe une suite finie d'opérations élémentaires permettant de passer d'un nombre x à son image f(x).

On retrouve cette notion d'opérations élémentaires dans les machines de Turing. Cette machine (théorique) permet de simuler tout ce qu'un programme informatique (une suite d'instructions) est capable d'exécuter. Un algorithme peut se réduire à une suite d'opérations élementaires, comme une fonction mathématique peut se réduire à une suite de calculs. Dès lors, on pourra considérer un algorithme comme une fonction.

Turing a démontré que l'ensemble des fonctions calculables, au sens de Church, était équivalent à l'ensemble des fonctions programmables sur sa machine. Certaines fonctions peuvent être calculables, ou ne pas l'être : c'est notamment le cas de notre fonction du problème de l'arrêt.

La notion de calculabilité jette un pont entre les mathématiques (la vision de Church, pour schématiser) et l'informatique (la vision de Turing) n'est pas simple à définir !

2.2 Rappel sur les ensembles

2.2.1 Les ensembles finis

a- Généralités :

Un ensemble peut s'apparenter à une liste d'objet ayant des propriétés communes. Par exemple, l'ensemble des nombres entiers pairs (ensemble infini), ou encore l'ensemble des nombres entiers de 0 à 100, l'ensemble des fonctions du second degré...

Un ensemble est entiérement défini par les éléments qui le constituent. Par convention, on note les éléments d'un ensemble entre accolades : A = {1 ;3 ;5 ;7 ;11 ;13}.

L'ensemble ne contenant aucun élément est appelé ensemble vide et est noté ϕ.

b- Union et intersection de deux ensembles :

Notation :

$\bigcap$ se lit « inter » et représente l'intersection de deux ensembles.

$\bigcup$ se lit « union » et représente la réunion de deux ensembles.

Soient A et B deux ensembles :

A $\bigcap$ B est l'ensemble contenant tous les éléments qui sont dans A ou dans B.

A $\bigcup$ B est l'ensemble contenant tous les éléments qui sont dans A et dans B.

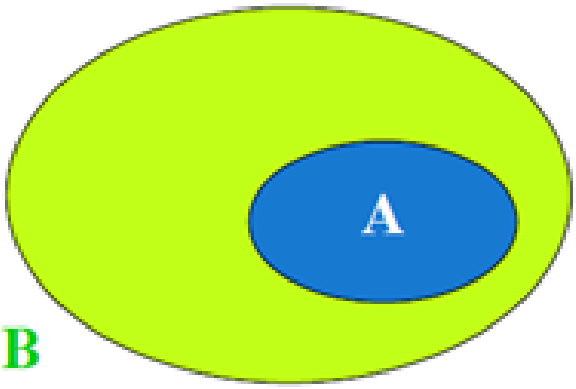


FIGURE 2.1 – Union et intersection de deux ensembles

Soit l'élément e :

- e $\in$ A$\bigcup$ B si et seulement si e $\in$ A ou e $\in$ B.
- e $\in$ A $\bigcap$ B si et seulement si e $\in$ A et e $\in$ B.

Remarque : si A et B n'ont aucun élément en commun, on a A $\bigcap$ B = ϕ.

c- Inclusion :

Un ensemble A est inclus dans un ensemble B (On note A $\subset$B) si tous les éléments de A appartiennent à B.

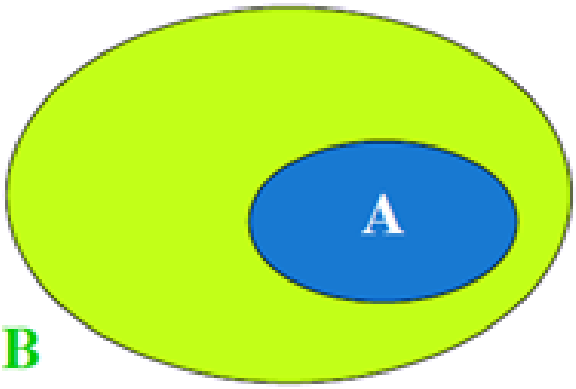


FIGURE 2.2 – Inclusion

d- Propriétés de relations :

Une relation R sur un ensemble E peut avoir les propriétés suivantes :

— Réflexivité : $\forall x \in E \quad (x, x) \in R$

— Symétrie : $\forall x, y \in E (x, y) \in R \Rightarrow (y, x) \in$

— Transitivité : $\forall x, y, z \in E, \quad (x, y) \in R \bigwedge (y, z) \in R \Rightarrow (x, z) \in \ R$

— Antisymétrie : $\forall x, y \in E, \quad (x, y) \in R \bigwedge (y, x) \in R \Rightarrow x = y$

— Relation d'équivalence : Une relation réflexive, symétrique et transitive est une relation d'équivalence.

— Relation d'ordre : Une relation R réflexive, antisymétrique et transitive est une relation d'ordre.

e- Cardinal d'un ensemble fini :

Le cardinal d'un ensemble fini E est le nombre d'éléments contenu dans cet ensemble. On le note Card(E).

2.2.2 Les ensembles infinis

En mathématiques, nous travaillons avec cinq grands ensembles de base, qui permettent de manipuler les nombres. Ces ensembles sont parfois complémentaires et peuvent aussi se distinguer par les types de nombres qu'ils contiennent. Une étude, du plus petit au plus grand de ces cinq grands ensembles de base (il en existe d'autres), nous permet non seulement de définir et de présenter chaque ensemble, mais aussi de ressortir avec des exemples à l'appui sur leurs différences. Ces ensembles sont :

— $\mathbb{N}$: Ensemble des entiers naturels.

— $\mathbb{Z}$: Ensemble des entiers relatifs.

— $\mathbb{D}$: Ensemble des nombres décimaux.

— $\mathbb{Q}$: Ensemble des nombres rationnels.

— $\mathbb{R}$: Ensemble des nombres réels.

L'ensemble $\mathbb{N}$:

C'est l'ensemble des nombres entiers naturels. Un entier naturel est un nombre positif ou nul, permettant de compter des objets. Exemples : 0, 1, 2, 3, 4, 5, 6, etc.

L'ensemble $\mathbb{Z}$:

C'est l'ensemble des nombres entiers relatifs. Un entier relatif est, non seulement, un entier naturel, mais se présente aussi comme un entier naturel muni d'un signe positif ou négatif. Exemples : ....-5, -4, -3, -2, -1, 0, +1, +2, +3, +4, +5, +6, +7, +8, etc.

L'ensemble $\mathbb{D}$:

C'est l'ensemble des nombres décimaux relatifs. Un nombre décimal relatif est, non seulement, un nombre entier relatif, mais peut aussi être un nombre à virgule flottante, positif ou négatif. Exemples : ....-5, -4, -4.2, -3, -2, -1.5, -1, 0, +0.7, +1, +2, +2.4, +3, +4, +5, +6, +6.75 +7, +8, etc.

L'ensemble $\mathbb{Q}$:

C'est l'ensemble des nombres rationnels. Un nombre rationnel est, non seulement, un nombre décimal relatif, mais peut aussi être un nombre qui peut s'exprimer avec le quotient de deux entiers relatifs. Le dénominateur étant non nul. Exemples : ....-5/4, -4, -4.2, -3, -2, -1.5, -1/2, 0, +0.7, +1, +2, +2.4, +3, +4/5, +5, +6, +6.75, +7/2, +8.

L'ensemble $\mathbb{R}$:

C'est l'ensemble des nombres réels. Un nombre réel est non seulement un nombre rationnel, mais peut aussi être un nombre dont le développement décimal est infini, et non périodique. Exemples : ....-5/4, -4, -4.2, -3, -2, -1.524, -1/2, 0, +0.7, +1, +2, +2.41, +3, +4/5, +5, +6, +6.75, +7/2, +8...

$\mathbb{N}$ est inclus dans $\mathbb{Z}$.

$\mathbb{Z}$ est inclus dans$\mathbb{D}$.

$\mathbb{D}$ est inclus dans $\mathbb{Q}$.

$\mathbb{Q}$ est inclus dans$\mathbb{R}$.

Exprimé mathématiquement cela donne :

$\mathbb{N} \subset \mathbb{Z} \subset \mathbb{D} \subset \mathbb{Q} \subset \mathbb{R}$

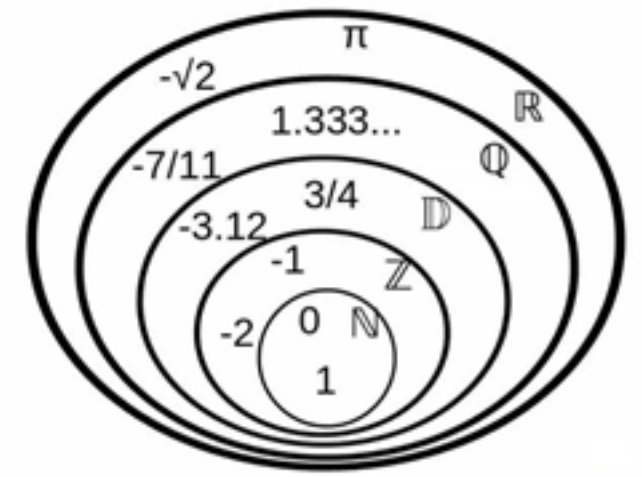


FIGURE 2.3 – Ensembles infinis

Les ensembles infinis suivants sont aussi utilisés sans cesse :

$\mathbb{N}$ = ensemble des entiers naturels: $0, 1, 2, \ldots$

$\mathbb{N}^*$ = ensemble des entiers naturels non nuls: $1, 2, \ldots$

$\mathbb{Z}$ = ensemble des entiers relatifs: $\ldots, -2, -1, 0, 1, 2, \ldots$

$\mathbb{Q}$ = ensemble des rationnels

$\mathbb{R}$ = ensemble des réels = $]-\infty, \infty[$

$\mathbb{R}^d$ = espace euclidien réel de dimension d (donc $\mathbb{R}^1 = \mathbb{R}$)

$\overline{\mathbb{R}} = [-\infty, \infty]$

$\mathbb{R}_+ = [0, \infty[$

$\overline{\mathbb{R}}_+ = [0, \infty]$

$\mathbb{C}$ = ensemble des nombres complexes.

2.3 Fonctions injectives, surjectives et bijectives

2.3.1 Injection

Une fonction g est dite injective si et seulement si tout réel de l'image correspond au plus à un seul réel du domaine de définition. En notation mathématique, on a :

$$\forall x1, x2 \in dom(g) : g(x1) = g(x2) \Rightarrow x1 = x2$$

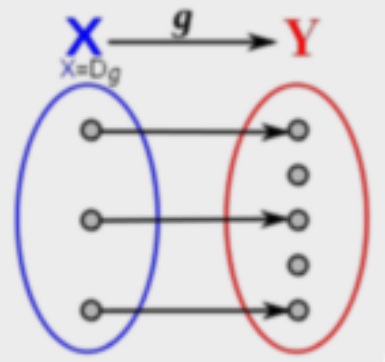


FIGURE 2.4 – Injection

Exemples de fonctions injectives :

- La fonction $f : \mathbb{R} \longrightarrow \mathbb{R} : x \longmapsto x^2$ n'est pas injective. En effet, $f(-1) = 1 = f(1)$ mais $-1 \neq 1$.
- La fonction $f : \mathbb{R} \longrightarrow \mathbb{R} : x \longmapsto 2x$ est injective. En effet, si $2x = 2y$ alors $x = y$.
- La fonction $f : \mathbb{R}^+ \longrightarrow \mathbb{R}^+ : x \longmapsto x^2$ est injective. En effet, si $x^2 = y^2$ avec $x, y \in \mathbb{R}^+$, alors $x = y$.

2.3.2 Surjection

Une fonction f est dite surjective si et seulement si tout réel de l'image correspond à au moins un réel du domaine de définition. En notation mathématique, on a :

$$\forall y \in im(f)\ (\exists x | f(x) = y)$$

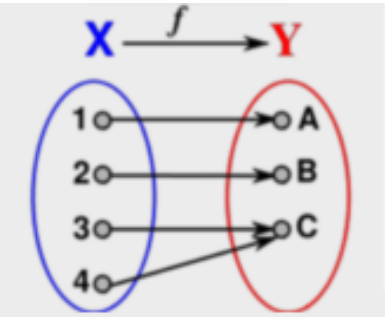


FIGURE 2.5 – Surjection

Exemples de fonctions surjectives :

— La fonction $f : \mathbb{R} \longrightarrow \mathbb{R} : x \longmapsto 2x$ est surjective. En effet, tout $x \in \mathbb{R}$ est l'image par f d'un réel : $f(\frac{x}{2}) = x$.

— La fonction $f : \mathbb{R} \longrightarrow \mathbb{R} : x \longmapsto x^2$ n'est pas surjective. En effet, $\nexists$ x$\in \mathbb{R}$ tel que : $f(x) = -1$.

— La fonction $f : \mathbb{Z} \longrightarrow \mathbb{Z} : z \longmapsto 2z$ n'est pas surjective. En effet, $\nexists$ z$\in \mathbb{Z}$ tel que : $f(z) = 1$.

2.3.3 Bijection

Une fonction h est dite bijective si et seulement si elle est et injective et surjective. En notation mathématique, on a :

$$\forall x1, x2 \in dom(h) : h(x1) = h(x2) \Rightarrow x1 = x2 \text{ ET } \forall y \in im(h)\ (\exists x | h(x) = y)$$

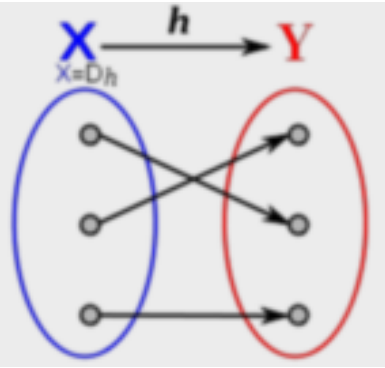


FIGURE 2.6 – Bijection

Exemples de fonctions bijectives :

— $f(x) = x$

— $f(x) = x^a$ (a impair)

— $f(x) = \sqrt{x}$

2.4 Dénombrabilité

2.4.1 Dénombrement des ensembles finis

Afin de comparer la taille des différents ensembles, on dispose en mathématique de la notion de cardinalité. Par exemple un ensemble de 3 pommes aura une cardinalité de 3. Et un ensemble de 3 ballons aura une cardinalité de 3 aussi. Mais on peut aussi par exemple associer la pomme verte au ballon foot, la pomme rouge au ballon basket et la pomme jaune au ballon de rugby. Ainsi chaque élément des 2 ensembles est associé un et un seul élément de l'autre ensemble. C'est ce qu'on appelle en mathématique une bijectivité.

A partir de cette correspondance, on peut conclure que deux ensembles possèdent la même cardinalité, donc ils sont de la même taille. Un ensemble est dénombrable si on peut le lister. D'une façon général, un ensemble A est dénombrable, si on peut lister les éléments de A en mettant en correspondance complète avec les entiers naturels N comme suit :

A	N
a0	↔ 0
a1	↔ 1
a2	↔ 2
a3	↔ 3

on dit que A est en en bijection avec $\mathbb{N}$.

2.4.2 Dénombrement des ensembles infinis

Tout ensemble fini est dénombrable, et c'est en comparant différents ensembles, cette fois-ci de grands tailles que Cantor, un mathématicien Allemand, s'est aperçu que certains ensembles sont plus grands que d'autres. Il a résolu d'une manière rigoureuse la notion de l'infini.

Cantor s'est posé la question suivante : Etant donné un ensemble A infini, et un ensemble B infini. Est-ce qu'un de ces ensembles pouvait être supérieur à l'autre ? autrement dit le nombre d'éléments de A peut être supérieur au nombre d'éléments de B, d'une manière mathématique :

$$|A| > |B| \, ?$$

Avant Cantor, la question n'avait aucun sens, mais en fait ce qu'a fait le mathématicien, c'est de définir une relation entre les ensembles infinis de la manière suivante : il a pris comme référence l'ensemble des entiers naturels $\mathbb{N}$! pourquoi $\mathbb{N}$, parceque c'est l'ensemble qui nous

permet de compter les choses, c'est pourquoi c'est l'ensemble le plus naturel pour l'être humain (ce qui est intéressant c'est qu'on peut lister ou énumérer les éléments de $\mathbb{N}$) et c'est cette nature intrinsèque des entiers naturelles, qu'on va essayer en mathématique de la généraliser à des ensembles plus infinis et dire qu'un ensemble est dénombrable si on peut le lister, comme on peut le faire pour $\mathbb{N}$.

D'une manière général, un ensemble A est dénombrable, si on peut lister les éléments de A en mettant en correspondance complète avec $\mathbb{N}$.

On dit qu'un ensemble infini A a la même cardinalité que $\mathbb{N}$ est dénombrable, s'il existe une bijection de $\mathbb{N}$ vers A.

Dénombrement des entiers relatifs $\mathbb{Z}$:

L'ensemble $\mathbb{Z}$ est constitué de nombres positifs et des nombres négatifs. . . . . .-4 -3 -2 -1 0 1 2 3 4 . . . . . .

La manière la plus facile de dénombrer l'ensemble $\mathbb{Z}$, c'est de redisposer les éléments de $\mathbb{Z}$ comme suit :

$\mathbb{N}$: 0 1 2 3 4 5 6 . . . . . .

$\mathbb{Z}$: 0 1 -1 2 -2 3 -3 . . . . . .

Un ensemble $\mathbb{Z}$ sera selon la définition de cantor dénombrable, s'il existe une bijection de $\mathbb{N}$ vers Z (qui associe à $\mathbb{N}$ un seul élément de $\mathbb{Z}$ et inversement). On peut facilement trouver une fonction F qui correspond à ce qu'on appelle un « dénombrement de $\mathbb{Z}$ ». La relation bijective F est de la manière suivante :

$$F(x) = \begin{array}{ll} -x/2 & \text{si } x \text{ est pair} \\ (x+1)/2 & \text{si } x \text{ est impair} \end{array}$$

On peut conclure que : $|\mathbb{Z}|=|\mathbb{N}|$

Exemple :

L'ensemble des nombres pairs $2\mathbb{N}$ est-t-il dénombrable ?

Autrement dit : Est-ce que ces 2 ensembles ($\mathbb{N}$ et $2\mathbb{N}$) ont la la même cardinalité. Ou encore, dans la terminologie de cantor, est-ce-qu'il existe une fonction bijective de $\mathbb{N}$ vers $2\mathbb{N}$.

De la même manière, on représente les éléments de$\mathbb{N}$ et $2\mathbb{N}$ comme suit :

$\mathbb{N}$: 0 1 2 3 4 5 6. . . . . .

$2\mathbb{N}$: 0 2 4 6 8 10 12 . . . .

Et on essaye de trouver une fonction bijective de $\mathbb{N}$ vers $2\mathbb{N}$.

La fonction $F(x) = 2x$, permet d'associer à chaque nombre de $\mathbb{N}$, un et un seul nombre de $2\mathbb{N}$. Donc de la même manière que $\mathbb{Z}$, l'ensemble des nombres pairs $2\mathbb{N}$, a le même nombre d'éléments que l'ensemble $\mathbb{N}$. On conclue que : $|2\mathbb{N}|=|\mathbb{N}|$

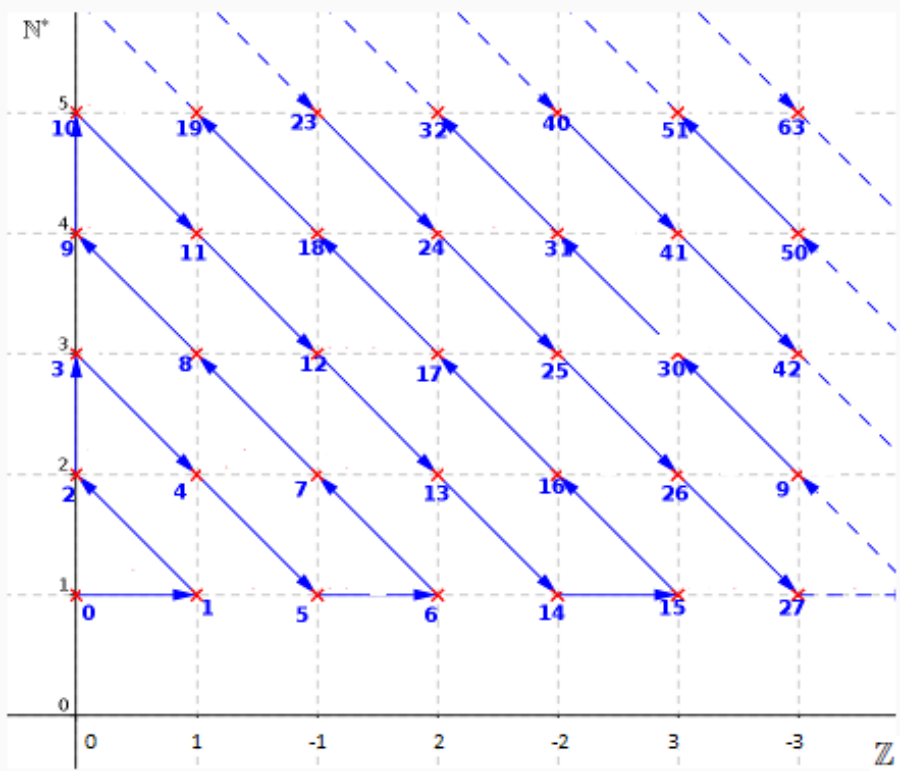
Dénombrement des nombres rationnels $\mathbb{Q}$:

Pour pouvoir voir la dénombrabilité de $\mathbb{Q}$, il faut se rappeler de cette définition :

$\mathbb{Q}$:p/q avec p$\in \mathbb{Z}$ et q$\in \mathbb{N}^*$

En d'autre terme, : $\mathbb{Q}$: peut être mis en correspondance avec $\mathbb{Z}x\mathbb{N}*$: Prendre un rationnel, revient à prendre un couple (entier relatif, entier naturel non nul).

Cette correspondance peut être représentée graphiquement comme suit :



À chaque nombre entier naturel n écrit en bleu sur l'image, on associe un et un seul élément de $\mathbb{Z}x\mathbb{N}*$ de manière bijective. Ce dessin est infini et recouvre tous les couples de points. On

parcourt ainsi l'ensemble $\mathbb{Z}x\mathbb{N}*$ en zigzag, et on associe un numéro à chacun de ses éléments. Donc $\mathbb{Q}$ est dénombrable : $|\mathbb{Q}| = |\mathbb{N}|$

L'ensemble des nombres réels $\mathbb{R}$ est-il dénombrable ?

Une question qui se pose : est-ce que tous les ensembles infinis ont le même nombre d'éléments que l'ensemble $\mathbb{N}$?

En fait, cantor a montré que ce n'était pas le cas, et en particulier l'ensemble des réels $\mathbb{R}$. L'ensemble des réels contient plus d'éléments que $\mathbb{N}$: $|\mathbb{R}| > |\mathbb{N}|$. Juste l'intervalle [0,1] contient plus d'éléments que $\mathbb{N}$: $|[0,1]| > |\mathbb{N}|$, imaginez alors tous les intervalles des nombres réels.

Démonstration :

Pour montrer que l'ensemble des réels de l'intervalle]0,1[est non dénombrable. Chaque nombre réel compris entre 0 et 1 possède un développement décimal illimité ou deux. Les nombres non décimaux en ont un seul, les nombres décimaux possèdent deux développements :

Par exemple : 0,95=0,95000=0,94999...

L'un comporte des 9 indéfiniment, on l'appelle développement impropre, l'autre, qu'on appelle développement propre, comporte seulement un nombre fini de chiffres non nuls après la virgule. Si on convient de n'utiliser que des développements propres, chaque nombre réel est donc associé à un unique développement propre ; si deux développements propres différent en un chiffre, ils sont les développements de nombres réels distincts.

On suppose que les nombres réels de l'intervalle]0,1[peuvent être rangés en une suite A0,A1,A2,..., et on écrit les développements décimaux propres de ces nombres :

A0=0,a$_{00}a_{01}a_{02}a_{03}a_{04}$...

A1=0,a$_{10}a_{11}a_{12}a_{13}a_{14}$...

A2=0,a$_{20}a_{21}a_{22}a_{23}a_{24}$...

A3=0,a$_{30}a_{31}a_{32}a_{33}a_{34}$...

:

An=0,a$_{n0}a_{n1}a_{n2}a_{n3}a_{n4}$...

où les a$_{ij}$ sont des chiffres .

Le procédé diagonal de Cantor : Le principe de la démonstration est de montrer qu'au moins un nombre réel échappe à ce classement et donc qu'il y a une contradiction.

On fabrique un nombre réel B par un procédé appelé procédé diagonal en prenant $B = 0, b_0 b_1 b_2 b_3 \ldots$ en imposant $b_0 a_{00}$, $b_1 a_{11}$, $b_2 a_{22}$, etc$\cdots$ et que de plus les b_i ne soient pas égaux à des 9 indéfiniment.

Comme on ne travaille qu'avec des développements propres, il est sûr que B_0 car le premier chiffre après la virgule de ces deux nombres diffère, B_1 car le deuxième chiffre après la virgule de ces deux nombres diffère, B_2 car le troisième chiffre après la virgule de ces deux nombres diffère, etc$\cdots$ Le nombre B n'a pas été classé et on a obtenu une contradiction. On ne peut donc pas ranger les nombres réels de l'intervalle]0,1[en une suite.

Cet ensemble n'est pas dénombrable. On en déduit immédiatement que l'ensemble des nombres réels R est non dénombrable : $|\mathbb{R}| > |\mathbb{N}|$.

La méthode la plus efficace pour convaincre la communauté mathématique c'est de fournir une preuve de consistance (des axiomes : i.e une preuve formelle dont ses axiomes ne sont pas contradictoire).

2.5 Calculabilité

La calculabilité (ou récursivité) cherche d'une part à identifier la classe des fonctions qui peuvent être calculées à l'aide d'un algorithme et à appliquer ces concepts à des questions fondamentales des mathématiques. Une bonne appréhension de ce qui est calculcable et de ce qui ne l'est pas permet de voir les limites des problèmes que peuvent résoudre les ordinateurs . Parmi les modèles de calcul utilisés en calculabilité, on peut citer les machines de Turing. Dans le chapitre dans lequel on a décrit la machine de Turing avec des exemples simples qu'on a résolus ; on peut penser que seuls les programmes simples peuvent être exécutés par la machine de Turing, en fait, c'est faux. La thèse de Church/Turing stipule la chose suivante : Pour tout problème pour lequel il existe un algorithme qui résoud ce probléme P, alors il existe une machine de turing qui résoud le probléme P :

$$\forall P \; \exists A \; tel \; que \; A \; resoud(P) \Rightarrow \exists M \; tel \; que \; M \; resoud(P)$$

Grâce à cette thèse, la notion de calculabilité possède désormais une signification précise.

Une fonction f Calculable, signifie qu'il existe un algorithme au sens de la machine de

Turing qui permet de la calculer. Même la notion d'algorithme pour la première fois dans l'histoire des maths a une signification. Un algorithme est une table de transition au sens de la machine de Turing. Parmi les généralisation possibles de la machine Turing, il existe une machine particulièrement intéressante qui est la machine de Turing universelle notée : Machine U. En informatique théorique, une machine de Turing universelle est une machine de Turing qui peut simuler n'importe quelle machine de Turing sur n'importe quelle entrée. Une machine universelle prend en entrée la description de la machine à simuler et l'entrée de cette dernière.

En 1936, Alan Turing a proposé le concept de machine universelle. Il montrait alors comment un système logique minimum permettait de déterminer toutes fonctions calculables. Ce travail a donné lieu à la thèse de Church-Turing, qui est à l'origine de l'ordinateur qu'on utilise aujourd'hui qui n'est rien d'autre qu'une machine universelle de Turing finie. Dans sa version de base, elle énonce que toute fonction calculable l'est par une machine de Turing. Cette thèse a été élargie dans ce que l'on qualifie de « version physique » selon laquelle une machine de Turing peut réaliser toute opération de traitement de l'information réalisable par un système physique.

L'idée est la même, quand on veut exécuter un programme on va passer comme paramètres pour la machine universelle U, d'abord la table de transition, ensuite le nombre en entrée (ou en lecture : voir l'exemple incrementer(n) dans le contexte de la machine de Turing). Ainsi, l'entrée pour la machine U est une table de transition et un élément en entrée. La thèse de Church-Turing devient encore plus simple :

$$\forall P \; \exists A \; tel \; que \; A \; resoud(P) \Rightarrow \; U \; resoud(P)$$

On vient de voir les principaux fondamentaux des ordinateurs d'aujourd'hui : - La notion de registre $\mapsto$ Etat - La notion de configuration $\mapsto$ tête de lecture - La notion de programme $\mapsto$ table de transition qu'on passe comme paramètre - Cycle d'instruction (ce qu'on exécute à un instant donné)$\mapsto$ déplacement case par case

Tous ces éléments sont les fondements des ordinateurs d'aujourd'hui et se sont aussi les fondements de l'informatique théorique : calculabilité, décidabilité, complexité.

2.6 Décidabilité

Bien avant l'existence des ordinateurs, les mathématiciens et les logiciens se sont intéressés aux méthodes de résolution des problèmes. La question était la suivante : « Chaque problème

a-t-il une méthode de résolution ? Tout problème peut-il être résolu par une méthode systématique, par un algorithme ? ». La réponse est NON.

En théorie de la calculabilité, on dit qu'un ensemble est décidable s'il existe un algorithme qui, étant donné un élément x, détermine en un nombre fini d'étapes si x appartient ou non à l'ensemble. Un ensemble est récursivement énumérable s'il existe un algorithme qui, étant donné un élément, répond oui en un nombre fini d'étapes si l'élément appartient à l'ensemble. Si l'élément n'appartient pas à l'ensemble, l'algorithme peut répondre non, ou bien tourner sans s'arrêter. De façon équivalente, un ensemble est récursivement énumérable s'il existe un algorithme qui liste tous les éléments de l'ensemble.

Un sens précis au mot "algorithme" peut être donné dans les définitions précédentes. Cela peut être par exemple les machines de Türing. On montre qu'il existe des ensembles qui sont récursivement énumérables, mais qui ne sont pas récursifs, un ensemble étant récursif si lui, ainsi que son complémentaire, sont récursivement énumérables.

Un problème est décidable si l'ensemble de ses solutions est récursif. Il est dit semi-décidable si l'ensemble de ses solutions est récursivement énumérable. Les problèmes décidables sont donc ceux que l'on sait résoudre à l'aide d'un algorithme en un temps fini. L'étude de l'indécidabilité des théories mathématiques relève de la théorie des fonctions récursives.

Un problème est caractérisé par les trois propriétés suivantes :

1. Le problème est une question générique qui s'applique à un ensemble d'éléments.

Exemple : « Déterminer si un nombre entier positif est pair ou impair » est un problème qui s'applique sur l'ensemble des entiers positifs.

2. Chaque instance du problème a une réponse.

Exemple : Si on considère le problème précédent, chaque entier positif a une réponse : on peut dire s'il est pair ou impair.

3. Le problème n'est pas défini par un algorithme ou par un programme.

Exemple : Le problème précédent ne dépend pas directement d'un algorithme, mais de la définition de ce qu'est un nombre pair.

Un problème est décidable si on peut le résoudre avec une méthode systématique, c'est-à-dire si chacune de ses instances sont résolubles par le même algorithme.

Exemple : Le problème « Déterminer si un nombre entier positif est pair ou impair » est

décidable car, si on considère un nombre entier positif quelconque, il suffit d'observer son chiffre des unités pour déterminer s'il est pair ou impair. Si ce chiffre est égal à 0, 2, 4, 6 ou 8, le nombre est pair, il est impair sinon.

Il existe des problèmes indécidables. Cela ne signifie pas simplement que nous ne savons pas les résoudre, mais qu'il n'existe aucune méthode systématique pour les résoudre.

Généralement, les démonstrations de décidabilité ou d'indécidabilité sont complexes à établir et sont hors programme.

Exemple : Un problème célèbre est celui du pavage du plan par des polygones. Paver le plan avec des polygones signifie recouvrir le plan avec ces polygones sans qu'ils ne se chevauchent et sans qu'il y ait d'espace entre eux. Le problème se pose de la manière suivante : « Déterminer si une collection de polygones peut paver le plan ou non. »

Ce problème est indécidable. Il n'existe pas d'algorithme, pas de méthode systématique qui détermine si une collection de polygones peut paver le plan ou non. Cette indécidabilité a été démontrée dans les années 1960.

Connaitre l'indécidabilité d'un problème a les deux avantages suivants :
a- Savoir qu'il est inutile de chercher un algorithme de résolution du problème.
b- Orienter l'étude du problème vers des sous-problèmes qui pourraient, quant à eux, être décidables.
Exemple : Si on réduit le problème du pavage à des carrés, qui sont des polygones particuliers (même de dimensions différentes), le sous-problème ainsi obtenu est décidable.

Un problème P est dit décidable s'il existe un algorithme A, une procédure qui termine en un nombre fini d'étapes, qui le décide, c'est-à-dire qui réponde par oui (ou vrai) ou par non(ou faux) à la question posée par le problème. S'il n'existe pas de tels algorithmes, le problème est dit indécidable.

$$\text{Decidabilité(P)} \Leftrightarrow \exists \text{ A / A résoud (P)}$$

On peut résumer la notion de décidabilité algorithmique d'un probléme P d'une manière

générale comme suit :

- P est décidable ⇔ il existe un algorithme pour P (c'est à dire une procédure qui s'arrête et répond oui si P est vrai, et répond par Non si P est faux).
- P est indécidable ⇔ il n'existe pas d'algorithme pour P.
- P est semi-décidable ⇔il existe un semi-algorithme pour P (c'est à dire une procédure qui s'arrête et répond oui si P est vrai, et ne donne pas de réponse si P est faux).

Il est clair qu'un problème décidable est aussi semi-décidable.

Remarque : Pour montrer qu'un problème est décidable, il suffit de trouver un algorithme (un suffit, et ceci peut se faire sans utiliser la théorie de calculabilité). Par contre, pour montrer qu'un problème est non-décidable, il faut considérer tous les algorithmes possibles et montrer qu'aucun d'eux ne résout le problème, ce qui est plus difficile voir impossible sans la notion rigoureuse d'algorithme.

Apair(x) vrai si $\frac{x}{2} \in \mathbb{Z}$

faux si $\frac{x}{2} \notin \mathbb{Z}$

Le problème qui permet de savoir si le nombre est pair ou pas est décidable. Pourquoi : parcequ'il est facile de présenter un algorithme qu'on appelle Apair(x), qui pour chaque élément x, permet de savoir s'il est pair ou pas. On suppose que la fonction Apair est de type booléenne.

2.7 Récursivité et problème d'arrêt

Partant des notions de décidabilité et calculabilité précédentes, nous pouvons tirer les points clés suivants :

- Tout problème n'a pas forcément une méthode systématique de résolution. On parle de problème indécidable.
- A contrario, les problèmes qui ont au moins une méthode systématique de résolution sont dits décidables.
- Le problème de l'arrêt est indécidable : il n'existe pas de méthode systématique pour démontrer qu'un programme s'arrête ou non.
- Les notions de calculabilité et de décidabilité sont équivalentes.

Maintenant, une question reste : « Toutes les fonctions sont-elles calculables ? » La réponse fut tranchée dans les années 1930. Toutes les fonctions ne sont pas calculables.

Etant donné un programme, peut-on savoir si, une fois démarré, il finira par s'arrêter, ou s'il tournera indéfiniment ? la réponse est non. Aucun programme ne peut exister pour cela : Le problème de l'arrêt en est un exemple.

Certaines informations sont spécifiques au monde mathématique (c.à.d qu'on ne peut pas les résoudre par un simple algorithme). En effet, Alain turing était le premier à montrer qu'une suite peut être mathématiquement parfaitement définie et pourtant non calculable par programme. Pour cela, il considère le problème d'arrêt d'un programme (ou pb d'arrêt d'une machine de Turing, ce qui revient au même).

Le problème de l'arrêt s'énonce de la manière suivante :

« Déterminer si un algorithme s'arrête ».

Problème de l'arrêt :

Le problème est ici de déterminer s'il existe une méthode systématique pour prouver qu'un algorithme se termine ou non.

Démonstration par l'absurde :

Le problème de l'arrêt est indécidable. Cela se démontre par l'absurde. On suppose que ce problème est décidable. On montre ensuite qu'il y a une incohérence.

On suppose qu'il existe une fonction arrêt(A) qui à tout algorithme A associe Vrai si l'algorithme A se termine et Faux sinon.

Pour tout entier naturel n, on définit un algorithme Turing(n) de la manière suivante :

- Lister les algorithmes qui s'écrivent avec au plus n caractères, qui prennent en entrée un entier et qui retournent un entier.
- Avec la fonction arrêt, sélectionner les algorithmes qui se terminent si l'entrée est n.
- Déterminer les nombres que chacun de ces algorithmes renvoie.
- Renvoyer le plus grand de ces nombres + 1.

On étudie le cas où k est le nombre de caractères de l'algorithme Turing(n). Si on teste Turing(k), on a alors :

- Turing fait lui même partie de la liste des algorithmes qui s'écrivent avec au plus k caractères prenant en entrée un entier et retournant un entier.
- Turing est sélectionné car cet algorithme s'arrête.
- Si Turing(k) renvoie m_0, alors m_0 fait partie des nombres que les algorithmes sélectionnés renvoient.
- Il y a deux possibilités :
- Si m_0 est le plus grand des nombres, alors Turing(k) renvoie $m_0 + 1$. C'est absurde car Turing(k) renvoie m_0.
- Si le plus grand des nombres est $p_0 > m_0$, alors Turing(k) renvoie $p_0 + 1 > m_0 + 1$. C'est absurde car Turing(k) renvoie m_0.

Ainsi la supposition initiale est fausse.

Donc le problème de l'arrêt est indécidable.

2.8 Exercices

1) Soit $E \subset \mathbb{N}$ un ensemble récursivement énumérable, énuméré par une fonction calculable f strictement croissante. Montrer que E est décidable.

2) En déduire que tout sous-ensemble récursivement énumérable infini de $\mathbb{N}$ contient un ensemble décidable infini.

3) Montrez que les fonctions suivantes sont primitives récursives :
a) la fonction factorielle(n)
b) la fonction pgcd(m, n)
c) le prédicat premier(n) qui vaut 1 si n est un nombre premier, 0 sinon.

Chapitre 3

Les systèmes formels

Sommaire

3.1 Théories axiomatiques

Une théorie axiomatique est un système logique qui représente une théorie mathématique, c'est-à-dire un ensemble de résultats se rapportant tous à un même type d'objet. Une théorie axiomatique est fondée sur un ensemble d'axiomes qui sont des formules définissant les objets et relations de base de la théorie ; à partir de ces axiomes et en utilisant les règles de raisonnement on dérive les théorèmes de la théorie. Par exemple, la théorie des ensembles est un système formel dont les axiomes définissent la notion d'ensemble. Un axiome est donc une proposition non démontrée qui sert de point de départ à un raisonnement : par exemple « par deux points il passe une et une seule droite » est un axiome de la géométrie euclidienne.

La vérité des axiomes ou des formules est définie relativement à un modèle, un univers possible, dans lequel les formules sont interprétées.

3.2 Problématiques

Un des paradoxes les plus célèbres est sans doute celui de Russell. Russell considère l'ensemble E de tous les ensembles qui ne se contiennent pas eux-mêmes. La question que pose alors Russell est la suivante : E se contient-il lui-même ? Examinons successivement les deux possibilités :

Si E se contient lui même, alors par définition il ne se contient pas lui-même, ce qui est contradictoire. Si E ne se contient pas lui-même, alors par définition il doit appartenir à E, ce qui est une fois de plus contradictoire. Ce type de paradoxes découle du flou laissé par Cantor autour de sa définition d'ensemble. Il fut nécessaire de reformaliser sa théorie, en fait de construire une axiomatique formelle des ensembles abstraits. Zermelo, dès 1908, fut le premier à proposer de construire un tel système pour réduire la notion intuitive d'ensemble à une notion formelle. Une excellente axiomatique formelle des ensembles fut fournie par Gödel en 1940. Cette classe de paradoxes est souvent désignée sous le nom de paradoxes logiques, dans la mesure où ils sont liés à la logique de la théorie en cause.

Les paradoxes sémantiques forment une toute autre classe de paradoxes. Un des plus célèbres est le paradoxe de Berry. On considère le nombre naturel défini par « le plus petit nombre entier ne pouvant être exprimé en moins de quinze mots ». On sait qu'il existe un ensemble

des nombres naturels qu'on ne peut définir en moins de quinze mots. En raison des propriétés des sous-ensembles de $\mathbb{N}$, cet ensemble admet une borne inférieure I. Il semble même que ce nombre I soit 1297297 (un million deux cent quatre-vingt-dix-sept mille deux cent quatre-vingt-dix-sept, soit exactement quinze mots). Mais ce nombre I est aussi défini par l'expression « le plus petit nombre entier ne pouvant être exprimé en moins de quinze mots ». Or cette expression comporte quatorze mots. Il y a donc contradiction. Ce type de paradoxe est d'origine sémantique, c'est-à-dire lié au langage dans lequel nous exprimons la définition.

La théorie des types de Russell a précisément été créée dans le but de formaliser correctement ce type de problèmes. Le but initial de Russell est la réduction des mathématiques à la théorie logique. Les Principia Mathematica publiés avec Whitehead de 1910 à 1913 constituent le travail qui fondera la logique moderne.

L'ensemble des problèmes rencontrés par les mathématiques que nous venons de décrire, semblait montrer qu'il était temps de renoncer aux mathématiques « intuitives » pour en venir à des constructions formelles pures, bien que la tentative logiciste formelle de Russell ait déjà montré quelques faiblesses. C'est dans ces conditions qu'éclata la controverse entre formalistes et intuitionnistes.

3.3 Introduction

Un système formel est une modélisation mathématique d'un langage en général spécialisé. Les éléments linguistiques, mots, phrases, discours, etc., sont représentés par des objets finis (entiers, suites, arbres ou graphes finis...). Le propre d'un système formel est que la correction au sens grammatical de ses éléments est vérifiable algorithmiquement, c'est-à-dire que ceux-ci forment un ensemble récursif.

Les systèmes formels s'opposent aux langues naturelles pour lesquels les algorithmes de traitement sont extrêmement complexes et surtout doivent évoluer dans le temps pour s'adapter aux transformations du langage.

Les systèmes formels sont apparus en logique mathématique afin de représenter mathématiquement le langage et le raisonnement mathématique, mais peuvent se trouver également dans d'autres contextes : informatique, chimie...

La réalisation d'un système formel fournit un cadre de représentation d'une réalité donnée ou des règles automatiques produisant tous les éléments vrais. Le but est d'avoir un petits nombre de vérités initiales (ce sont les axiomes) et de disposer de mécanismes de raisonnement (ce sont des règles de production) pour révéler des vérités cachées.

Exemples de systèmes formels :

- *un système expert*, est constitué d'une base de faits et de règles permettent d'inférer de nouvelles connaissances, constitue un système formel.
- *Le calcul des prédicats*, système logique modélisant le langage mathématique.
- *Le lambda-calcul*, langage de programmation théorique utilisé pour étudier les liens entre logique et informatique ; plus généralement tout langage de programmation est par définition un système formel.
- *La théorie des ensembles*, système formel dont les axiomes définissent la notion d'ensemble.
- *La nomenclature des molécules organiques* est un système formel.

3.4 Définition et composants

Pour construire une langue naturelle (par exemple le français) , on a besoin des quatres points suivants :

- un alphabet (a, b, ..., z, blanc, virgule, parenthèse ouvrante, ...),
- un procédé de formation des mots, qui est la concaténation,
- un dictionnaire, qui permet de savoir que "chat" est anglais, alors que "cat" ne l'est pas,
- des règles de grammaire, qui permettent de savoir que "chattes" est français, alors qu'il n'est pas dans le dictionnaire.

Pour construire un système formel, nous aurons besoin de quatre choses analogues :

- un alphabet, ensemble de symboles pas nécessairement réduit à des caratères,
- un procédé de formation des expressions, pas nécessairement la concaténation,
- un ensemble d'axiomes, c'est-à-dire d'expressions obéissant aux deux premiers points ci-dessus, et dont on décide arbitrairement qu'ils appartiennent au système,
- des règles de dérivation qui, à partir des axiomes, permettent de produire des théorèmes (c'est-à-dire des expressions appartenant au système), et peuvent ensuite s'appliquer aux théorèmes pour en produire d'autres.

Langage, Axiomes, Règles Inférences

Un système formel S est compose d'un quadruplet S = (A, γ, Ax, R) où :

— A est le langage, il est composé d'un vocabulaire ou alphabet fini de symboles,

— γ est un procédé de construction de mots (formules bien formées fbf),

— Ax est un ensemble fini ou dénombrable de mots particuliers appelés axiomes,

— R est un ensemble fini de règles d'inférences, dites encore règles de déduction ou de dérivation.

Remarques : Il y a deux types de règles d'inférence :
- les règles de production, qui produisent de nouveaux mots à partir des mots initiaux, et
- les règles de réécriture, qui fournissent une nouvelle forme pour le même mot.

3.5 Systèmes de déduction

Un système de déduction (ou système déductif, ou système d'inférence, ou encore un calcul) comprend un ensemble de formules prises comme point de départ (des axiomes), et un ensemble de règles (qu'on appelle règles d'inférence) qui permettent de déduire (ou prouver) des formules, à partir d'un ensemble de formules bien formées (fbf). Une formule prouvable à partir de (fbf) s'appelle un théorème. Parmi les systèmes déductifs les plus classiques, on trouve la méthode des tableaux sémantiques, la résolution, les systèmes à la Hilbert, la déduction naturelle, et le calcul des séquents :

— La méthode des tableaux a été inventée en 1955 par le philosophe et logicien hollandais Evert Willhem Beth (1908-1964), et la résolution est en grande partie due au mathématicien et informaticien John Alan Robinson (1930-2016). Ces deux systèmes sont parfois appelés systèmes de réfutation (plutôt que de déduction), car leur principe général consiste à prouver une formule F en justifiant que la formule $\neg F$ (la réfutation de F) conduit à une contradiction.

— Les systèmes à la Hilbert (il y a de nombreuses variantes) sont des systèmes de déduction qui suivent un modèle proposé par le mathématicien allemand David Hilbert (1862-1943).

— La déduction naturelle est un système proposé par le mathématicien et logicien allemand Gerhard Gentzen (1909-1945) en 1934, proche de la manière de raisonner des mathématiciens.

— Le calcul des séquents a été créé par Gerhard Gentzen à partir de son système de déduc-

tion naturelle, pour des raisons techniques en rapport avec la métathéorie des systèmes déductifs (son calcul des séquents lui permettait de démontrer des propriétés intéressantes du système déductif, qu'il ne pouvait justifier avec le système de déduction naturelle).

Les systèmes de déduction précédents sont tous équivalents, dans la mesure où ils permettent de démontrer exactement les mêmes théorèmes (de logique classique).

Les systèmes à la Hilbert sont des systèmes simples, permettant aussi de justifier rapidement certaines propriétés de la logique. Leur principal défaut est le fait de ne pas pouvoir poser d'hypothèse, ce qui peut alourdir les démonstrations. La déduction naturelle est le système formel le plus proche du raisonnement naturel des mathématiciens. Il corrige certains défauts des systèmes à la Hilbert, en permettant en particulier de poser des hypothèses. Les autres systèmes sont principalement utilisés en logique pure ou en informatique (en particulier pour des techniques de démonstration automatique).

3.6 Exemples de systèmes formels

Quelques exemples pouvant servir d'introduction aux systèmes formels :

3.6.1 Système (G.P.)

Le système (G.P.), encore appelé "axiomatique de Peano" : - alphabet : x, y, * - mots : S1 * S2 où S1 et S2 sont des suites quelconques de symboles x ou y - un unique axiome : x * x - une seule règle, qui est une règle de production : w1 * w2 y w1 * y w2 où w1, w2 sont des mots.

3.6.2 Système (D.H.)

Le système (D.H.) dû à Douglas Hoffstadter : - alphabet : M, I, U - mots : toute suite de lettres de l'alphabet - axiome : M I - règles de déduction : 1- w I w I U où w est un mot (production) 2-Mw M ww où w est un mot (production) 3- III U (réécriture) 4- UU (réécriture)

3.6.3 Système (p q -)

Le système (p q -) : - alphabet : p, q, - - mots : suites finies de symboles de l'alphabet - axiomes : les mots x p - q x - où x est formé uniquement de - - une seule règle de déduction (production) : si x, y et z sont des mots formés uniquement de -, du mot x p y q z on peut déduire le mot x p y - q z -.

3.7 Notion de démonstration

Dans un système formel S, une preuve (ou une démonstration) est une suite finie de mots (ou fbf) w1 w2...wk où chaque wi ($1 \leq i \leq n$) est :
• soit un axiome de S,
• soit se déduit de wj, j<i par application d'une Règle d'inférence Ri.
l'entier k est appelé la longueur de la preuve.

3.8 Notion de théorème

Une formule d'un système formel est dite un théorème si cette formule est :
- Soit un axiome,
- Soit la formule est obtenue par application d'une règle d'inférence à un théorème.

Un théorème est un mot t tel qu'il existe une démonstration w1 w2 ... wk avec $t = wk$. On note alors : $\vdash t$. Un axiome est un théorème, sa démonstration est de longueur 1. Nous avons donc :

$$\{axiomes\} \subset \{theoremes\} \subset \{mots\} \subset \{fbf\}$$

3.8.1 Exemples

Système "peu"

Considérons le système "peu" :

- alphabet = l'ensemble des trois symboles "p" , "e" , et "u",
- p.f.e. = concaténation,
- axiome = upueuu,
- règles :

— R1 : si une expression de la forme AeB est un théorème (où "A" désigne n'importe quelle suite de "u", de "p", et de "e", et B de même), alors l'expression uAeBu est aussi un théorème.

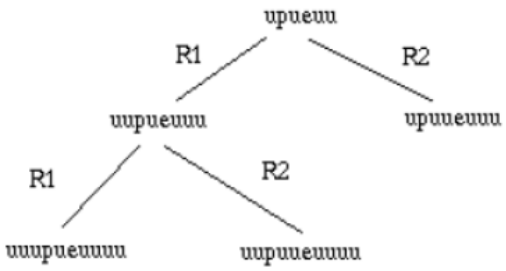
— R2 : si une expression de la forme AeB est un théorème, alors l'expression AueuB est aussi un théorème.

Questions :

1. Q1 = uupuueuuuu est-il un théorème ?
2. Q2 = upuueuuuu ?
3. Q3 = upupueuuu ?

Réponses : Pour démontrer qu'une expression est un théorème, nous avons une procédure (développer l'arbre) infaillible, qui répond en 2n/2, n étant la longueur de la chaîne, donc en temps fini.

1. Q1, oui. Preuve ? On développe l'arbre :



2. Q2, non : il y a un nombre impair de "u", ce qui n'est pas possible.
3. Q3, non : il y a deux "p".

Système "BAB"

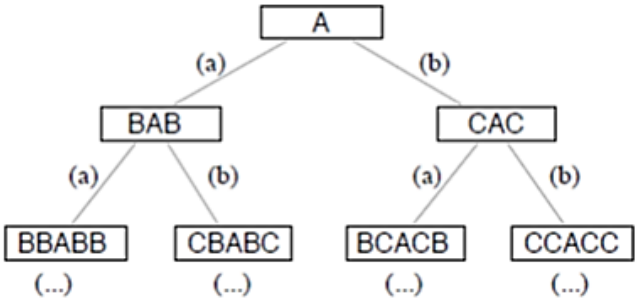
Soit le système « BAB » :

1. alphabet = les trois symboles A, B et C,
2. mots = toute suite finie de symboles de l'alphabet. Par exemple ""(chaîne vide), "A", "AAAAB", "BABBCA", etc ...
3. axiome unique : "A",
4. règles de déduction : soient n et m deux mots quelconques :

— a) mAn ==> BmAnB

— b) mAn ==> CmAnC

Ceci peut se traduire par un graphe, aussi appelé arbre de dérivation :



Remarque : pour démontrer un théorème il n'est pas nécessaire de savoir ce qu'il veut dire.

3.9 Déduction naturelle

En logique mathématique, la déduction naturelle est un système formel où les règles de déduction des démonstrations sont proches des façons naturelles de raisonner. C'est une étape importante de l'histoire de la théorie de la démonstration pour plusieurs raisons :

— contrairement aux systèmes à la Hilbert fondés sur des listes d'axiomes logiques plus ou moins ad hoc, la déduction naturelle repose sur un principe systématique de symétrie : pour chaque connecteur, on donne une paire de règles duales (introduction/élimination) ;

— elle a conduit Gentzen à inventer un autre formalisme très important en théorie de la démonstration, encore plus « symétrique » : le calcul des séquents ;

— elle a permis dans les années 1960 d'identifier la première instance1 de l'isomorphisme de Curry-Howard.

La terminologie « déduction naturelle » a été suggérée, par Gentzen, eu égard à l'aspect peu intuitif des systèmes à la Hilbert.

La déduction naturelle, dans sa forme actuelle, est un système formel proposé par Gerhard Gentzen en 1934.

De nombreux logiciens, à commencer par Gottlob Frege et David Hilbert, mais également

Bertrand Russell et Alfred North Whitehead avec leurs Principia Mathematica, ont développé la logique sous une forme axiomatique inspirée par la méthode euclidienne : les lois logiques sont déduites à partir d'axiomes en utilisant essentiellement la règle de modus ponens. Cette méthode simple a révélé des difficultés liées au fait que les raisonnements sous hypothèses, pratique pourtant courante en mathématiques, ne sont pas directement représentables.

Gerhard Gentzen est le premier à avoir développé des formalismes qui, en abandonnant partiellement la méthode euclidienne, redonnent à la logique le caractère d'un cheminement naturel, c'est-à-dire se rapprochant mieux de la pratique mathématique. La principale idée de Gentzen était simple : remplacer les axiomes logiques nécessaires, mais peu naturels, des systèmes à la Hilbert4 par des règles de déduction comme l'introduction de la flèche qui code formellement le fait de « poser une hypothèse » dans le cours d'un raisonnement. Ce faisant, Gentzen a développé pour la première fois une présentation très symétrique de la logique dans laquelle chaque connecteur est défini par une paire de règles duales : les introductions et les éliminations. Il a également développé un formalisme dans lequel les déductions ne sont pas des suites de phrases, mais des arbres (ou plus précisément des graphes orientés acycliques). Cette méthode, très suggestive pour l'intuition, a conduit Gentzen à faire de belles découvertes5, mais elle nuit à l'idée originale qui était de reproduire les formes naturelles de raisonnement. Fitch a modifié la méthode de Gentzen en introduisant une notation fondée sur l'indentation pour représenter astucieusement la structure arborescente des déductions. Cette représentation à la fois formelle et plus naturelle n'est toutefois pas la mieux adaptée pour obtenir les résultats principaux de la déduction naturelle comme la normalisation. Pour cette raison, on s'en tiendra dans cet article à une présentation à la Gentzen, le lecteur intéressé pourra se reporter à l'article dédié au style de Fitch.

Dans les années 1960, Dag Prawitz a poursuivi l'étude de la déduction naturelle et y a démontré un théorème d'élimination des coupures.

3.9.1 Notion de règle

La déduction naturelle est fondée sur des règles d'inférence qui permettent de déduire des théorèmes à partir d'autres. Par exemple la règle suivante, le modus ponens mais que l'on appelle élimination de la flèche dans ce contexte :

$$\frac{p \qquad p \to q}{q}$$

permet de déduire q à partir d'une démonstration de p et d'une démonstration de $p \Rightarrow q$. Les formules au-dessus de la barre s'appellent les prémisses et la formule sous la barre s'appelle la conclusion. Dans l'exemple, p et $p \Rightarrow q$ sont les prémisses et q est la conclusion.

3.9.2 Notion d'arbre de preuve

Une démonstration en déduction naturelle est un arbre de preuve. Voici un arbre de preuve déduisant l'énoncé le sol est glissant des trois hypothèses : (1) il pleut, (2) s'il pleut alors le sol est mouillé et (3) si le sol est mouillé alors le sol est glissant :

$$\dfrac{\dfrac{\text{il pleut} \qquad \text{il pleut} \to \text{sol mouillé}}{\text{sol mouillé}} \qquad \text{sol mouillé} \to \text{sol glissant}}{\text{sol glissant}}$$

La règle d'élimination de la flèche y est appliquée deux fois.

3.9.3 Notion d'hypothèse

La déduction naturelle fait des raisonnements sous des hypothèses. Une démonstration de q sous l'hypothèse p a la forme :

$$\begin{array}{c} p \\ \vdots \\ \hline q \end{array}$$

où les points de suspension verticaux représentent un arbre de preuve de conclusion q et dont certaines feuilles comportent l'hypothèse p.

3.9.4 Notion d'hypothèse déchargée

Il existe aussi des règles où une hypothèse est déchargée, c'est-à-dire que l'hypothèse n'est plus supposée à partir de l'application de la règle. Par exemple, à partir d'une démonstration de q sous l'hypothèse p, on peut construire une démonstration de l'implication $\quad q$. On note :

$$\frac{\begin{array}{c}[p]\\ \vdots\\ q\end{array}}{p \to q}$$

et les crochets dans $[p]$ signifient que l'hypothèse p est déchargée. Cette règle appelée introduction de l'implication est une internalisation du théorème de déduction des systèmes à la Hilbert.

3.10 Systèmes formels à la Hilbert

L'école formaliste emmenée par le mathématicien David Hilbert choisit une voie totalement différente de la voie intuitionniste. Constatant que les paradoxes venaient de systèmes d'axiomes insuffisamment formalisés, Hilbert proposa de rédiger une théorie purement formelle des mathématiques et de démontrer sa consistance formelle.

En logique, les systèmes à la Hilbert servent à définir les déductions formelles en suivant un modèle proposé par David Hilbert au début du xxe siècle : un grand nombre d'axiomes logiques exprimant les principales propriétés de la logique que l'on combine au moyen de quelques règles, notamment la règle de modus ponens, pour dériver de nouveaux théorèmes. Les systèmes à la Hilbert héritent du système défini par Gottlob Frege et constituent les premiers systèmes déductifs, avant l'apparition de la déduction naturelle ou du calcul des séquents, appelés parfois par opposition systèmes à la Gentzen. L'idée originelle de Hilbert était de construire un système formel des mathématiques, ou tout au moins de l'analyse. Mais le problème se révéla tellement complexe que lui et ses élèves se limitèrent à construire un système formel de l'arithmétique.

Le système formel H (pour Hilbert) prend un langage propositionnel où figurent seulement les deux connecteurs : $\neg$ et $\Rightarrow$. La seule règle d'inférence est le Modus Ponens, et les axiomes de H sont les schéma d'axiomes suivants :

a1. $\phi \to (\psi \to \phi)$
a2. $(\phi \to (\psi \to \chi)) \to ((\phi \to \psi) \to (\phi \to \chi))$
a3. $(\neg\phi \to \neg\psi) \to (\psi \to \phi)$

Le système système formel "à la Hilbert" est composé de plusieurs axiomes et d'un nombre très réduit de règles d'inférences. Comme conséquence, les systèmes à la Hilbert peuvent être caractérisés comme synthétiques : pour trouver une preuve pour un énoncé, on part d'un ensemble de vérités générales (les axiomes) et on voit si, et comment, un énoncé particulier peut être dérivé de cet ensemble. En pratique, pour tester si une formule est dérivable ou est un théorème, il faut partir des axiomes et "deviner" quelle est la bonne instance qui va nous permettre de dériver ou prouver la formule. C'est donc très difficile de construire une dérivation ou une preuve pour une formule en utilisant ce type de système formel.

3.11 Interprétation d'un système formel

Une interprétation est l'attribution de certaines valeurs de vérité ou non vérité aux symboles et aux formules d'un système formel. Un même système formel peut recevoir plusieurs interprétations différentes. On retiendra comme correctes, les interprétations qui associent à tous les théorèmes la seule valeur vraie.

• On appelle tautologie tout mot d'un système formel qui se traduit par un énoncé vrai dans toutes les interprétations du système formel.

• Un système est dit consistant lorsqu'il existe une interprétation transformant tout théorème en énoncé vrai.

Exemple :

Considérons le système formel suivant :

alphabet = l'ensemble des trois symboles "plus ", "egal ", "un " p.f.e. = concaténation axiome = un plus un egal un un règles :

— R1 : si une expression de la forme A egal B est un théorème (où "A" désigne n'importe quelle suite de "un ", de "plus ", et de "egal ", et B de même), alors l'expression un A egal B un est aussi un théorème.

— R2 : si une expression de la forme A egal B est un théorème, alors l'expression A un egal un B est aussi un théorème.

Questions :

1. Q1 = un un plus un un egal un un un un est-il un théorème ?
2. Q2 = un plus un un egal un un un un ?

3. Q3 = un plus un plus un egal un un un ?

Réponses :

1. oui
2. non
3. non

Remarque : les deux systèmes sont isomorphes.

Interprétation :

Une interprétation est appelée modèle.

Si nous remplaçons "plus " par "+", "egal " par "=", et "un " par "1", l'axiome s'écrit : 1+1=2 , en base 1 (les chiffres romains, concaténation = addition).

R1 se lit : si A=B, alors 1+A=B+1

R2 : si A=B, alors A+1=1+B

Q1 : 2+2=4 est un théorème

Q2 : 1+2=4 est un non-théorème

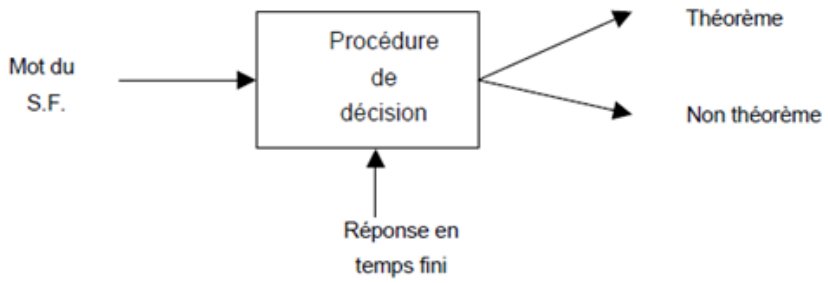
Q3 : 1+1+1 =3 est-un non-théorème !

Ce que nous venons de manipuler n'est pas un système formel : le symbole "2" ne fait pas partie de l'alphabet. Ce que nous venons de manipuler est l'interprétation du système formel. "1+1+1=3", dans l'interprétation, est vrai, bien que ce soit un non-théorème.

3.12 Propriétés des systèmes formels

3.12.1 Décidabilité

On dira qu'un système formel est décidable lorsqu'il existe une procédure de décision unique qui permet en un temps fini de décider si un mot quelconque du système est un théorème ou un non-théorème (mot dont on peut prouver qu'il n'est pas un théorème).



Un système formel qui n'est pas décidable peut être :

— semi-décidable : lorsqu'il existe une procédure qui, si une formule est démontrable, le dira en un temps fini, mais qu'il n'existe pas de procédure capable de faire la même chose pour tous les non théorèmes.

— indécidable : dans les autres cas.

3.12.2 Cohérence

Un " bon " système formel S doit être cohérent. S est cohérent s'il ne permet pas de démontrer une contradiction (formellement : s'il n'existe pas de proposition A telle que S démontre A et non A)

S'il n'est pas cohérent et le principe du raisonnement par l'absurde est accepté parmi les règles de déduction logique alors toutes les formules sont prouvables. Un système qui prouve tout et son contraire ne prouve rien du tout. Une théorie axiomatique est cohérente s'il existe des formules qui ne sont pas conséquence de ses axiomes. Par exemple l'arithmétique de Peano est cohérente car elle ne démontre pas la formule « 0 = 1 ». On peut équivalemment définir la cohérence comme le fait de ne pas démontrer de contradiction. Kurt Gödel a montré que toute théorie axiomatique suffisamment puissante pour représenter l'arithmétique ne pouvait démontrer sa propre cohérence de manière formelle.

Kurt Gödel a montré au début du siècle que tout système formel un tant soit peu complexe contenait des propositions vraies mais non démontrables (voir Théorème d'incomplétude de Gödel).

3.12.3 Cohérence sémantique

Un système S est sémantiquement cohérent si et seulement si toute formule dérivable du système est une formule valide (tautologie).

3.12.4 Quelques autres propriétés des systèmes formels

• On appelle tautologie tout mot d'un système formel qui se traduit par un énoncé vrai dans toutes les interprétations du système formel.
• Un système formel dans lequel tout théorème est une tautologie est dit correct.
• Un système formel dans lequel toute tautologie est un théorème est dit complet.
• Un système est dit consistant lorsqu'il existe une interprétation transformant tout théorème en énoncé vrai. Cette interprétation est appelée modèle.

Remarque : un système correct est consistant.

3.13 Systèmes formels en logique

Les systèmes formels ont été conçus par les logiciens afin de poser et étudier mathématiquement certains problèmes liés au langage mathématique. De ce point de vue on peut les considérer comme des métathéories générales, des théories sur les théories (mathématiques).

Les systèmes logiques visant à modéliser le langage mathématique résolvent trois problèmes :
- Comment formalise-t-on les énoncés mathématiques (théorèmes, lemmes, définitions, etc.), c'est-à-dire comment définit-on la notion de formule ?
- Réciproquement, comment interprète-t-on les objets formels que sont les formules de façon à les voir comme des énoncés mathématiques signifiants ?
- Comment prouve-t-on des formules, c'est-à-dire comment formalise-t-on les règles du raisonnement mathématique ?

Les systèmes formels ont permis l'émergence d'une épistémologie des mathématiques appelée point de vue formaliste au terme de laquelle les mathématiques apparaissent comme un jeu de manipulation de symboles suivant des règles rigoureuses mais a priori dépourvues de sens ; le sens des formules est reconstruit a posteriori par les interactions qu'elles entretiennent les unes avec les autres au travers des règles de raisonnement.

3.14 Les systèmes formels en informatique

Les langages de programmation sont des systèmes formels : comme les systèmes logiques servent à formaliser le langage mathématique (théorème, démonstrations), les langages de programmation formalisent les algorithmes. Les langages de programmation sont formés d'au moins deux constituants :

• les programmes qui sont représentés par des fichiers informatiques, c'est-à-dire des suites finies de nombres ; pour l'affichage les nombres sont en général interprétés comme des codes de caractères de façon à présenter le programme comme un texte. Mais ça n'est pas toujours le cas : un programme en langage machine est une suite de codes d'instructions du processeur. Bien entendu toute suite de nombres n'est pas un programme correct ; par contre on peut tester algorithmiquement au moyen d'une analyse syntaxique si un fichier contient un programme qui satisfait les contraintes syntaxiques du langage dans lequel il est écrit.

• une sémantique opérationnelle qui détermine la façon dont un programme doit s'exécuter, c'est-à-dire la façon dont l'ordinateur doit interpréter le programme comme une séquence d'instructions. La sémantique opérationnelle d'un langage est en général réalisée par un programme (qui parfois est même écrit dans le même langage !) appelé compilateur ou interpréteur selon son mode de fonctionnement ; dans le cas du langage machine la sémantique opérationnelle est réalisée par le processeur lui-même.

Auxquels on ajoute parfois un troisième constituant : • une sémantique dénotationnelle qui détermine abstraitement le comportement du programme, typiquement en l'interprétant comme une fonction : par exemple un programme de tri sera interprété par une fonction sur les listes d'entiers et à valeurs dans les listes d'entiers.

En dehors des langages de programmation, l'informatique définit plusieurs autres types de systèmes formels : les langages de spécification qui servent à définir le comportement désiré d'un programme de façon à pouvoir ensuite tester, vérifier ou démontrer qu'une implantation donnée satisfait la spécification ; les protocoles de communication qui définissent rigoureusement les échanges d'informations entre ordinateurs (ou téléphones mobiles) sur un réseau, ...

3.15 Exercices

Exercice 1 :

Soit le système formel S défini par :

- L'alphabet V={a,b,c}

- Les formules bien formées sont constituées de suites de lettres de V ne contenant qu'une seule fois la lettre b et une seule fois la lettre c (b doit se trouver avant c dans la suite, pas nécessairement immédiatement avant),

- L'axiome bc,

- Les deux règles d'inférence R1 et R2 définies par :

R1 : $\zeta \Rightarrow a\zeta a$

R2 : $\zeta \Rightarrow \alpha b\ \alpha^{'}\ ac\ \alpha^{"} a$

avec ($\zeta = \alpha b\ \alpha^{'}\ c\ \alpha^{"}$ une formule de S où $\alpha\alpha^{'}\alpha^{"}$ sont des suites quelconques de symboles a éventuellement vides).

a. Quelle est l'incidence des règles s'inférence R1 et R2 sur la longueur des formules de S ? Que peut-on déduire sur la longueur de tout théorème de S ?

b. Donner quatre théorèmes de S.

c. Montrer que le théorème *abacaa* de S admet deux démonstrations distinctes. Que peut-on-dire du système formel S dans ce cas ?

d. Donner deux formules bien formées de S qui sont des non-théorèmes (des formules pour lesquelles on peut affirmer qu'il n'existe pas de démonstration). Justifier votre choix ?

Solution :

a. Les deux régles d'inférence ajoutent deux symboles a à la chaîne sur laquelle elles s'appliquent. Elles augmentent donc chacune la longueur de la chaîne de 2. Or, l'axiome est de longueur 2. Donc tout théorème obtenu à partir de l'axiome par n application des règles R1 et R2 sera de longueur $2 + 2 * n = 2 * (n + 1)$; c'est-à-dire de longueur paire supérieure ou égale à 2.

b. Donner quatre théorèmes de S : *aaaabcaaaa* , *aabacaaa* , *bc* , *baaacaaa*

c. *bc* $\vdash$ abca $\vdash$abacaa *et* bc $\vdash$ baca $\vdash$ abacaa .

On dira que S est un système formel ambigu.

d. Pour qu'une formule bien formée de S soit un non-théorème, il suffit qu'elle soit de longueur impaire. Par exemple : *aabcaaa* et *abcaa* sont des non-théorèmes.

Exercice 2 :

Une grammaire G = (V, S, P) sur A est très simple si elle vérifie les deux conditions suivantes :

(i) pour toute règle $x \Rightarrow \alpha$, on a $\alpha \in AV*$

(ii) pour toute lettre $a \in A$, il existe exactement une règle dont le membre droit commence par a

1. Quelles sont les grammaires très simples parmi les grammaires suivantes :

- $S \Rightarrow aSS|b$
- $S \Rightarrow aSbS|aSb|abS|ab$

2. Montrer qu'une grammaire très simple est inambiguë.

Chapitre 4

Le calcul propositionnel

Sommaire

4.1 Introduction

La logique sert à préciser ce qu'est un raisonnement correct, indépendamment du domaine d'application. Un raisonnement est un moyen d'obtenir une conclusion à partir d'hypothèses données. Un raisonnement correct ne dit rien sur la vérité des hypothèses, il dit seulement qu'à partir de la vérité des hypothèses, nous pouvons déduire la vérité de la conclusion. Nous commençons l'étude de la logique par les lois de la logique propositionnelle pour :
- modéliser les connaissances et les raisonnements.
- établir des déductions qui permettent d'aboutir à des conclusions.

Une proposition est une phrase susceptible d'être vraie ou fausse. En français, on parle souvent de "phrase énonciative" ou « assertion ». Voici quelques exemples de propositions écrites en langage naturel, en formalisme mathématique ou dans un mélange des deux.

— Un plus deux égalent trois

— 1+1 = 3

— 2^3=8

— Je donne cours de logique propositionnelle le mardi

Considérons l'exemple :
si le feu est rouge ou orange et si je respecte le code de la route alors je ne passe pas.
le feu est rouge et je passe
je suis en infraction si et seulement si je ne respecte pas le code de la route.
donc je suis en infraction

Ce raisonnement et composé de trois énoncés (les hypothèses) et d'une conclusion. On souhaiterai savoir s'il existe une démonstration formelle de la conclusion à partir des hypothèses.

Dans la logique contemporaine on observe deux courants. L'un qui voit la logique comme un langage interprété, une théorie qui traite des aspects les plus abstraits de la réalité, comme par exemple, dans les travaux de Frege, de Russell, de Quine. L'autre voit la logique comme un calcul, comme par exemple dans les travaux de Boole, de Skolem, de Hintikka, cette conception permet de traiter les systèmes axiomatiques formalisés, non pas comme des formules vraies absolument, mais comme des collections de formules vraies relativement à un modèle, c'est à dire relativement à un choix de domaine pour les variables et d'interprétation pour les constantes.

4.2 Définition

Le calcul propositionnel est un système formel pour la représentation des propositions logiques et des raisonnements tel que :
- Si je suis adulte ou vieux et si je suis riche alors j'achète une voiture.
- Je suis adulte et je n'achète pas de voiture
- Je suis dans ennuie si et seulement si je n'achète pas de voiture
- Donc je suis dans l'ennuie

Ce raisonnement est composé de trois hypothèses et d'une conclusion, on souhaiterait savoir s'il existe une démonstration formelle de la conclusion à partir des hypothèses.

Dans la syntaxe du système formel du calcul propositionnel, les mots substituables appelés **variables propositionnelles** sont remplacés par des symboles (je suis adulte est remplacé par p), je suis dans l'ennuie (remplacé par q), ...) et sont en nombre illimité. Les mots non substituables (soulignés) appelés **connecteurs logiques** constituent les articulations du raisonnement et sont en nombre très limité et sont binaires.

4.3 Le langage de la logique propositionnelle

Le langage propositionnel est construit à partir :
d'un ensemble dénombrable de variables propositionnelles (symboles ou lettres), qui peut prendre l'une des deux valeurs suivantes : 0 (Faux) et 1 (Vrai) parfois notées ($\perp$) pour Faux et (T) pour Vrai,
des connecteurs logiques de base en nombre de cinq qui sont :
$\vee$: la disjonction (ou) ,
$\wedge$: la conjonction (et),
$\neg$: la negation (non)on l'appelle aussi connecteur unaire parcequ'il n'a qu'un seul opérande
$\Rightarrow$: l'implication (si . . . alors) et
$\Leftrightarrow$: l'equivalence (Si et seulement si).
Et de parenthèses (,)

4.4 Syntaxe et sémantique du calcul propositionnel

Le calcul propositionnel comporte deux aspects :

- Aspect syntaxique : qui concerne les formules par elles-même.
- Aspect sémantique : qui donne un sens à ces formules grâce à un formalisme mathématique.

4.4.1 Syntaxe du calcul propositionnel

A partir de propositions élémentaire ou atomiques, on peut construire grâce aux connecteurs, des propositions plus complexes appelées ***formules***.

On définit une formule bien formée (fbf) par :

- p, où p est une variable propositionnelle
- (A $\vee$B)
- (A $\wedge$B)
- ($\neg$A)
- (A $\Rightarrow$ B) : si A alors B
- (A $\Leftrightarrow$ B)

Où A et B sont des variables propositionnelles ou des formules bien formées.

Exemple 1 :

s'il pleut et que je ne possède pas de d'argent alors je prends mon parapluie
si je prends mon parapluie alors c'est l'hiver.

Ce problème peut être modélisé sous la forme suivante :

- variable propositionnelle b = il pleut
- variable propositionnelle s = je possède de l'argent
- variable propositionnelle f = je prends mon parapluie
- variable propositionnelle p = c'est l'hiver

Representation syntaxique :

La syntaxe du langage des propositions est donnée par l'ensemble de règles suivant :

- $((b \bigwedge (\neg s)) \Rightarrow\!\!\!\!\longmapsto f)$
- $(f \Rightarrow p)$

En l'absence de parenthèses complet, les priorités sont les suivantes : $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$.

4.4.2 Sémantique du calcul propositionnel

Notion d'interprétation

Pour calculer la valeur de vérité de propositions logiques, il est nécessaire d'attribuer une interprétation aux formules de ce système. Pour cela, on note par 1 la valeur vrai et 0 la valeur faux. Toute formule du calcul propositionnel a une valeur de vérité égale à l'interprétation qu'elle a dans ce système.

Valuation d'une formule propositionnelle :

On appelle valuation d'une proposition, toute attribution de valeur de vérité à chacun de ses atomes. La valeur de vérité d'une proposition du calcul propositionnel dépend de la valeur de vérité des atomes qui la composent. Ainsi, la valeur de vérité d'une proposition comportant n atomes (ou variables) admet 2^n interprétations qui sont donnée par une table appelé table de vérité.
Les tables de vérité des opérations élémentaires sont données comme suit :

X	¬X	X	Y	X ∧ Y	X ∨ Y	X → Y	X ↔ Y
1	0	0	0	0	0	1	1
0	1	0	1	0	1	1	0
		1	0	0	1	0	0
		1	1	1	1	1	1

Propositions logiques composées :

Soit p,q,r trois propositions logiques :
P :=$((\neg q) \bigwedge (\neg p \bigvee q)) \Rightarrow p$

Si p, q et r sont trois propositions logiques, alors P est aussi une proposition logique. La valeur de vérité de P dépend des valeurs de vérité de p, q et r. Comment exactement montre le tableau suivant (qui montre aussi des calculs de vérité intermédiaires) :

p	q	r	$\neg q$	$p \vee r$	$(\neg q) \wedge (p \vee r)$	$((\neg q) \wedge (p \vee r)) \rightarrow p$
V	V	V	F	V	F	V
V	V	F	F	V	F	V
V	F	V	V	V	V	V
V	F	F	V	V	V	V
F	V	V	F	V	F	V
F	V	F	F	F	F	V
F	F	V	V	V	V	F
F	F	F	V	F	F	V

En regardant la dernière colonne on se rend compte que P est faux si et seulement si p est faux et q est faux et r est vraie, ou en formule que :

$(\neg p) \bigwedge ((\neg q) \bigwedge r))$ est vrai.

Ou que P est vraie si et seulement si : $\neg((\neg p) \bigwedge ((\neg q) \bigwedge r))$ est vraie.

Posons : Q := $\neg((\neg p) \bigwedge ((\neg q) \bigwedge r))$ et calculons de nouveau ses valeurs de vérité :

Considérons aussi la proposition :

$R = p \vee (q \vee (\neg r)$.

On peut aussi calculer son tableau.

p	q	r	P	Q	R
V	V	V	V	V	V
V	V	F	V	V	V
V	F	V	V	V	V
V	F	F	V	V	V
F	V	V	V	V	V
F	V	F	V	V	V
F	F	V	F	F	F
F	F	F	V	V	V

Conclusion :

les propositions composées P,Q et R ont la même valeur de vérité,n'importe quelles les valeurs de p,q,r. On dit que P,Q et R sont des propositions composées (ou formules logiques) logiquement

équivalentes.

Calculons aussi les vérités de la proposition P⇔Q :

p	q	r	P	Q	$P \leftrightarrow Q$
V	V	V	V	V	V
V	V	F	V	V	V
V	F	V	V	V	V
V	F	F	V	V	V
F	V	V	V	V	V
F	V	F	V	V	V
F	F	V	F	F	V
F	F	F	V	V	V

On voit que la proposition logique composée P⇔Q est toujours vraie, pour tous les valeurs de vérité de p,q et r. On dit P⇔Q est une tautologie, notation P⇔Q.

Tautologie et contradiction

Définition 1 : Une proposition logique composée (ou une formule logique) qui est toujours fausse, quelles que soient les valeurs de vérité des propositions qui la composent, est appelée une contradiction. Notation : P ⇔ F.

Par exemple : p∧(¬p)⇔ F.

Définition 2 : Deux formules logiques P et Q sont appelées logiquement équivalentes si la proposition logique P ⇔ Q est une tautologie.

Notation : P ⇔ Q.

Par exemple : (p ⇔ q) ⇔ [(p ⇒ q)∧ (q ⇒ p)].

Définition 3 : Si P ⇒ Q est une tautologie, où P et Q sont deux formules logiques, on dit que P ⇒ Q est une règle d'inférence. Notation P ⇒ Q.

Par exemple : (p ∧ q) ⇒ q.

4.5 Quelques propriétés importantes sur l'interprétation sémantique des formules

- Deux propositions P et Q sont équivalentes sémantiquement, si et seulement si, pour toute valuation v, v(P)=v(Q).
- Une proposition P est satisfaisable (ou cohérente) si et seulement si il existe une valuation v, v(P)=1.
- Une proposition P est réfutable si et seulement si il existe une valuation v, v(P)=0.
- Une proposition P est une tautologie si et seulement si elle n'est pas réfutable : pour toute valuation v, v(P)=1. On la note 1. Exemple : P $\vee \neg$ P est une tautologie.
- Une proposition P est une contradiction (ou inconsistante) si et seulement si elle n'est pas satisfaisable : pour toute valuation v, v(P)=0. On la note 0. Exemple : P $\wedge \neg$P est une contradition.

Exemple :
- (Z $\wedge$ Y) (X $\vee$ Y) et X $\vee \neg$ X sont deux propositions équivalente sémantiquement et sont toutes deux des tautologies.
- X $\Rightarrow$ Y et $\neg$X $\vee$ Y sont sémantiquement équivalentes mais ne sont pas des tautologies.

Il suffit pour s'en convaincre de vérifier leurs tables de vérité.

4.6 Tautologie et méthodes de démonstration

Pour signifier qu'une formule F est une tautologie, on écrira :$\vdash F$

4.6.1 Principales propriétés des connecteurs : $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$

On utilisera les mêmes propriétés vues en algèbre de Boole. La démonstration de ces propriétés peut se faire par table de vérité et est laissée à vérifier.

1. Commutativité de $\wedge$ et $\vee$:

$\vdash P \wedge Q \leftrightarrow Q \wedge P$

$\vdash P \vee Q \leftrightarrow Q \vee P$

2. Associativité de $\wedge$ et $\vee$:

$\vdash P \wedge (Q \wedge R) \leftrightarrow \vdash (P \wedge Q) \wedge R$

$\vdash P \vee (Q \vee R) \leftrightarrow \vdash (P \vee Q) \vee R$

3. Double distributivité :

$\vdash P \wedge (Q \vee R) \leftrightarrow \vdash (P \wedge Q) \vee (P \wedge R)$

$\vdash P \vee (Q \wedge R) \leftrightarrow \vdash (P \vee Q) \wedge \vdash (P \vee R)$

4. Idempotence de $\wedge$ et $\vee$:

$\vdash P \wedge P \leftrightarrow P$

$\vdash P \vee P \leftrightarrow P$

5. Principe de double négation :

$\vdash \neg \neg P \leftrightarrow P$

6. Lois de Morgan

$\vdash \neg (P \wedge Q) \leftrightarrow \vdash \neg P \vee \neg Q$

$\vdash \neg (P \vee Q) \leftrightarrow \vdash \neg P \wedge \neg Q$

4.6.2 Equivalences tautologiques

Les tautologies suivantes valident des techniques de démonstration.

$\vdash (P \rightarrow Q) \leftrightarrow \neg P \vee Q$

$\vdash (P \rightarrow Q) \leftrightarrow (\neg Q \rightarrow \neg P)$ (permet de remplacer la preuve du théorème par sa contraposée)

$\vdash \neg (P \rightarrow Q) \leftrightarrow P \wedge \neg Q$

$\vdash (P \rightarrow Q \vee R) \leftrightarrow (P \wedge \neg Q \rightarrow R)$

$\vdash (P \vee Q \rightarrow R) \leftrightarrow (P \rightarrow R) \wedge (Q \rightarrow R)$

$\vdash (P \rightarrow Q \wedge R) \leftrightarrow (P \rightarrow Q) \wedge (P \rightarrow R)$

4.6.3 Implications tautologiques (ou règles d'inférences)

Quelques tautologies importantes en termes d'implication (première partie avant l'implication est appelée hypothèse, la deuxième après l'implication est appelée thèse).Elles sont appelées des ***règles d'inférences***.

1. $\vdash \neg P \rightarrow (P \rightarrow Q)$
2. $\vdash P \wedge (P \rightarrow Q) \rightarrow Q$ (c'est la règle de ***modus ponens*** : si P est vrai et $P \rightarrow Q$ est vrai alors Q est vrai)
3. $\vdash (P \rightarrow Q) \wedge \neg Q \rightarrow \neg P$ (cette règle d'inférence fournit une démonstration par l'absurde ou par refutation. Le sens intuitif est évident : s'il est vrai que la proposition P implique Q et que l'on sait que Q est fausse, alors P ne peut être que fausse. En pratique, la proposition Q est une anthologie, du genre $t \wedge \neg t$.)
4. $\vdash (P \rightarrow Q) \wedge (Q \rightarrow R) \rightarrow (P \rightarrow R)$ (justifie la transitivité de l'implication)
5. $\vdash (P \vee Q) \wedge (\neg P \vee R) \rightarrow Q \vee R$

4.7 Système formel et aspect axiomatique de la logique propositionnelle

Il convient maintenant de savoir qu'en calcul propositionnel qu'il existe deux manières d'établir qu'une proposition est une loi ou un théorème de la logique propositionnelle. Nous pouvons utiliser soit :

- Aspect sémantique : consiste à employer des procédures non axiomatisées tels que les tables de vérités.
- Aspect syntaxique : Recourir à des procédures axiomatiques et démonstratives tels que les déductions.

L'aspect axiomatique de la logique propositionnelle revient à définir un système formel dans lequel les déductions que l'on peut faire conduisent à des théorèmes (des lois). Ces théorèmes sont les et seulement les tautologies du langage des propositions avec l'interprétation donnée par les tables de vérité.

4.7.1 Système formel pour le calcul propositionnel

Pour construire un système formel pour le calcul propositionnel, on a besoin de :

1. *Le langage propositionnel* :

Le langage propositionnel est construit à partir d'un ensemble dénombrable de variables propositionnelles (symboles ou lettres), qui peut prendre l'une des deux valeurs suivantes : 0 et 1 et des connecteurs logiques tels que : $\vee$, $\wedge$, $\neg$, $\Rightarrow$, $\Leftrightarrow$ et de parenthèses (,).

2. Les formules du calcul propositionnel. *(Les formules sont tout simplement des expressions logiques).*

3. Un sous ensemble de formules formant les axiomes.

4. Une ou plusieurs règles d'inférence.

Une preuve dans T est une séquence de formules $f1$, $f2$,.... fn telles que chaque fi, est :

- soit un axiome,
- soit un théorème déjà démontré,
- soit soit obtenue à partir de l'application d'une des règles d'inférence.

Si une formule f admet une preuve, on dit que f est démontrable ou f est un théorème et on note $\vdash f$.

4.7.2 Les axiomes

Il existe divers systèmes axiomatiques, chacun étant constitué d'un certain nombre d'axiomes. Le plus simple est le système de Lukasiewicz (1930), qui est basé sur les trois axiomes suivants n'utilisant que les connecteurs $\neg$ et $\Rightarrow$:

A1 : $P \to (Q \to P)$;
A2 : $(P \to (Q \to R)) \to ((P \to Q) \to (P \to R)$;
A3 : $(\neg P \to \neg Q) \to (Q \to P)$.

Avec P et Q des formules quelconques. Dans le cadre de la théorie de la démonstration, on revient au connecteur d'implication $\Rightarrow$ appelé aussi parfois le "conditionnel" (si ... alors ....)

Nous partons donc de quelques axiomes dont la vérité est posée (et non démontrée). Nous déterminerons des règles de déduction permettant de manipuler les axiomes ou toute expression obtenue à partir de ceux-ci. L'enchaînement de ces déductions est une démonstration qui conduit à un théorème, à une loi.

4.7.3 Les règles de déductions

Les règles de déduction ont pour objet la déduction de nouveaux théorèmes à partir de théorèmes connus. Une règle de déduction n'est pas un axiome mais en quelque sorte un procédé permettant d'obtenir de nouveaux théorèmes.

Afin de pouvoir effectuer des déductions, des règles d'inférences sont définies qui sont (il existe d'autres) :
- la substitution
- le modus ponens.

La règle de substitution

La règle de substitution permet de remplacer dans un théorème une variable propositionnelle, partout où elle figure, par une autre variable propositionnelle ou par une formule bien formée. (Les formules sont tout simplement des expressions logiques).

La règle du modus ponens

La règle du modus ponens (ou de détachement, ou la règle de conclusion, souvent appelée règle de dérivation aussi) est une règle primitive du raisonnement, elle est commune à toutes

les théories. On l'écrit formellement (suivant le contexte) :

de $\vdash \alpha$, $\alpha \Rightarrow \beta$, on déduit $\vdash \beta$

Cette régle peut être formule ainsi :

Soit deux formules α et β, telles α et $\alpha \Rightarrow \beta$ sont deux théorèmes, alors β est un théorème.

On peut utiliser le schéma suivant pour illustrer cette régle :

$$\frac{\vdash \alpha \qquad \vdash \alpha \Rightarrow \beta}{\vdash \beta}$$

Cette figure fait bien apparaître pourquoi la régle s'appelle *régle de détachement* : elle permet de détacher le conséquent β de l'antécédent α dans l'implication $\alpha \Rightarrow \beta$, pourvu que $\vdash \alpha$ et $\vdash \alpha \Rightarrow \beta$.

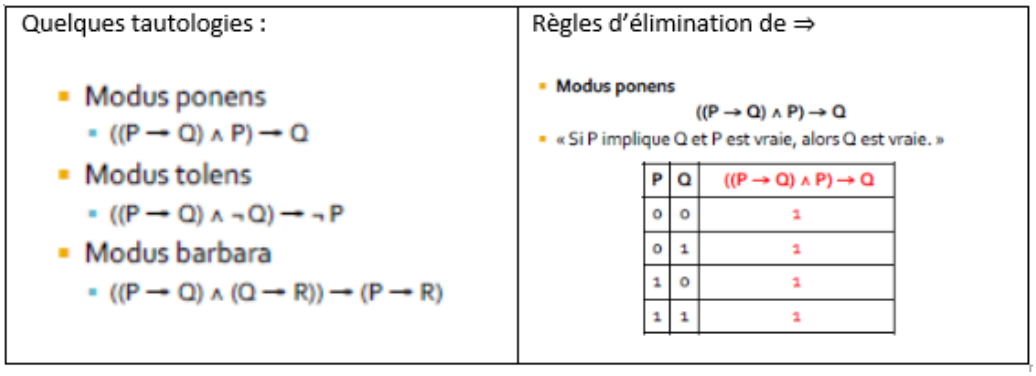


FIGURE 4.1 – Quelques Tautologies

4.7.4 Quelques équivalences logiques

Il est impossible de donner une liste complète des théorèmes du calcul des propositions car elle est infinie. Néanmoins, on peut donner les plus utilisés.

Ces théorèmes, combinés avec les règles de déduction ainsi que les règles dérivées, permettent de démontrer les autres théorèmes du calcul des propositions.

On peut donner la liste des théorèmes les plus connus :

$$
\begin{aligned}
p \wedge V &\Leftrightarrow p \ \textit{(Identité)} \\
p \vee F &\Leftrightarrow p \ \textit{(Identité)} \\
p \vee V &\Leftrightarrow V \ \textit{(Domination)} \\
p \wedge F &\Leftrightarrow F \ \textit{(Domination)} \\
p \vee p &\Leftrightarrow p \ \textit{(Idempotence)} \\
p \wedge p &\Leftrightarrow p \ \textit{(Idempotence)} \\
\neg(\neg p) &\Leftrightarrow p \ \textit{(Loi de la double négation)} \\
p \wedge q &\Leftrightarrow q \wedge p \ \textit{(Commutativité)} \\
p \vee q &\Leftrightarrow q \vee p \ \textit{(Commutativité)} \\
((p \vee q) \vee r) &\Leftrightarrow (p \vee (q \vee r)) \ \textit{(Associativité)} \\
((p \wedge q) \wedge r) &\Leftrightarrow (p \wedge (q \wedge r)) \ \textit{(Associativité)} \\
(p \vee (q \wedge r)) &\Leftrightarrow ((p \vee q) \wedge (p \vee r))) \ \textit{(Distributivité)} \\
(p \wedge (q \vee r)) &\Leftrightarrow ((p \wedge q) \vee (p \wedge r))) \ \textit{(Distributivité)} \\
\neg(p \wedge q) &\Leftrightarrow ((\neg p) \vee (\neg q)) \ \textit{(Loi de De Morgan)} \\
\neg(p \vee q) &\Leftrightarrow ((\neg p) \wedge (\neg q) \ \textit{(Loi de De Morgan)}
\end{aligned}
$$

FIGURE 4.2 – Equivalences logiques

Exemples de preuve par tableau de vérité

Si P := p∨(q∧r) et Q := (p∨q)∧(p∨r) on a :

p	q	r	$q \wedge r$	P	$p \vee q$	$p \vee r$	Q
V	V	V	V	V	V	V	V
V	V	F	F	V	V	V	V
V	F	V	F	V	V	V	V
V	F	F	F	V	V	V	V
F	V	V	V	V	V	V	V
F	V	F	F	F	V	F	F
F	F	V	F	F	F	V	F
F	F	F	F	F	F	F	F

FIGURE 4.3 – loi de la distributivité

D'aprés la table de vérité, on conclue que P ⇔ Q.

Montrons ¬(p∧q) ⇔ ((¬p)∨(¬q)) (Loi de De Morgan).

Soit P = ¬(p∧q) et Q = ((¬p)∨(¬q)). On a :

p	q	$p \wedge q$	P	$\neg p$	$\neg q$	Q
V	V	V	F	F	F	F
V	F	F	V	F	V	V
F	V	F	V	V	F	V
F	F	F	V	V	V	V

FIGURE 4.4 – Loi de De Morgan

D'aprés la table de vérité, on conclue que P ⇔ Q.

Preuve d'equivalences logiques par de l'algèbre :

Remarquons tout d'abord que si P,Q et R sont trois formules logiques (ou propositions composées) et P⇔Q et Q⇔R alors aussi P⇔R. (Et si P⇔Q alors aussi Q⇔P (c.-à.d., l'équivalence logique est symmétrique), et on a toujours P⇔P.) Donc on peut enchaîner des équivalences logiques pour obtenir d'autres. Nous avons déjà un certain nombre d'équivalence montrées, que nous pouvons utiliser. À la place d'utiliser un tableau pour vérifier une équivalence logique, on peut aussi utiliser les petites règles déjà montrées, comme en algèbre. Donnons un exemple. La proposition soi-même est sans importance.

Proposition 1.

Soit P := ((¬q)∧(p∨r)) ⇒ p

et Q := (p∨(q∨(¬r))).

On a P⇔Q.

Preuve. Nous allons utiliser une suite d'équivalences logiques (avec les raisons) :

$$
\begin{aligned}
P &\Leftrightarrow (\neg((\neg q) \wedge (p \vee r))) \vee p \text{ (Car } (p \to q) \Leftrightarrow ((\neg p) \vee q)) \\
&\Leftrightarrow (\neg(\neg q) \vee \neg(p \vee r))) \vee p \text{ (Selon De Morgan)} \\
&\Leftrightarrow (q \vee \neg(p \vee r))) \vee p \text{ (Double négation)} \\
&\Leftrightarrow (q \vee ((\neg p) \wedge (\neg r))) \vee p \text{ (Selon De Morgan)} \\
&\Leftrightarrow q \vee (p \vee ((\neg p) \wedge (\neg r))) \text{ (Assoc. et comm. pour } \vee) \\
&\Leftrightarrow q \vee ((p \vee (\neg p)) \wedge (p \vee (\neg r))) \text{ (Distrib.)} \\
&\Leftrightarrow q \vee (V \wedge (p \vee (\neg r))) \text{ (Selon } (p \vee (\neg p)) \Leftrightarrow V) \\
&\Leftrightarrow q \vee (p \vee (\neg r)) \text{ (Selon comm. et } (p \wedge V) \Leftrightarrow p) \\
&\Leftrightarrow Q \text{ (Assoc. et commut.)}
\end{aligned}
$$

FIGURE 4.5

Règles d'inference de base.

Les règles d'inference suivantes sont des équivalences logiques :

$$
\begin{aligned}
p &\Rightarrow (p \vee q) \textit{ (Addition)} \\
p \wedge q &\Rightarrow p \textit{ (Simplification)} \\
[p \wedge (p \to q)] &\Rightarrow p \textit{ (Modus ponens)} \\
[\neg q \wedge (p \to q)] &\Rightarrow q \wedge p \textit{ (Modus tollens)} \\
[(p \to q) \wedge (q \to r)] &\Rightarrow (p \to r) \textit{ (Syllogisme par hypothèse)} \\
[(p \vee q) \wedge \neg q] &\Rightarrow p \textit{ (Syllogisme disjonctif)}
\end{aligned}
$$

FIGURE 4.6 – Règles d'inférence

Preuve. Aussi facile à montrer avec un tableau de vérité. Par exemple Modus ponens :

$$
\begin{aligned}
p &\Rightarrow (p \vee q) \textit{ (Addition)} \\
p \wedge q &\Rightarrow p \textit{ (Simplification)} \\
[p \wedge (p \to q)] &\Rightarrow p \textit{ (Modus ponens)} \\
[\neg q \wedge (p \to q)] &\Rightarrow q \wedge p \textit{ (Modus tollens)} \\
[(p \to q) \wedge (q \to r)] &\Rightarrow (p \to r) \textit{ (Syllogisme par hypothèse)} \\
[(p \vee q) \wedge \neg q] &\Rightarrow p \textit{ (Syllogisme disjonctif)}
\end{aligned}
$$

FIGURE 4.7 – Modus ponens

La dernière colonne contient seulement des V , donc correspond à une tautologie.

4.8 Exercices

Exercice 1 : 1. Soit le système de Hilbert-Ackerman formé de 4 axiomes suivants :

Les axiomes d'H. A sont :

1. $(H.A)_1 \quad (x \vee x) \Rightarrow x$;
2. $(H.A)_2 \quad x \Rightarrow x \vee y$;
3. $(H.A)_3 \quad x \vee y \Rightarrow y \vee x$;
4. $(H.A)_4 \quad (x \Rightarrow y) \Rightarrow (z \vee x \Rightarrow z \vee y)$.

FIGURE 4.8 – système de Hilbert-Ackerman

Montrons le théorème suivant :$\vdash \neg x \vee x$

Solution de la preuve :

(1) $(x \vee x) \Rightarrow x$	$(H.A)_1$
(2) $x \Rightarrow x \vee y$	$(H.A)_2$
(3) $x \Rightarrow x \vee x$	sub $y := x$ dans (2)
(4) $(x \Rightarrow y) \Rightarrow (z \vee x \Rightarrow z \vee y)$	$(H.A)_4$
(5) $(x \vee x \Rightarrow y) \Rightarrow (z \vee (x \vee x) \Rightarrow z \vee y)$	sub $x := x \vee x$ dans (4)
(6) $(x \vee x \Rightarrow x) \Rightarrow (z \vee (x \vee x) \Rightarrow z \vee x)$	sub $y := x$ dans (5)
(7) $(z \vee (x \vee x) \Rightarrow z \vee x)$	Modus Ponens (1,6)
(8) $(\overline{z} \vee (x \vee x) \Rightarrow \overline{z} \vee x)$	sub $z := \overline{z}$ dans (7)
(9) $(z \Rightarrow (x \vee x)) \Rightarrow (z \Rightarrow x)$	abréviation
(10) $(x \Rightarrow (x \vee x)) \Rightarrow (x \Rightarrow x)$	sub $z := x$ dans (9)
(11) $x \Rightarrow x$	Modus Ponens (3,10)
(12) $\overline{x} \vee x$	Définition de $\Rightarrow$.

2. Soit α ,β ,γ des formules, alors :

de $\vdash \alpha \Rightarrow \beta$ et $\vdash \beta \Rightarrow \gamma$ on déduit $\vdash \alpha \Rightarrow \gamma$

Solution de la preuve :

(1) $\alpha \Rightarrow \beta$	Hypothèse
(2) $\beta \Rightarrow \gamma$	Hypothèse
(3) $(x \Rightarrow y) \Rightarrow (z \vee x \Rightarrow z \vee y)$	$(H.A)_4$
(4) $(\beta \Rightarrow \gamma) \Rightarrow (\overline{\alpha} \vee \beta \Rightarrow \overline{\alpha} \vee \gamma)$	sub $(x := \beta,\ y := \gamma,\ z := \overline{\alpha})$
(5) $(\beta \Rightarrow \gamma) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$	Abréviation
(6) $(\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)$	Modus Ponens $(5, 2)$
(7) $\alpha \Rightarrow \gamma$	Modus Ponens $(6, 1)$.

3. Soit α ,β ,γ des formules, alors : de $\vdash \alpha \Rightarrow \beta$ on déduit $\vdash \gamma \vee \alpha \Rightarrow \gamma \vee \beta$

Solution de la preuve :

(1) $(x \Rightarrow y) \Rightarrow (z \vee x \Rightarrow z \vee y)$	$(H.A)_4$
(2) $(\alpha \Rightarrow \beta) \Rightarrow (\gamma \vee \alpha \Rightarrow \gamma \vee \beta)$	sub $(x := \alpha,\ y := \beta,\ z := \gamma)$
(3) $\alpha \Rightarrow \beta$	Hypothèse
(4) $\gamma \vee \alpha \Rightarrow \gamma \vee \beta$	Modus Ponens $(2, 3)$.

Exercice 2 : Soit la liste d'axiomes suivante :

1. $\alpha \Rightarrow(\beta \Rightarrow \alpha)$
2. $(\neg\alpha\Rightarrow\neg\beta) \Rightarrow (\beta \Rightarrow\alpha)$
3. $(\alpha \Rightarrow(\beta\Rightarrow\gamma)) \Rightarrow ((\alpha\Rightarrow\beta)\Rightarrow(\alpha\Rightarrow\gamma))$
4. $\neg\neg\alpha\Rightarrow\alpha$
5. $\alpha\Rightarrow\alpha$
6. $(\beta \Rightarrow\alpha)\Rightarrow(\neg\alpha\Rightarrow\neg\beta)$.

où α, β et γ sont des formules quelconques.

Les axiomes 1-3 sont dus à Łukasiewicz, les axiomes de 4-6 sont dérivés des trois premiers et sont ajoutés pour simplifier les preuves.

Sachant que la seule régle autorisée est le Modus Ponens :

$$\frac{\vdash \alpha \qquad \vdash \alpha \Rightarrow \beta}{\vdash \beta}$$

Q1)- Prouver que : P $\vdash$ Q $\Rightarrow$ P.

Q2)- Prouver que : $\neg$ Q $\vdash$ Q $\Rightarrow$ P.

Q3)- Prouver le raisonnement pas contrapposée : P $\Rightarrow$ Q, $\neg$ Q $\vdash$ $\neg$ P.

Q4)- Montrez que l'axiome 6 peut être déduit des axiomes 1 et 3.

Chapitre 5

Le calcul des prédicats

Sommaire

5.1 Introduction

Le calcul des prédicats du premier ordre, ou calcul des relations, logique du premier ordre, logique quantificationnelle, ou tout simplement calcul des prédicats, est une formalisation du langage des mathématiques, proposée par Gottlob Frege, entre la fin du xixe siècle et le début du xxe siècle. La logique du premier ordre comporte deux parties :

— la syntaxe définit le vocabulaire symbolique de base ainsi que les règles permettant de construire des énoncés complexes,

— la sémantique interprète ces énoncés comme exprimant des relations entre les éléments d'un domaine, également appelé modèle.

Sur le plan syntaxique, les langages du premier ordre opposent deux grandes classes linguistiques :

— les constituants servant à identifier ou nommer des éléments du domaine : variables, symboles de constantes, termes ;

— les constituants servant à exprimer des propriétés ou des relations entre ces éléments : prédicats et formules.

La logique des prédicats a pour but de généraliser la logique des propositions. On peut considérer un prédicat comme un énoncé général où apparaissent des variables. Par exemple :
"X" est la soeur de "Y"
"si X est le père de Y et Y le père de Z alors X est le grand-père de Z"
Si l'on remplace toutes les variables d'un prédicat par des valeurs définies on obtient une proposition à laquelle on pourra associer une interprétation (vrai, faux).
Par exemple :
X = Lina et Y = Yanis, le premier énoncé donne : "Lina est la soeur de Yanis".
Un prédicat repésente donc potentiellement une classe de propositions. Par l'introduction de quantificateurs on peut représenter le fait qu'un énoncé est vrai pour toutes les valeurs possibles des variables ou qu'il existe au moins une valeur des variables qui rend l'énoncé vrai. Par exemple :
"quel que soit X si X est un homme alors X est mortel". Les variables pouvant prendre leur valeur dans des ensembles infinis, les quantificateurs permettent donc d'énoncer des faits correspondant à une infinité de propositions.

Un prédicat est une expression linguistique qui peut être reliée à un ou plusieurs éléments du domaine pour former une phrase. Par exemple, dans la phrase « Mars est une planète »,

l'expression « est une planète » est un prédicat qui est relié au nom (symbole de constante) « Mars » pour former une phrase. Et dans la phrase « Jupiter est plus grand que Mars », l'expression « est plus grand que » est un prédicat qui se relie aux deux noms, « Jupiter » et « Mars », pour former une phrase.

En logique mathématique, lorsqu'un prédicat est lié à une expression, on dit qu'il exprime une propriété (telle que la propriété d'être une planète), et lorsqu'il est lié à deux ou plusieurs expressions, on dit qu'il exprime une relation (telle que la relation d'être plus grand ou la relation est la soeur de dans l'exemple en haut). Ainsi on peut raisonner sur des énoncés comme « Tout x est gentil » et « Il existe un x tel que pour tout y, x est ami avec y », ce qui exprimé symboliquement se traduit par la formule :

$$\forall\ x\ \text{gentil}(x)\ \text{et}\ \exists\ x\ \forall\ y\ \text{amis}(x, y)$$

Il convient de noter cependant que la logique du premier ordre ne contient aucune relation spécifique (comme telle relation d'ordre, d'inclusion ou d'égalité) ; en fait, il ne s'agit que d'étudier la façon dont on doit parler et raisonner avec les expressions du langage mathématique.

Les traits caractéristiques de la logique du premier ordre sont :

— l'utilisation de variables comme x, y, etc. pour dénoter des éléments du domaine d'interprétation ;

— l'utilisation de prédicats (ou relations) sur les éléments ;

— l'utilisation de connecteurs logiques (et, ou, implique etc.) ;

— l'utilisation de deux quantificateurs, l'un universel (« Quel que soit », « pour tout » noté $\forall$) et l'autre existentiel (« il existe au moins un ... tel que », noté $\exists$), appliqués aux variables uniquement.

Le calcul des prédicats du premier ordre égalitaire adjoint au calcul des prédicats un symbole de relation, l'égalité, dont l'interprétation est l'affirmation que deux éléments sont les mêmes, et qui est axiomatisée en conséquence. Suivant le contexte, on peut parler simplement de calcul des prédicats pour le calcul des prédicats égalitaire.

On parle de logique du premier ordre par opposition aux logiques d'ordre supérieur, où l'on peut aussi appliquer les quantificateurs et les prédicats aux prédicats ou aux fonctions, en plus des variables. En outre, cet article ne traite que de la logique du premier ordre classique, mais on notera qu'il existe aussi une logique du premier ordre intuitionniste.

5.2 Histoire

Emmanuel Kant croyait à tort que la logique de son temps, celle d'Aristote, était une science complète et définitivement achevée (préface de la seconde édition de la critique de la raison pure, 17871). Les logiciens2 du xixe siècle se sont rendu compte que la théorie d'Aristote ne dit rien ou presque sur la logique des relations. Gottlob Frege et beaucoup d'autres ont comblé cette lacune en définissant le calcul des prédicats au premier ordre.

Kurt Gödel a démontré en 1929 (dans sa thèse de doctorat) que le calcul des prédicats est complet, au sens où on peut donner un nombre fini de principes (axiomes logiques, schémas d'axiomes logiques et règles de déduction3) qui suffisent pour déduire de façon mécanique toutes les lois logiques (voir le théorème de complétude de Gödel). Il y a équivalence entre la présentation sémantique et la présentation syntaxique du calcul des prédicats. Tout énoncé universellement valide (c'est-à-dire vrai dans tout modèle du langage utilisé) est un théorème (c'est-à-dire qu'il peut être déduit d'un calcul, un système de règles logiques et d'axiomes, de façon équivalente un système à la Hilbert, la déduction naturelle, ou le calcul des séquents), et réciproquement. La logique du premier ordre est donc achevée au sens où le problème de la correction logique des démonstrations y est résolu. Elle continue cependant à faire l'objet d'importantes recherches : théorie des modèles, théorie de la démonstration, mécanisation du raisonnement...

5.3 Rappels

Alors que la logique propositionnelle traite des propositions déclaratives simples, la logique du premier ordre couvre en plus les prédicats et la quantification.

Un prédicat prend en entrée une ou plusieurs entités du domaine du discours et en sortie, il est soit Vrai, soit Faux. Considérons les deux phrases "Socrate est un philosophe" et "Platon est un philosophe". En logique propositionnelle, ces phrases sont considérées comme non liées et peuvent être désignées, par exemple, par des variables telles que p et q. Le prédicat "est un philosophe" apparaît dans les deux phrases, dont la structure commune est "a est un philosophe". La variable a est instanciée en tant que "Socrate" dans la première phrase, et est instanciée en tant que "Platon" dans la deuxième phrase. Alors que la logique du premier ordre permet l'utilisation de prédicats, tels que "est un philosophe" dans cet exemple, la logique propositionnelle ne le permet pas.

Les relations entre les prédicats peuvent être énoncées à l'aide de connecteurs logiques.

Considérons, par exemple, la formule du premier ordre "si a est un philosophe, alors a est un savant". Cette formule est un énoncé conditionnel dont l'hypothèse est " a est un philosophe " et la conclusion " a est un savant ". La vérité de cette formule dépend de l'objet désigné par a, et des interprétations des prédicats "est un philosophe" et "est un savant".

Les quantificateurs peuvent être appliqués aux variables d'une formule. La variable a de la formule précédente peut être quantifiée universellement, par exemple avec la phrase du premier ordre "Pour tout a, si a est un philosophe, alors a est un savant". Le quantificateur universel "pour chaque" dans cette phrase exprime l'idée que l'affirmation "si a est un philosophe, alors a est un savant" est valable pour tous les choix de a.

La négation de la phrase "Pour tout a, si a est un philosophe, alors a est un savant" est logiquement équivalente à la phrase "Il existe a tel que a est un philosophe et a n'est pas un savant". Le quantificateur existentiel "il existe" exprime l'idée que l'affirmation "a est un philosophe et a n'est pas un savant" est valable pour un certain choix de a.

Les prédicats "est un philosophe" et "est un savant" prennent chacun une seule variable. En général, les prédicats peuvent prendre plusieurs variables. Dans la phrase du premier ordre "Socrate est le professeur de Platon", le prédicat "est le professeur de" prend deux variables.

En logique des prédicats, les éléments de base du langage ne sont plus des propositions mais des prédicats.

Ex : la mer est bleue

sujet prédicat

Le langage des prédicats du premier ordre se démarque du calcul propositionnel par l'utilisation de variables pouvant représenter des individus (des constantes ou des termes). Ces variables sont incluses dans des prédicats ou relations dont l'interprétation dépend des individus que représentent ces variables.

Ex : le prédicat IL Pleut(x) : signifie il pleut le jour x. si l'on substitue une constante à la variable x, le prédicat devient par exemple, IL Pleut(Lundi) et signifie alors il pleut lundi.

Un prédicat peut être vu comme une fonction propositionnelle bleu(x) qui prend la valeur vraie lorsque x est la mer et fausse lorsque x est le soleil. bleu(mer)=vrai ; bleu(soleil)=faux

Le prédicat Père prend deux arguments : le père et l'enfant. Ex : la relation ou le prédicat x est le père de y est définie par : Père(x,y)

On appellé arité d'un prédicat, le nombre d'argument qu'il requiert. Ex : le prédicat bleu

est d'arité 1 (fonction à 1 paramétre) Le prédicat Père est d'arité 2 (fonction à 2 paramètres)

Il faut noter que les prédicats ne sont jamais des variables, afin de rester en logique du premier ordre.

Remarque : De la même façon que pour le langage propositionnel, on peut donner une grammaire du langage des prédicats. Le langage propositionnel est un sous-ensemble du calcul des prédicats où tous les prédicats sont d'arité 0.

5.4 Le langage des prédicats du premier ordre

Le vocabulaire de la logique des prédicats est constitué des classes de symboles :

— de connecteurs : $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ qui se lisent respectivement non, et, ou, implique et équivalent. (NB : on retrouve les opérateurs de l'algébre de Boole : (non), (et), (ou))

— de délimiteurs : les parenthèses ()

— des deux constantes propositionelles V (vrai) et F (faux)

— de constantes (minuscules de l'alphabet latin et les concaténations de telles lettres) C = a, b, c,...z, aa,...

— de variables (majuscules de l'alphabet latin, et les concaténations de telles lettres) V = A, B, Z, AA..

— de prédicats (majuscules P)

L'arité d'un prédicat est le nombre d'argument du prédicat. C'est un nombre positif

Si le prédicat est d'arité 0, il correspond à la notion de proposition de la logique des propositions

— de fonctions (en minuscules : f, g sucesseur). Chaque symbole de fonction a une arité fixée.

L'arité d'une fonction est le nombre d'argument de la fonction. C'est un nombre positif.

Si la fonction est d'arité 0, elle correspond à la notion de constante.

— de quantificateurs

$\exists$ prononcé "il existe" est le quantificateur existentiel

$\forall$ prononcé "quel que soit" est le quantificateur universel

5.5 Les termes

Par définition, tout terme est engendré par application des deux lois suivantes :

— Les constantes et les variables sont des termes,

— si f est un symbole de fonction d'arité n (n>=1) et si t1,..,tn sont des termes alors f(t1,..,tn) est un terme.

Exemple :

- successeur(X) est un terme
- poids(b) est un terme
- successeur(poids(b)) est un terme
- P(X, bleu) n'est pas un terme
- poids(P(X)) n'est pas un terme

5.6 Les atomes

Par définition, tout atome est engendré par application des deux lois suivantes :

— propositions sont des atomes,

— si P est un prédicat d'arité n (n >=1) et si t1..tn sont des termes alors P(t1..tn) est un atome.

Exemple : - P(X, bleu) est un atome
- VIDE est un atome
- ENTRE(table,X, appui(fenetre)) est un atome
- successeur(X) n'est pas un atome
- appui(fenetre) n'est pas un atome

- Un **littéral** est soit :
- un atome,
- soit un atome précédé du symbole de négation ¬.

5.7 Les formules bien formées

Le langage est constitué de l'ensemble des Formules Bien Formées (appelées aussi : FBFs ou Well Formed-Formula WFF) ou expressions bien formées défini comme suit :

— atomes sont des fbfs,

— si F et G sont des fbfs alors ($\neg$G), (F $\wedge$ G), (F $\vee$ G), (F $\Rightarrow$ G) et (F $\Leftrightarrow$ G) sont des fbfs,

— si G est une fbf et X une variable alors ($\exists$X)G et ($\forall$X)G sont des fbfs,

— toutes les fbfs sont obtenues par application des 3 règles ci-dessus.

Priorité des connecteurs et quantificateurs :

$\neg$, $\wedge$, $\vee$, $\forall$, $\exists$, $\Rightarrow$, $\Leftrightarrow$.

Relation entre les quantificateurs universel et existentiel :

$$\overline{\forall x \alpha} \equiv \exists x \overline{\alpha};$$

$$\overline{\exists x \alpha} \equiv \forall x \overline{\alpha};$$

5.8 Variables libres et liées

Les variables qui apparaissent dans une formule sont dites libres ou liées, selon le principe suivant :

• toutes les variables d'une formule sans quantificateurs sont libres.

• si x est libre dans une formule , x est liée dans x. et dans x.

Exemple : Dans les prédicats suivants :

P(z,f(x)) ; x et z sont libres

$\forall x \ \forall z$ P(z,f(x)) ; x et z sont liées.

Exemples :

A = $\forall$ X ($\exists$ Y P(X, Y) $\wedge$ Q(X, Z)) $\wedge$ R(X)

B = $\forall$ X (($\exists$ Y Q(X, Y)) $\wedge$ P(X, Y, Z))

Une variable est libre (respectivement liée) si au moins une de ses occurences est libre (respectivement liée).

Dans les exemples précédents :

- Variables libres de A = {Z, X}

- Variables liées de A = {X, Y}
- Variables libres de B = {Z, Y}
- Variables liées de B = {X, Y}

Exemple :

Dans $\forall$ y (($p(x) \vee \exists$ x $p(x)) \wedge q(y)$) , la première occurrence de x est libre, tandis que la deuxième occurrence est liée. L'occurrence de y est liée.

5.9 Formule fermée / Ouverte

Une formule dont toutes les variables sont liées est dite fermée (close). Par exemple :

$$= \forall.P(x) \Rightarrow (\exists y.Q(x, y))$$

Si ce n'est pas le cas, la formule est dite ouverte. Pour une formule ouverte dont les variables x, y, et z sont libres on notera (x, y, z) lorsqu'on désirera indiquer quelles sont les variables libres de .

$$(x, z) = P(x) \Rightarrow (\exists\ y.Q(x, y, z))$$

Est une formule ouverte où x et z sont libres.

Exemples de formules fermées et ouvertes :

La formule A = $\forall$ y (($p(x) \vee \exists$ x $p(x)) \wedge q(y)$) est ouverte, car il y a une occurrence de variable libre.

La formule $\forall$ x $\forall$ y (($p(x) \vee \exists$ x $p(x))$ $q(y)$) est fermée (c'est la fermeture universelle de A).

La formule $\exists$ x $\forall$ y (($p(x) \vee \exists$ x $p(x)) \wedge q(y)$) est également fermée (c'est la fermeture existentielle de A).

5.10 Substitution et instantiation

L'opération de substitution consiste à remplacer (purement syntaxiquement) certaines variables libres d'une formule w par des termes.

Si w est une formule où x1, . . . , xn sont des variables libres, et ϕ est la substitution (t1/x1,. . . , tn/xn), wϕ dénote la formule obtenue en remplaçant toutes les occurences de xi dans w par ti (pour i = 1 à n).

Exemples :

Soit la formule F(x) donnée par :

F(x) = ((R(c,x) ∧ ¬x=c) ∧(∃yg(y,y)=x))

Le résultat de substitution du terme f(z) à la variable x :

F(f(z)/x) = ((R(c, f(z)) ∧ ¬ f(z)=c) ∧ (∃ yg(y,y)= f(z)))

Exemples de substitutions :

(p(x) ∨ q(x,y)){x/z} = p(z) ∨ q(z,y)

(p(x) ∨ q(x,y)){x/y} = p(y) ∨ q(y,y)

(∀ x q(x,y)){x/z} = ∀ x q(x,y)

(p(x) ∨ q(x,y)){x/z,y/z} = p(z) ∨ q(z,z)

(p(x) ∨ q(x,y)){x/f(x)} = p(f(x)) ∨ q(f(x),y)

(p(x) ∨ q(x,y)){x/y,y/a} = p(y) ∨ q(y,a)

(p(x) ∨ q(x,y)){x/y,y/x} = p(y) ∨ q(y,x)

N.B. : les trois dernières substitutions sont particulières :

dans {x/f(x)}, la variable substituée apparaît dans le terme, et n'est donc pas éliminé lors de l'application de la substitution ;

dans {x/y,y/a} et {x/y,y/x}, la variable y apparaît et comme variable à substituer et dans le terme remplaçant x, et n'est donc également pas éliminé lors de l'application de la substitution.

5.11 Sémantique du calcul des prédicats

5.11.1 Introduction

La sémantique attribue une signification aux expressions. Cette signification est compositionnelle, c'est-à-dire qu'elle est en fonction de ses constituants. Les constituants d'une formule en langage des prédicats sont : - les constantes, - fonctions et les prédicats. Et le sens d'une formule fermée prendra l'une des valeurs (Vrai ou Faux), qui sera calculée de à partir des interprétations des constantes, fonctions et prédicats et par application des règles propres aux connecteurs logiques et aux quantificateurs.

Le sens d'une formule fermée sera l'une des valeurs t ou f de l'anneau booléen. Cette valeur sera calculée à partir des interprétations des constantes, fonctions et prédicats et par application des règles propres aux connecteurs logiques et aux quantificateurs. La présence de variables nécessite l'introduction d'opérations de substitutions.

5.11.2 Interprétations

Pour évaluer la formule ∀ x F(x) il faut connaître l'ensemble des valeurs que peut prendre la variable x. Cet ensemble est appelé le **domaine de discours**, noté **D**.

Une interprétation d'une fbf F est définie par les cinq étapes suivantes :

1. Choix d'un domaine d'interprétation non vide D
2. Assignation à chaque constante de F d'un élément de D
3. Assignation à chaque proposition de F d'un élément de {V,F}
4. Assignation à chaque prédicat d'arité n (n>=1) d'une application de Dn dans {V,F}
5. Assignation à chaque fonction d'arité (n>=1) d'une application de D^n dans D.

on dit alors qu'on a une interprétation I de F sur D. et l'ensemble d'un domaine de discours et une fonction d'interprétation constituent un modèle M. La présence de variables nécessite l'introduction d'opérations de substitutions dans D.

L'évaluation d'une formule F nécessite donc la spécification de :

- Un modèle M.
- Une fonction d'assignation ou de substitution.

Exemples :

1- Soit la fbf : F1 : (∀ X) P(X)

Soit une interprétation I1 de F1

i1 : D1 = 1, 2 où

i1[P(1)] = F

i1[P(2)] = V

2- Soit la fbf : F2 : (∀ X) (∃ Y) Q(X, Y)

Soit une interprétation I2 de F2

i2 : D2 = 1, 3 où

i2[Q(1, 1)] = F

i2[Q(1, 3)] = V

i2[Q(3, 1)] = F

i2[Q(3, 3)] = F

3- Soit la fbf : F3 : (∀ X) (R(X) ∧T(f(X), a))

Soit une interprétation I3 de F3

i3 : D1 = 4, 5 a = 4 f(4) = 5 f(5) = 4

i3[R(4)] = V

i3[R(5)] = F

i3[T(4, 4)] = V

i3[T(5, 4)] = V

pour F3 :

T(f(X), a) = V si X = 4 et a = 4.

5.11.3 Valeur selon une interprétation

Soit une interprétation I de domaine D d'une fbf F :

— 1. Si F est une proposition alors la valeur qui lui est assignée par définition de I est appelée valeur de F selon I.

— 2. Si F est un littéral non propositionnel alors pour chaque choix de valeurs dans D pour les variables de F (s'il en existe) on obtiendra une valeur V ou F en suivant la définition de I. Cette valeur est dite valeur de F selon I pour le choix des valeurs de variables.
Exemple : pour F3 ; T(f(X), a) = V si X = 4 et a = 4

— Si F est de la forme (∀ X)F', la valeur de F sera Vraie, si la valeur de F' selon I pour toutes les valeurs de la variable (dans D) est Vraie sinon la valeur de F sera Fausse.
Ex : Soit la fbf : F1 : (∀ X) P(X)
Soit une interprétation i1 de F1
i1 : D1 = {1,2} où
i1[P(1)] = F
i1[P(2)] = V
F1 est Fausse selon l'interprétation i1 car i1[P(1)] = F

— Si F est de la forme (∃ X) F', la valeur de F sera Vraie si la valeur de F' selon I pour au moins une valeur de X (dans D) est Vraie sinon la valeur de F sera Fausse.
Exemple : Soit la fbf : F2 : (∀ X) (∃ Y) Q(X, Y)
Soit une interprétation I2 de F2
i2 : D2 = {1, 3} où
i2[Q(1, 1)] = F
i2[Q(1, 3)] = V
i2[Q(3, 1)] = F

i2[Q(3, 3)] = F

Valeur de Q(X, Y) dans I2 est V quand X = 1 et Y = 3 donc $\exists$ Y Q(X, Y) est V selon I2 quand X=1.

— Si F est de la forme ($\neg$F') ou (F' $\wedge$ F") ou (F' $\vee$ F") ou (F' $\Rightarrow$ F") ou (F' $\Leftrightarrow$ F"), les connecteurs gardent la même sémantique qu'en calcul propositionnel. On définira la valeur de F selon I (quand les valeurs F' et F" selon I seront définies) au moyen des tables de vérité.

Remarques :

• Il y a une infinité d'interprétations pour F.

G1 = $\forall$ X $\exists$ Y P(X, Y)

Si D = N

i1 : P(X, Y) est équivalent à X $\geq$ Y G1 est V dans i1 i2 : P(X, Y) est équivalent à X > Y G1 est F dans i2 si X=0 il n'existe pas Y<0 dans N

• On ne peut pas interpréter une fbf contenant des variables libres.

G4 : Q(Y) Y libre dans G4

Soit I : D où Q : D élément de {V,F}

Quel sens attribuer à la variable libre ?

- soit constante ?

- soit variable parcourant D ? ==> V

==> On se limitera aux fbfs fermées.

Exercice :

On considère les formules suivantes dans le L={R2, =

ϕ1 : $\forall$ X $\forall Y$ ($\exists$ Z (R(X, Z) $\wedge$ R(Z, Y)) $\Rightarrow$ R(X, Y))

ϕ2 :$\forall$ X $\forall Y$ (R(X, Y) $\wedge$ R(Y, X) $\Leftrightarrow$ Y=X)

Déterminer la validité de ces formules dans les interprétations suivantes :

D1=N et R(X,Y) est vraie ssi X $\leq$ Y

et R(X,Y) est fausse sinon

(le prédicat = a l'interprétation standard)

D2=$\mathbb{N}^*$ et R(X,Y) est vraie ssi X Y et X divise Y

et R(X,Y) est fausse sinon

(le prédicat = a l'interprétation standard)

Solution :

Les formules proposées sont closes, elles sont valides ou non satisfaisables dans les domaines proposées. Lorsque la formule est valide, nous donnons les valuations correspondantes des variables liées.

1- $\phi 1$: $\forall$ X $\forall Y$ ($\exists$ Z (R(X, Z) $\wedge$ R(Z, Y)) $\Rightarrow$ R(X, Y))

Cette formule est valide dans D1 et D2 en raison de la transitivité de ces deux relations d'ordre.

2- $\phi 2$:$\forall$ X $\forall Y$ (R(X, Y) $\wedge$ R(Y, X) $\Leftrightarrow$ Y=X)

Cette formule est valide dans D1 et D2 en raison de l'antisymétrie de ces deux relations d'ordre.

Exercice :

On considère un sous-ensemble du calcul des prédicats avec :

a et b comme symboles de constantes

f comme symbole de fonction unaire

P comme symbole de prédicat binaire

Soit i une interprétation de ce langage définie par son domaine D={1,2} par :

i[a] = 1 ;

i[b] = 2 ;

i[f(1)] = 2 ;

i[f(2)] = 1 ;

i[P(U, V)] = V si et seulement si U = 1

Etablir la valeur de vérité des formules suivantes :

a) P(a,f(a))

b) P(b,f(b))

c) $\forall$ X $\forall$ Y P(X,Y)

d) $\forall$ X $\forall$ Y (P(X,Y) $\Rightarrow$ P(f(X),f(Y)))

e) $\exists$ X $\forall Y$ (P(X,Y) $\Rightarrow$ P(f(X),f(Y)))

5.12 Quelques propriétés

5.12.1 Validité

Une fbf A est une tautologie (valide) si et seulement si elle est vraie dans toute interprétation ; on écrit alors : $\models$ A

Exemple :

$\neg$ A $\vee$ A est une formule valide.

Une fbf est invalide si et seulement si elle n'est pas valide.

Exemple :

A $\wedge$ B et A $\vee$ B sont deux formules invalides il suffit que A et B soient fausses.

Exercice :

Etablir la validité des formules suivantes :

a) ($\forall$ X $\exists$ Y P(X, Y)) $\Rightarrow$ ($\exists$ X $\exists$ Y P(X, Y))

b) ($\exists$ X $\forall$ Y $\neg$P(X, Y) $\Rightarrow$ $\exists$ X P(X,X)).

5.12.2 Insatisfaisabilité

Une fbf est inconsistante ou insatisfiable si et seulement si elle est fausse dans toute interprétation.

Exemple :

$\neg$A $\wedge$ A est une formule inconsistante.

Une fbf A est consistante ou satisfiable :

- si et seulement si elle n'est pas inconsistante.
- si il existe une interprétation i telle que i[A] = V.
- si elle admet un modèle.

Exemple :

A $\wedge$ B et A $\vee$ B sont deux formules consistantes il suffit que A et B soient vraies.

5.12.3 Conséquence logique

A est une conséquence logique de E si et seulement si toutes les interprétations qui rendent vraies toutes les formules de E rendent vraie la formule A. On écrit alors E $\models$ A.

On dit qu'une formule C est une conséquence logique de H1.. Hn :

- si et seulement si tout modèle de $H1...Hn$ est un modèle de C

- si et seulement si $H1\ H2\wedge\ ...\ \wedge$ Hn $\Rightarrow$ C est valide.

Dans ce contexte les formule Hi sont les hypothèses et C est la conclusion.

5.12.4 Indécidabilité et semi-décidabilité de la logique des prédicats

Lorsqu'une formule ne contient pas de variable, on peut, comme en calcul propositionnel, en utilisant les tables de vérité, déterminer en un nombre fini d'opérations si cette formule est valide ou non inconsistante ou non. La situation est plus complexe en présence de variables et donc de quantificateurs car il y a une infinité d'interprétations. On montre qu'il est impossible de proposer un algorithme général capable de décider en un nombre fini d'opérations de la validité ou de la non validité de n'importe quelle formule de la logique des prédicats du premier ordre.On dit que la logique des prédicats est indécidable. (Théorème d'indécidabilité de Church) Cependant, on peut proposer des algorithmes généraux pour décider de la validité de certaines familles de fbfs :

- si la fbf est valide ils s'arrêteront
- si la fbf est non valide ils risquent de ne pas s'arrêter

La logique des prédicats est semi-décidable

Les principales techniques proposées sont :

- le théorème d'Herbrant
- la méthode de Davis et Putman
- le principe de résolution

5.13 Exercices

Exercice 1 :

Dans le langage $L = \{R_2, =\}$, on considère les formules suivantes : (le prédicat $=$ a l'interprétation standard).

$\Gamma_1 : \forall x\ \exists y\ R(x, y)$

$\Gamma_2 : \forall x\ \forall y\ (\exists z\ (R(x, z)\wedge\ R(z, y)) \Rightarrow R(x, y))$

$\Gamma_3 : \forall x\ \forall y\ (R(x, y)\wedge\ R(y, x) \Leftrightarrow y = x))$

$\Gamma_4 : \forall x\ \forall y\ (\exists z\ \neg R(z, x)\vee\ \neg R(z, y)\ \vee\ x = y)$

Déterminer la validité de ces formules dans les interprétations suivantes :

- $D_1 = \mathbb{N}$ *et* $R(x, y)$ est vraie *si et seulement si* x≤y et $R(x, y)$ est fausse sinon.
- $D_2 = \mathbb{N}^*$ et $R(x, y)$ est vraie *si et seulement si* x et x *divise* y et $R(x, y)$ est fausse sinon.

Solution :

Les formules proposés étant closes, elles sont valides et non satisfaisables dans les domaines proposées. Lorsque la formule est valide, nous donnons les valuations correspondantes des variables liées.

$\Gamma_1 : \forall x\ \exists y\ R(x, y)$

Cette formule est valide dans D_1. Pour toute assignation de x_0 à x, l'assignation de x_0 à y rend Γ_1 vraie. Il est en de même pour dans D_2 avec l'assignation de 2 X x_0 à y.

$\Gamma_2 : \forall x\ \forall y\ (\exists z\ (R(x, z) \wedge\ R(z, y)) \Rightarrow R(x, y))$

Cette formule est valide dans D_1 et D_2 en raison de la transitivité de ces deux relations d'ordre.

$\Gamma_3 : \forall x\ \forall y\ (R(x, y) \wedge\ R(y, x) \Leftrightarrow y = x))$

Cette formule est valide dans D_1 et D_2 en raison de l'antisymétrie de ces deux relations d'ordre.

$\Gamma_4 : \forall x\ \forall y\ (\exists z\ \neg R(z, x) \vee\ \neg R(z, y) \vee\ x = y)$

Cette formule est valide dans D_1 et D_2. Détaillons les valuations des variables liées qui autorisent cette assertion :

- Dans D_1 et D_2, toute valuation de z convient pour les valuations dans lesquelles x et y sont égaux.
- Dans D_1, dans le cas où x est inférieur à y et où x et y ont pour valuations x_0 et y_0, l'assignation de y_0 à z rend la formule vraie (et l'assignation de x_0 à z dans le cas contraire).
- Dans D_2, dans le cas où x divise y, l'assignation de y_0 à z rend la formule vraie, l'assignation de x_0 à z dans le cas où y divise x. N'importe laquelle de ces deux assignations convient si x et y ne sont pas dans une relation de divisibilité.

Exercice 2 :

Dans le langage $L = \{=, \neq\}$, on considère les formules suivantes :

$x \neq y \wedge \mathrm{y} \neq z \wedge \mathrm{z} \neq x$

$x = y \vee \mathrm{x} = \mathrm{z} \vee \mathrm{x} = \mathrm{t} \vee \mathrm{y} = \mathrm{z} \vee \mathrm{y} = \mathrm{t} \vee \mathrm{z} = \mathrm{t}$

$\forall t(\ \mathrm{x} \neq y \wedge \mathrm{y} \neq z \wedge \mathrm{z} \neq x \wedge (t = x \vee \mathrm{t} = \mathrm{y} \vee \mathrm{t} = \mathrm{z}\ ))$

Déterminer la validité des ces formules dans les cinq interprétations données par les cinq domaines d'interprétation suivants. L'interprétation du prédicat $x = y$ (resp. x) est vraie (resp. fausse) *si et seulement si* x et y sont égaux et fausse (resp. vraie) sinon.

$D_1 = \{0\}$

$D_2 = \{0,1\}$

$D_3 = \{0,1,2\}$

$D_4 = \{0,1,2,3\}$

$D_5 = \mathbb{N}$

Solution :

$x \neq y \wedge \mathrm{y} \neq z \wedge \mathrm{z} \neq x$

Cette formule n'est pas satisfaisable dans D_1 et D_2, car on ne peut pas trouver une valuation pour x, y et z telle que la formule soit vraie. Cette formule est satisfaisable dans les trois autres domaines, une valuation assignant $0, 1, 2$ respectivement à x, y et z convient. Elle n'est pas valide dans ces trois domaines car la valuation assignant 0 aux trois variables x, y et z rend la formule fausse.

$x = y \vee \mathrm{x} = \mathrm{z} \vee \mathrm{x} = \mathrm{t} \vee \mathrm{y} = \mathrm{z} \vee \mathrm{y} = \mathrm{t} \vee \mathrm{z} = \mathrm{t}$

Cette formule est valide dans D_1, car la seule valuation possible pour x, y, zet t est 0 et rend la formule vraie. Cette formule est satisfaisable mains non valide dans les quatres autres domaines, car elle est vraie pour cette même valuation, mais elle est fausse, par exemple, pour une valuation assignant 0 à t et 1 à x, y et z.

$\forall t(\ \mathrm{x} \neq y \wedge \mathrm{y} \neq z \wedge \mathrm{z} \neq x \wedge (t = x \vee \mathrm{t} = \mathrm{y} \vee \mathrm{t} = \mathrm{z}\ ))$

L'interprétation d'une formule ne dépend que des valuations de ses variables libres qui sont, ici, les variables x, y et z. Cette formule est satisfaisable *si et seulement si* le domaine d'interprétation a trois valeurs. Donc elle est satisfaisable dans D_3 et non satisfaisable dans les quatres autres domaines. Cependant, dans D_3 la formule n'est pas valide, car la valuation assignant 0 à x, y et z rend la formule fausse.

Chapitre 6

Applications et résolution

Sommaire

6.1 Introduction

La méthode de résolution étendue au calcul des prédicats est plus complexe que sa version en calcul propositionnel puisqu'elle doit prendre en compte l'existence des variables. Elle sert de base au langage Prolog, mais ce langage se limite à un type particulier de propositions et de conséquents sans en réstreindre la portée. Le démonstrateur implanté dans le langage Prolog réalise la démonstration automatique de théorème en appliquant systématiquement la méthode de résolution et en reportant les valuations ayant conduit aux succès de la démonstration.

6.2 Équivalence des formules bien formées

6.2.1 Formules équivalentes

Deux formules sont équivalentes quand elles ont même valeur dans toutes interprétation (notation : A $\equiv$ B). Soient $A(x)$ et $B(x)$ deux formules atomiques bien formées. Les formules équivalentes de la logique des propositions demeurent équivalentes en logique des prédicats :

$\forall x A(x) \wedge \forall x B(x) \equiv \forall x(A(x) \wedge$ B(x))
$\exists x A(x)$vee$\exists x B(x) \equiv \exists x(A(x) \vee$ B(x))
$\neg(\forall x A(x)) \equiv \exists \neg A(x)$
$\neg(\exists x A(x)) \equiv \forall \neg A(x)$

Remarques :
$\forall x A(x) \vee \forall x B(x) \not\equiv \forall x(A(x) \vee$ B(x))
$\exists x A(x) \wedge \exists x B(x) \not\equiv \exists x(A(x) \wedge$ B(x))

6.2.2 Forme prénexe

Elle consiste à accumuler les quantificateurs au début de la formule. L'utilit é de cette forme est qu'elle permet de mettre en évidence certaines relations logiques qui ne sont pas évidentes à voir sous les formes habituelles des formules.
- Une matrice est une formule du calcul des prédicats ne contenant aucun quantificateur.
- Toute formule admet une forme prénexe équivalente.
- Une forme prénexe est une formule du calcul des prédicats de la forme $Q1x1 ::: QnxnM$, où

Q désigne s ou $\forall$ et Mest une matrice.

Théorème :

Toute formule du calcul des prédicats peut être transformée en une formule équivalente sous forme normale prénexe conjonctive.

Définition :

Une formule est dite sous forme normale prénexe conjonctive si elle est de la forme :

$(\mathrm{Qv}_1)(Qv_2)...(Qv_n)[A_{11} \vee \mathrm{A}_{12}...\vee \mathrm{A}_{1n}] \vee ... \vee [\mathrm{A}_{m1} \wedge \mathrm{A}_{m2} \vee ... \vee \mathrm{A}_{mq}]$

Où Q est soit le quantificateur $\forall$ ou soit le quantificateur $\exists$, les v_i sont des variables distinctes qui ont une occurrence libre dans les A_{ij} . Chaque A_{ij} est une formule atomique ou une négation d'une formule atomique.

Algorithme de construction

Pour construire la forme prénexe, il faut :

1. Supprimer les connecteurs d'équivalence et d'implication.

2. Renommer certaines variables liées de manière à n'avoir plus de variable quantifiée deux fois.

3. Supprimer les quantificateurs inutiles (dont la variable quantifiée n'apparaît pas dans leur portée) s'il y en a.

4. Transférer toutes les occurrences de la négation devant les atomes en utilisant les règles de réécriture vues pour la logique des propositions et les règles supplémentaires suivantes :

$\neg(\forall x A) \equiv \exists \neg A$

$\neg(\exists x A) \equiv \forall \neg A$

5. Faire passer les quanticateurs en tête en utilisant les règles de réécriture suivantes (et en utilisant l'associativité, la commutativité ou le renommage de variable si nécessaire) :

- $(\forall\ x\ A \wedge B) = \forall\ x\ (A \wedge B)$ si B ne contient pas x ;
- $(\exists\ x\ A \wedge B) = \exists\ x(A \wedge B)$ si B ne contient pas x ;
- $(\forall\ x\ A \vee B) = \forall\ x\ (A \vee B)$ si B ne contient pas x.
- $(\exists\ x\ A \vee B) = \exists\ x(A \vee B)$ si B ne contient pas x ;

6.2.3 Forme de Skolem

La forme Skolem est constituée que des quantications universelles. La procédure de construction est comme suit :

1. Prendre la fermeture existentielle de D (c.à.d quantifier les variables libres de D par le quantificateur $\exists$) ;

2. Éliminer dans D tous les quantificateurs redondants ;

3. Renommer chaque variable quantifiée dans D plus d'une fois ;

4. Éliminer les occurrences des connecteurs autres que : $\neg$, $\vee$ et $\wedge$.

5. Pousser à droite tous les connecteurs en remplaçant :

$\neg(\forall$x A) par $\exists(\neg$A)

$\neg(\exists$ x A) par $\forall$ x ($\neg$A)

$\neg$(A$\vee$B) par $\neg$A $\wedge$ $\neg$B

$\neg$(A$\wedge$B) par $\neg$A $\vee$ $\neg$B

$\neg(\neg$A) par A

Jusqu'à ce que toutes les occurrences de $\neg$ précède immédiatement une formule atomique.

6. Pousser les quanticateurs à droite en remplaçant :

$\exists$ (A$\vee$B) par :

- A $\vee$ $\exists$B si x n'est pas libre dans A ;

- $\exists$ A$\vee$B si x n'est pas libre dans B.

$\exists$ (A$\wedge$B) par :

- A $\wedge$ $\exists$B si x n'est pas libre dans A ;

-$\exists$ A$\wedge$B si x n'est pas libre dans B.

$\forall$ (A$\wedge$B) par :

- A $\wedge$ $\forall$ x B si x n'est pas libre dans A ;

-$\forall x$ A$\wedge$B si x n'est pas libre dans B.

7. Eliminer les quantificateurs existentiels. Prendre la formule la plus à gauche de la forme (y) et la remplacer par $B(f(x1;:::;xn))$ où

- $x1;:::;xn$ sont des variables distinctes de (y) quantifiées universellement à gauche de (y).

- f est fonction à n arguments qui n'occure pas dans la formule. Répéter ce processus jusqu'à l'élimination complète des tous les quantificateurs extensionnels. Dans le cas particulier n = 0, remplacer (y) par $B(a)$ où a est une nouvelle constante qui n'occure pas dans la formule.

8. Déplacer les quantificateurs $\forall$ à gauche ;

9. Distribuer $\wedge$ et $\vee$;

10. Simplifier en utilisant les règles préservant la satisfaction.

6.2.4 Forme Clausale

Définition :

Une formule est sous forme clausale si elle est sous forme de Skolem et si sa matrice est sous forme normale conjonctive.

Une fois que la formule a été skolémisée, il ne reste que des variables quantifiées universellement. On peut donc se passer de l'écriture explicite des quantificateurs. La formule restante est composée d'atomes et de connecteurs logiques. On peut alors appliquer la même technique qu'en logique propositionnelle pour la mettre en forme normale conjonctive, puis sous forme d'un ensemble de clauses.

Un littéral est défini comme étant une formule atomique ou une négation d'une formule atomique. Une clause est une disjonction de plusieurs littéraux. Une formule du calcul des prédicats est dite sous forme clausale si elle est sous la forme :

$$\forall x_1 ... \forall x_n (\mathrm{C}_1 \wedge C_2 \wedge ... \wedge C_k)$$

Algorithme de construction : Pour construire la forme clausale, il faut :

1. Prendre la fermeture existentielle de D (c.à.d. quantifier les variables libres de D par le quantificateur $\exists$),
2. Éliminer dans D tous les quanticateurs redondants,
3. Renommer chaque variable quantiée dans D plus d'une fois,
4. Éliminer les occurrences des connecteurs autres que : $\neg$;$\vee$ et $\wedge$.
5. Pousser à droite tous les connecteurs en remplaçant :

$\neg(\forall$x A) par $\exists(\neg$A)

$\neg(\exists$ x A) par $\forall$ x ($\neg$A)

$\neg$(A$\vee$B) par $\neg$A $\wedge$ $\neg$B

$\neg$(A$\wedge$B) par $\neg$A $\vee$ $\neg$B

$\neg(\neg$A) par A

Lorsqu'une formule est sous forme clausale, on peut ensuite la décomposer en appliquant la règle :

$$\forall\ x\ (\mathrm{A} \wedge \mathrm{B}) \equiv (\forall\ x\mathrm{A}\ ) \wedge (\forall\ x\mathrm{B})$$

Exemple :

Si on repart de la formule skolémisée :

$F_{sk} = \forall\ x \forall\ y(P(x, f(x)) \Rightarrow (Q(f(x), y)\ \wedge R(y, g(x, y)))$

1. l'élimination des quantificateurs universels, donne :

$\neg\ P(x, f(x)) \vee (Q(f(x), y) \wedge R(y, g(x, y)))$

2. l'application de la transformation $\Rightarrow$, $\vee$, on obtient :

$\neg\ P(x, f(x)) \vee (Q(f(x), y) \wedge R(y; g(x, y)))$

3. Puis par la distributivité, on obtient :

$(\neg P(x, f(x)) \vee Q(f(x), y)) \wedge (\neg\ P(x, f(x)) \vee R(y, g(x, y)))$

4. Nous obtenons à la fin les deux clauses :

(a) $\neg\ P(x, f(x)) \vee Q(f(x), y)$

(b) $\neg\ P(x, f(x)) \vee R(y, g(x, y))$

Théorème

Soit S un ensemble de clauses résultant de la mise sous forme clausale d'une formule A. Alors, A est insatisfiable si et seulement si S est insatisfiable.

Ce théorème forme la base de nombreux démonstrateurs automatiques utilisant une représentation des formules sous forme de clauses. Il établit que la recherche de l'insatisfiabilité d'une formule A est équivalente à la recherche d'insatisfiabilité de sa représentation sous forme clausale S. Cependant, A et S ne sont pas logiquement équivalentes : seule la satisfiabilité est préservée.

Remarque : Toute formule est équivalente à une formule prénexe, Skolem et clausale.

6.3 Complétude et décidabilité

Théorème 1

(Théorème de complétude de Gödel) Le calcul des prédicats est complet : on peut donner un nombre fini de principes (axiomes logiques, schémas d'axiomes logiques et règles de déduction) qui suffisent pour déduire de façon mécanique toutes les formules universellement valide de la logique du premier ordre.

Théorème 2

La logique du premier ordre n'est pas décidable, mais elle est semi-décidable.

6.4 Exercices

Exercice 1 :

Transformer la formule suivante sous forme prénexe :

$$\forall\ x\ p(x) \wedge \exists\ y\ q(y) \Rightarrow \forall\ y\ (\mathrm{p(y)} \wedge \mathrm{q(y)})$$

1. :$\neg\ (\forall\ x\ p(x) \wedge \exists\ y\ q(y)) \vee \exists\ y\ (p(y) \wedge q(y))$ par suppression de $\Rightarrow$;
2. :$\neg\ (\forall\ x\ p(x) \wedge \exists\ y\ q(y)) \vee \exists\ z\ (p(z) \wedge q(z))$ par renommage des variables ;
3. $(\exists\ x\ \neg p(x) \wedge \forall\ y\ \neg q(y)) \vee \exists\ z\ (p(z) \wedge q(z))$ par transfert de la négation ;
4. $\exists\ x\ \forall\ y\ \exists\ z\ (\neg\ p(x) \vee \neg\ q(y) \vee (\ \mathrm{p(z)} \wedge q(z)))$ par déplacement des quanticateurs.

Exercice 2

Transformer la formule suivante en forme prénexe conjonctive :

$$\forall x[\forall y p(x) \vee\ \forall z q(z; y) \Rightarrow\ \neg \forall y r(x; y)]$$

1. $(\forall x)[(p(x) \wedge\ (\forall z) q(z; y)) \Rightarrow\ \neg(\forall y) r(x; y)]$;
2. $\forall x[(p(x) \wedge\ (\forall z) q(z; y)) \Rightarrow\ \neg \forall y_1 r(x; y_1)]$;
3. $\forall x[\neg(p(x) \wedge\ \exists z(z; y)) \vee \exists y_1 \neg r(x; y_1)]$;
4. $\forall x \exists z \exists y_1[\neg(\mathrm{p(x)} \wedge \neg \mathrm{q(z\,;y)}) \vee \neg \mathrm{r(x\,;y_1)}]$;
5. $\forall x \exists z \exists y_1[\neg(\mathrm{p(x)} \vee \neg \mathrm{r(x\,;y_1)}) \wedge (\neg \mathrm{q(z\,;y)} \vee \neg \mathrm{r(x\,;y_1)}]$.

Exercice 3 :

Transformer la formule suivante sous forme Skolem.

$\forall x(p(x) \Rightarrow\ \exists z(\neg \forall y(q(x, y) \Rightarrow\ \mathrm{p(f(t))}) \wedge \forall y(q(x, y) \Rightarrow\ \mathrm{p(x)}))$

1. Prendre la fermeture existentielle de D et éliminer le quantificateur redondant $\exists z$.

$\exists\ t \forall\ x(p(x) \Rightarrow (\neg\ \forall\ y(q(x, y) \Rightarrow\ p(f(t))) \wedge\ \forall\ y(q(x, y) \Rightarrow p(x))))$

2. Renommer la variable y, elle est quantifiée deux fois.

$\exists\ t \forall\ x(p(x) \Rightarrow (\neg\ \forall\ y(q(x, y) \Rightarrow\ p(f(t))) \wedge\ \forall\ z(q(x, z) \Rightarrow p(x))))$

3. Éliminer toutes les occurrences de $\Rightarrow$.

$\exists\, t \forall\, x(\neg\, p(x) \vee (\neg\, \forall\, y(\neg q(x,y) \vee p(f(t))) \wedge \forall\, z(\neg q(x,z) \vee p(x))))$

4. Pousser les quantificateurs $\exists$ y et $\forall$ z à droite.

$\exists\, t \forall\, x(\neg\, p(x) \vee (\exists\, y(q(x,y) \wedge \neg p(f(t))) \wedge \forall\, z(\neg q(x,z) \vee p(x))))$

5. Eliminer les quantificateurs $\exists$ t et $\exists$ y.

$\forall\, x(\neg\, p(x) \vee ((\mathrm{q(x,g(x))} \wedge \neg p(f(a))) \wedge (\forall\, z(\neg q(x,z) \vee p(x))))$

6. Déplacer le quantificateur $\forall$ à gauche.

$\forall\, x\, \forall\, z(\neg\, p(x) \vee ((\mathrm{q(x,g(x))} \wedge \neg p(f(a))) \wedge ((\neg q(x,z) \vee p(x))))$

8. Après distribution de $\vee$ et $\wedge$ simplification.

$\forall\, x\, (\neg\, p(x) \vee ((\mathrm{q(x,g(x))}) \wedge \neg p(f(a))$

Exercice 4 :

Soit F la formule :

$$\exists\, x\, \forall\, y\, R(x) \Rightarrow R(y)$$

1. Trouver un modèle pour F.

2. F est-elle valide ?

3. Donner la forme prénexe de F.

4. Donner la forme Skolem de F.

Solution :

$F \equiv \exists\, x\, \forall\, y\, R(x) \Rightarrow R(y)$

on a la variable x est liée et la variable y est libre.

1. On va trouver un modèle pour la formule F. Soit alors l'interprétation I de domaine de base $D = 2, 6, 20, 40$ et $R(x)$: "x est pair". Pour toutes les valuations possibles v(y) = 2 ; v(y) = 6 ; v(y) = 20 ; v(y) = 40 de la variable libre y, on constate que F est vraie, car :

$\exists\, x\, R(x)$ est vraie et $\exists\, x\, R(x) \Rightarrow R(y)$ est vraie

Donc I est un modèle de F

2. La forme prénexe de F est la suivante :

$F \equiv \exists\, x\, R(x) \Rightarrow R(y)$

$F \equiv \forall\, x\, (R(x) \Rightarrow R(y))$

Remarque : on a enlevé le quantificateur de y car la variable y est libre, elle n'est pas dans le champs de ce quantificateur donc ça ne sert à rien de garder ce quantificateur.

3. La forme prénexe de la négation de la formule F c'est exactement la négation de la forme prénexe de F, c'est à dire :
$\neg\ F \equiv \neg\ [\forall\ x\ (R(x) \Rightarrow R(y))]$
$\neg\ F \equiv \exists$x(R(x)$\Rightarrow\ R(y)$)

4. La forme skolem de F est exactement la forme prénexe de F car il n'y a pas de quantificateur existentiel dans la forme prénexe de F, donc :
$F_{sk} \equiv \forall\ x\ (R(x) \Rightarrow$ R(y)) .

6.5 Conclusion

Bien qu'il existe un système de preuve correct et complet, le calcul des prédicats n'est pas décidable. Si par exemple on considère le calcul des séquents, on ne va pas pouvoir déterminer qu'une formule n'est pas universellement valide car il y a un nombre non borné d'explorations possibles.

Toutefois, puisqu'une preuve est un objets finis et que tous les objets considérés sont dénombrables, et donc énumérables, il est possible d'énumérer toutes les preuves, et donc tous les théorèmes. Le calcul des prédicats est donc semi-décidable : si on veut déterminer si une formule A est universellement valide, il suffit de la comparer avec chacun des théorèmes : le calcul se terminera si Aest effectivement un théorème (mais dans un temps non borné), le calcul peut ne pas se terminer mais dans ce cas, on n'a pas la certitude que$A$ n'est pas un théorème. En effet, on ne sait pas si le calcul des théorèmes n'a pas été mené assez loin pour identifier A comme théorème ou si ce calcul ne se terminera pas parce que A n'est pas un théorème.

Bibliographie

[1] A. Turing, (1937). On Computable Numbers, with an Application to the Entscheidungsproblem, Proc. London Math. Soc., vol. 42, 230-265

[2] Stuart Russel, Peter Norvig. « Intelligence Artificielle. » 3ème édition.

[3] Olivier Bournez. Fondements de l'Informatique : Logique, Modèles, Calculs. Ecole Polytechnique, 2011. hal-00760775

[4] D. Hilbert, (1900). Les 23 problèmes de Hilbert, congrès international des mathématiciens, Paris, France.

[5] Pierre Wagner. Logique et philosophie. Ellipses, 2014.

[6] Rozière, P. (2004). Logique mathématique : introduction.

[7] Logic and complexity. Discrete Mathematics and Theoretical Computer Science. Springer.

[8] Logique et mathématiques pour l'informatique et l'IA. Christian Jacquemin (Auteur) 109 exercices corrigés Paru en décembre 1997.

[9] Lassaigne and de Rougemont, 2004, Lassaigne, R. and de Rougemont, M. (2004).