Université de Rennes

Master de cryptographie

Logique, théorie des modèles, théorie de la complexité

Notes de cours

Dimitri Petritis

UFR Mathématiques

Rennes
Janvier 2023

Dimitri Petritis
Institut de recherche mathématique de Rennes
Université de Rennes et CNRS (UMR6625)
Campus de Beaulieu
35042 Rennes Cedex
France

Mathematics subject classification (2010): 03-00 05-00 06-00 94-00

Table des matières

Table des matières iii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Bref aperçu historique . . . 2
  - 1.2.1 Le dixième problème de Hilbert . . . 2
  - 1.2.2 La théorie des ensembles avant 1900 . . . 3
  - 1.2.3 La théorie des ensembles après 1900 ; incomplétude de Gödel . . . 5
- 1.3 Calculabilité . . . 7
- 1.4 La logique comme langage . . . 8

2 Relations et graphes 11
- 2.1 Rappels sur le produit cartésien . . . 11
- 2.2 Relations . . . 12
- 2.3 Graphes dirigés . . . 17
- 2.4 Arbres . . . 20
- 2.5 Exercices . . . 25

3 Langages et grammaires formels 31
- 3.1 Motivation . . . 31
- 3.2 Grammaires formelles . . . 32
  - 3.2.1 Aspects combinatoires de $\mathbb{A}^*$ . . . 32
  - 3.2.2 Grammaires . . . 33
- 3.3 Hiérarchie de Chomsky . . . 35
- 3.4 Arbres de dérivation pour des grammaires de type 2 ou 3 . . . 36
- 3.5 Exercices . . . 38

4 Logique et langages 41
- 4.1 Algèbres de Boole, ensembles partiellement ordonnés, treillis . 42
- 4.2 Fonctions booléennes . . . 45
- 4.3 La logique comme langage . . . 47
  - 4.3.1 Logique propositionnelle . . . 47
  - 4.3.2 Logique du premier ordre (ou calcul des prédicats) . . . 52
- 4.4 Structures, modèles, théories, déductions . . . 56
- 4.5 Intermède : arithmétiques de Robinson et de Peano . . . 58

4.6 Le premier théorème d'incomplétude et les idées essentielles de sa démonstration . . . 59
4.7 Notes . . . 60

5 Automates à nombre fini d'états et langages réguliers 61
5.1 Automates finis déterministes . . . 62
5.2 Automates finis non-déterministes . . . 65
5.3 Équivalence entre automates déterministes et non-déterministes 67
5.4 Expressions régulières . . . 69
5.5 Automates finis probabilistes et langages stochastiques . . . 72
5.6 Exercices . . . 73

6 Langages non-réguliers et machines de Turing 79
6.1 Tests de non-régularité . . . 79
6.1.1 Lemme de l'itération . . . 79
6.1.2 Théorème de Myhill-Nerode . . . 80
6.2 Machines de Turing . . . 81
6.3 Définition d'un algorithme . . . 85
6.4 Décidabilité, reconnaissance, problème d'arrêt . . . 87
6.5 Calculabilité . . . 90
6.6 Exercices . . . 92

7 Complexité 97
7.1 Classe de complexité P . . . 97
7.2 Machines de Turing non-déterministes et classe de complexité NP . . . 99
7.3 Complétude NP . . . 102
7.4 Machines de Turing probabilistes, classe de complexité BPP . . 104
7.5 Exercices . . . 106

8 Stochasticité, complexité de Kolmogorov, entropie 107
8.1 Stochasticité, chaoticité, typicité : trois aspects de l'aléa . . . 108
8.1.1 Stochasticité . . . 108
8.1.2 Chaoticité et complexité de Kolmogorov . . . 110
8.1.3 Typicité et ensembles effectivement négligeables . . . 113
8.2 Réalisation de l'aléa par un procédé physique . . . 115

Références 119

Index 124

1 Introduction

> Nommer un ensemble, c'est proclamer l'existence d'un nouvel objet rassemblant des objets qui partagent une propriété.
>
> Patrick DEHORNOY, *La théorie des ensembles*, Paris (2017) [22].

1.1 Motivation

Dans une formation mathématique orientée vers des applications cryptographiques — et plus généralement calculatoires — nous souhaitons répondre aux questions suivantes :

1. qu'est-ce un ordinateur (ou un programme informatique, ou un algorithme) ?
2. que peut (en principe [1]) calculer un ordinateur (algorithme) ?
3. que peut (pratiquement [2]) calculer un ordinateur (algorithme) ?

Les deux premières questions semblent — aujourd'hui — saugrenues car *nous croyons que nous savons* ce qu'est un ordinateur et ce qu'on peut calculer avec un algorithme. Nous avons donc tendance à nous préoccuper principalement de la dernière question. Par exemple, si N est un grand entier naturel (à approximativement $n = \log N$ digits), selon l'algorithme et le type d'ordinateur utilisés, on peut le factoriser en $\mathcal{O}(\exp(n))$, $\mathcal{O}(\exp(n^{1/3}\log^{2/3} n))$, ou $\mathcal{O}(n^3)$ opérations. Ceci a des implications spectaculaires sur le temps nécessaire pour arriver au résultat de la factorisation comme le montre la table 1.1.

Cependant, historiquement il s'est avéré que les trois questions ont été extrêmement fertiles et donné naissance à plusieurs développements fondamentaux et ont permis de clarifier des notions algorithmiques. En mathé-

1. I.e. disposant des ressources finies mais non-bornées.
2. I.e. disposant des ressources bornées.

n	$\mathcal{O}(\exp(n))$	$\mathcal{O}(\exp(n^{1/3}(\log n)^{2/3}))$	$\mathcal{O}(n^3)$
100	1.26×10^{21}s $= 4.01 \times 10^{13}$a	3.13s $= 9.93 \times 10^{-8}$a	1×10^{-3}s $= 3.17 \times 10^{-11}$a
500	3.27×10^{141}s $= 1.31 \times 10^{134}$a	6.74×10^{10}s $= 2139$a	0.125s $= 3.96 \times 10^{-9}$a
1000	1.07×10^{292}s $= 3.39 \times 10^{284}$a	6.42×10^{17}s $= 2.03 \times 10^{10}$a	1s $= 3.17 \times 10^{-8}$a

TABLE 1.1 – Une estimation grossière de l'ordre de grandeur du temps nécessaire pour factoriser un entier à n bits (avec $n = 100, 500, 1000$), sous l'hypothèse que l'algortihme s'exécute sur un ordinateur qui effectue une opération par nanoseconde comme fonction de la complexité algorithmique de la tâche. Lorsque le protocole RSA fut proposé [58] en 1978, le meilleur algorithme avait une complexité en $\mathcal{O}(\exp(n))$. Le meilleur algorithme classique connu de nos jours [39] s'effectue en un temps $\mathcal{O}(\exp(n^{1/3}\log^{2/3} n))$. L'algorithme de factorisation quantique de Shor [61] nécessite un temps $\mathcal{O}(n^3)$. Pour mémoire : l'« âge de l'univers » est 1.5×10^{10} a.

matiques fondamentales, elles ont contribué à une meilleure formulation de la **théorie des ensembles** — initialement conçue pour donner un sens précis à la notion de l'**infini** (ou plutôt des infinis) — et une nouvelle approche de la **logique mathématique**. En algorithmique elles ont clarifié la notion de **calculabilité effective**.

1.2 Bref aperçu historique

1.2.1 Le dixième problème de Hilbert

Lors du congrès international des mathématiciens tenu en 1900 à Paris, Hilbert [3] a énoncé 23 problèmes dignes d'être étudiés durant le siècle qui s'ouvrait. Nous nous intéressons plus particulièrement au dixième problème de la liste que Hilbert posait ainsi [4] ([32]) :

> **10. Détermination de la résolubilité d'une équation diophantienne** : Soit une équation diophantienne à coefficients entiers rationnels comportant un nombre arbitraire d'inconnues : *on doit spécifier un processus qui, en un nombre fini d'étapes, décide si l'équation est résoluble dans l'ensemble de nombres entiers.*

3. David Hilbert, né à Königsberg (1862) — jadis en Prusse orientale, aujourd'hui Kaliningrad, enclave russe sur la mer Baltique entre la Lituanie et la Pologne — mort à Göttingen (1942). Mathématicien allemand de renom ayant des contributions majeures dans des nombreuses branches de mathématiques : logique, théorie des ensembles, géométrie, analyse fonctionnelle où il a, en particulier, introduit les espaces dont il est l'éponyme. Pour l'anecdote, mentionnons que même le nom de sa ville natale est relié à un problème mathématique célèbre, le « problème de sept ponts de Königsberg », résolu par Leonhard Euler en 1736.

4. **10. Entscheidung der Lösbarkeit einer Diophantischen Gleichung :** Eine Diophantische Gleichung mit irgend welchen Unbekannten und mit ganzen rationalen Zahlencoefficienten sei vorgelegt : *man soll ein Verfahren angeben, nach welchem sich mittelst einer endlichen Anzahl von Operationen entscheiden läßt, ob die Gleichung in ganzen rationalen Zahlen lösbar ist.*

Ce problème est mentionné dans la littérature comme l'*Entscheidungsproblem*. On remarque que, dans son énoncé, Hilbert utilise le terme — quelque peu désuet de nos jours — d'« entiers rationnels » (ganze rationale Zahlen) pour signifier les entiers ordinaires (de $\mathbb{Z}$)[5]. Il s'agit d'un problème de décision qui attend une réponse « oui » ou « non ».

Si on restreint la classe des polynômes à ceux d'une variable, c'est un exercice de montrer que le *Entscheidungsproblem* est décidable. Baker a établi en 1968 [2] que ce problème est encore décidable à l'intérieur de la classe des polynômes à deux variables ; il a été montré par Matiyasevich en 1970 (voir [46] pour la version traduite du théorème) — en se basant sur des résultats antérieurs de Davis, Putnam et Robinson [20] — qu'il n'existe pas de tel algorithme pour des polynômes à 3 variables ou plus[6].

La thèse de Hilbert était que les mathématiques pouvaient être formalisées comme un système de preuves finies obtenues en se basant sur une petit nombre d'axiomes et que tous les problèmes mathématiques admettraient une solution. Il l'avait formulée, on ne peut pas plus clairement [32, p. 262], lors de son exposé[7] au congrès :

> Cette conviction de la solubilité de tout problème mathématique est une incitation puissante pour nous au cours de notre travail ; nous entendons en nous l'injonction constante : voici le problème, cherche la solution. Tu peux la trouver par pure pensée car, en mathématiques, il n'y a pas d'*ignorabimus !*

La stèle funéraire de Hilbert (cf. figure 1.1) porte par ailleurs l'épitaphe *„Wir müssen wissen, wir werden wissen"*[8].

1.2.2 La théorie des ensembles avant 1900

On verra que ce problème a été à l'origine de nombreux développements théoriques qui ont influencé les fondements mêmes de mathématiques et ont eu des conséquences majeures en algorithmique.

Mais, pour l'instant, replaçons-nous dans le contexte scientifique de 1900. À cette époque, on disposait d'une théorie des ensembles telle que développée, jusqu'alors par

5. En les distinguant ainsi des entiers algébriques qui sont des racines d'un polynôme de la forme $x^m + a_{m-1}x^{m-1} + \ldots a_0 = 0$, pour un certain m, avec $a_i \in \mathbb{Z}$, pour $i = 0, \ldots, m-1$ (le terme $a_m = 1$).

6. Précisions historiques sur la contribution essentielle de Julia Robinson à la résolution du problème.

7. Diese Überzeugung von der Lösbarkeit eines jeden mathematischen Problems ist uns ein kräftiger Ansporn während der Arbeit ; wir hören in uns den steten Zuruf : Da ist das Problem, suche die Lösung. Du kannst sie durch reines Denken finden ; denn in der Mathematik giebt es kein *Ignorabimus !*

8. Nous devons savoir, nous saurons.

FIGURE 1.1 – Stèle funéraire de Hilbert sise à Göttingen Stadtfriedhof.

— Cantor [9] qui donne une construction des réels, ressemblant à la construction de Méray [10] en utilisant les classes d'équivalence de suites de Cauchy de nombres rationnels et, surtout, la première démonstration, dans [7], que $\mathbb{R}$ est non-dénombrable. Plus tard dans [8, 9], il précise [11] que

> Par ensemble nous entendons toute collection M d'objets m de notre intuition ou de notre pensée, définis et distincts, ces objets étant appelés les éléments de M.

— Frege [12], un des fondateurs de la logique symbolique moderne, a défendu l'idée que les mathématiques sont réductibles à la logique. Dans [27], il introduit, en 1879, la notion de système logique avec négation, implication, quantification universelle et tables de vérité. Il a introduit une notation *ad hoc* pour préciser toutes les opérations logiques. Cependant cette notation est tombé en désuétude. Une table de correspondance avec la notation moderne est disponible sur la version allemande de Wikipedia, au lemme *Begriffsschrift*.

— Dedekind [13] qui propose, en 1872, une nouvelle construction de réels

9. Georg Ferdinand Ludwig Philipp Cantor (St Petersbourg 1845 – Halle 1918), a des contributions en théorie des nombres mais, à partir de 1870 il se tourne vers l'analyse.

10. Hugues Charles Robert Méray, (Chalon-sur-Saône 1835 – Dijon 1911), mathématicien français qui a donné en 1869 la première construction rigoureuse des nombres réels.

11. Unter einer ‚Menge' verstehen wir jede Zusammenfassung M von bestimmten wohlunterscheidbaren Objekten m unserer Anschauung oder unseres Denkens (welche die 'Elemente' von M genannt werden) zu einem Ganzen ([8, §1, p. 481]).

12. Friedrich Ludwig Gottlob Frege (Wismar, Mecklenburg-Schwerin 1848 – Bad Kleinen 1925). Ses travaux sur la philosophie de la logique, philosophie de mathématiques et philosophie du langage ont joué un rôle majeur dans nos conceptions modernes.

13. Julius Wilhelm Richard Dedekind (Braunschweig 1831 – Braunschweig 1916) a commencé comme enseignant de mathématiques dans plusieurs universités avant d'être re-

(qu'il a conçue en 1858) — connue de nos jours comme « coupure de Dedekind » — basée sur l'idée que tout nombre réel r sépare les rationnels en deux sous-ensembles, ceux inférieurs à r et ceux supérieurs à r. Dans [53], sont reproduits les deux textes majeurs de Dedekind [14] avec leur retranscription en langage moderne et sont mis dans le contexte du développement historique.

— Peano [15] propose une construction de $\mathbb{N}$ et introduit l'axiome d'induction.

Ces théories, n'étaient pas exemptes de contradictions mises en évidence par les logiciens de l'époque. Par exemple, la théorie de Cantor n'excluait pas l'existence d'un ensemble des ensembles. Cette hypothèse menait au **« paradoxe de Russell »** [16], découvert en 1901 et qui peut être formulé comme suit :

> Soit $E := \{x : x \text{ est un ensemble}\}$ et $R := \{x \in E : x \notin x\}$. R appartient-il dans R ? On voit que si $R \in R$ alors $R \notin R$ et si $R \notin R$ alors $R \in R$.

1.2.3 La théorie des ensembles après 1900 ; incomplétude de Gödel

Pour pallier ces pathologies, plusieurs tentatives d'une description axiomatique de la théorie des ensembles furent entreprises. Une première approche était initiée par Zermelo [17]. En 1904, Zermelo montre le théorème que tout ensemble peut-être bien ordonné [18] en utilisant l'axiome du choix [19] et en 1908 le premier système axiomatique des ensembles [78]. Cependant, le système de Zermelo ne permet pas de montrer l'existence de l'ensemble

cruté, en 1858, par l'École polytechnique fédérale de Zürich, où il fut recommandé comme un « excellent pédagogue » par Dirichlet. L'idée de la « coupure » lui est venue durant son séjour suisse, le 24 novembre 1858, lorsqu'il refléchissait sur la meilleure façon d'enseigner pour la première fois le cours de Calcul différentiel et intégral.

14. *Stetigkeit und irrationale Zahlen*, Braunscheig (1872) et *Was sind und was sollen die Zahlen ?*, Braunschweig (1888).

15. Giuseppe Peano, (Spinetta di Cuneo 1858 – Cavoretto 1932), mathématicien et linguiste italien. Outre l'axiomatisation de l'arithmétique, il a développé une langue (forme simplifiée, sans cas, du latin, appelée *Latino sine flexione*), dans laquelle il a rédigé plusieurs de ses œuvres, dont l'*Arithmetices principia*.

16. Bertrand Arthur William 3rd Earl Russell (Trellech, Monmouthshire 1872 – Penrhyndeudraeth, Caernarfonshire 1970), philosophe, logicien, mathématicien, historien, écrivain, critique social et activiste politique pacifiste britannique. Lauréat du Prix Nobel de littérature en 1950 *"in recognition of his varied and significant writings in which he champions humanitarian ideals and freedom of thought"*.

17. Ernst Zermelo (Berlin 1871 – Freiburg im Breisgau 1953) a commencé à travailler sous l'influence de Hilbert et en 1902 il publia son premier travail sur l'addition des nombres transfinis.

18. Un ensemble totalement ordonné X est bien ordonné (pour cet ordre) si toute partie non-vide de X possède un plus petit élément.

19. Pour toute collection U d'ensembles non vides, il existe une fonction définie sur U, appelée *fonction du choix*, qui à chaque ensemble $A \in U$ associe un élément de cet ensemble A

$\{Z_0, Z_1, Z_2, \ldots\}$, où $Z_0 := \mathbb{N}$ et pour $n \geq 1$, $Z_n = \mathcal{P}(Z_{n-1})$. Le système de Zermelo et Fraenkel[20] (avec — ZFC — ou sans — ZF — axiome du choix) étend le système de Zermelo et permet de traiter convenablement des collection qui ne contiennent que de purs ensembles.

Whitehead[21] et Russell entreprennent une nouvelle formulation de mathématiques dans les 3 volumes des *Principia Mathematica* [74, 75, 76] en introduisant la notion de type[22].

Mais, tout comme le système ZFC ne permet pas de montrer sa propre cohérence, l'entreprise de Whitehead et Russell sera aussi vouée à l'échec car Kurt Gödel[23] publia en 1931, à l'âge de 25 ans, le fameux article [28]. Ce travail fut traduit en anglais 3 fois ; Gödel — atteint de paranoïa à la fin de sa vie — renia les deux premières traductions comme trahissant sa pensée. Le lecteur non-germanophone peut consulter la traduction de 1962, reimprimée en 1992 [29]. Une autre source d'information est la traduction hypertextuelle et ré-écriture du théorème en notation moderne faite par Martin Hirzel. Le livre [65], consacré aux théorèmes de Gödel, offre une présentation très pédagogique — mais beaucoup plus longue — aux deux théorèmes d'incomplétude.

Pour expliquer la signification des théorèmes d'incomplétude, prenons le cas de l'arithmétique : il existe un entier, noté 0, et une application S, appelée « successeur de ». L'entier 1 est défini comme $S0$, l'entier 2 comme $SS0$, etc. La suite des successeurs continue pour toujours et ne reviennt jamais sur ses pas. Les entiers ainsi obtenus constituent l'ensemble $\mathbb{N}$. L'addition et la multiplication peuvent être facilement définies sur $\mathbb{N}$ ainsi que la relation d'ordre $\geq$.

Une fois les entiers ainsi déterminés, nous pouvons formuler des relations entre les entiers et souvent les démontrer. Par exemple, nous pouvons démontrer que pour deux entiers m et n, la formule $m + n = n + m$ est vraie. Il est plausible de supposer que toute affirmation concernant des entiers ait une réponse déterminé dans ce cadre. En d'autres termes, il est plausible de supposer que toute proposition formulée dans le langage de l'arithmétique est soit vraie soit fausse et la valeur de vérité découle des axiomes. Et ceci, même si nous ne connaissons pas de preuve. Nous nous attendons à ce que, si ϕ est une formule, soit ϕ découle des axiomes, et alors ϕ est vraie, soit $\neg\phi$ découle des axiomes, et alors ϕ est fausse. En logique, une théorie T est **complète par négation**, si pour toute phrase ϕ dans le langage T, soit ϕ, soit

20. Abraham Halevi (Adolf) Fraenkel (Münich 1891 – Jerusalem 1965), était un mathématicien israélien (né allemand) et fervent sioniste avec des contributions importantes en théorie des anneaux mais connu surtout pour ses travaux sur la théorie des ensembles [26].

21. Alfred North Whitehead (Ramsgate, Kent 1861 – Cambridge, MA 1947) était un mathématicien et philosophe britannique.

22. On peut trouver dans [77] une version plus accessible des articles originaux.

23. Kurt Gödel (né à Brünn en 1906 — jadis dans l'Empire austro-hongrois, aujourd'hui Brno en République Tchèque — mort à Princeton NJ en 1978). Logicien autrichien (naturalisé américain en 1947) de renom qui démontra entre autres les deux fameux théorèmes d'incomplétude. À la fin de sa vie, il fut atteint de paranoïa. Persuadé que l'on veuille l'empoisonner par sa nourriture, il refusa de s'alimenter et il est mort de mal-nutrition à l'hôpital de Princeton, en ne pesant que 45 kg.

$\neg\phi$ découle des axiomes. Une question naturelle donc est : l'arithmétique est-elle complète par négation ?

Dans ses théorèmes d'incomplétude, Gödel répond par la négative. Plus précisément,

Théorème 1.2.1 (Premier théorème d'incomplétude de Gödel)**.** *Toute théorie mathématique du premier ordre, cohérente*[24]*, récursivement axiomatisable*[25]*, et suffisamment riche pour contenir l'arithmétique est incomplète. C'est-à-dire, il existe une formule fermée G dans la théorie telle que ni G, ni $\neg G$ ne soit une conséquence des axiomes (i.e. un théorème) de la théorie.*

La preuve complète du premier théorème d'incomplétude est technique et fastidieuse. Cependant, en admettant certaines parties, on peut distiller une version simplifiée des idées dans la preuve qui permet d'en donner un aperçu très intuitif. Les ingrédients de la preuve (qui sera esquissée en §4.6) sont les suivants :

— on peut représenter toute preuve mathématique comme un mot appartenant à un langage (celui de la logique de 1er ordre) dénombrable,
— par conséquent, on peut énumérer les propositions de la théorie et
— utiliser la méthode de la diagonale de Cantor pour montrer qu'il existe des propositions indémontrables.

Pour l'information du lecteur, l'énoncé du deuxième théorème d'incomplétude de Gödel (non démontré dans ce cours) est aussi donné :

Théorème 1.2.2 (Second théorème d'incomplétude de Gödel)**.** *Si T est une théorie récursivement axiomatisable qui permet de formaliser l'arithmétique et si T démontre un énoncé exprimant qu'elle est cohérente, alors elle est contradictoire.*

1.3 Calculabilité

Une autre question fondamentale que l'on peut se poser est de savoir quelles fonctions (avec un ou plusieurs arguments entiers) sont vraiment calculables, i.e. leurs valeurs sont obtenues point par point par un procédé « mécanique » — aujourd'hui on dirait par un programme. Dans une lettre adressée en 1931 à Gödel, Herbrand[26] décrivit un mécanisme d'évaluation symbolique par valeur pour un modèle de calcul des systèmes d'équations. En 1934, Gödel présenta ces idées, qu'il a précisées entre temps, en leur donnant le nom de **fonctions récursives générales**.

24. Des propositions fausses ne peuvent pas être des théorèmes de la théorie.
25. Il est possible de reconnaître les axiomes de la théorie parmi les énoncés du langage de façon purement mécanique (i.e. automatique, algorthmique).
26. Jacques Herbrand, (Paris 1908 – Saint-Christophe-en-Oisans 1931), brillant mathématicien et philosophe français, reçu premier au concours de l'École normale supérieure en 1925, reçu premier à l'agrégation de mathématiques en 1928. Il est décédé à l'âge de 23 ans dans un accident de montagne, après avoir soumis son premier (et dernier) article sur *Le développement moderne de la théorie des corps algébriques : corps de classes et lois de réciprocité*, paru *post mortem*.

Church [27] montre que le *Entscheidungsproblem* est indécidable et développe le λ-calcul par lequel il montre qu'une fonction est calculable si elle est une fonction partielle sur les nombres de Church.

En 1936, Turing [28], avant d'avoir eu connaissance de travaux de Church, introduisit un moyen automatique — la machine qui porte son nom [29] — afin de montrer l'indécidabilité de certaines propositions [67]. Dans la démonstration (cf. §**??**), il utilise aussi la méthode de la diagonale de Cantor mais la formalisation et la preuve sont plus simples que celles de Gödel. Durant son séjour à Princeton, Turing se joint à l'équipe de Church et s'initie au λ-calcul et obtient une nouvelle preuve du résultat d'indécidabilité par le λ-calcul [68].

Un pas important vers la compréhension et l'unification des différentes approches est fait dans un théorème démontré par Church [14] et Turing [67] stipulant que les classes de fonctions

— partielles récursives (selon Herbrand-Gödel),
— Turing-calculables et
— λ-calculables

coïncident et sont les fonctions qu'intuitivement nous appelons **fonctions effectivement calculables** [30]. Ce résultat est souvent appelé **thèse de Church et Turing**, car tandis que l'équivalence entre les trois classes ci-dessus est démontré (il s'agit des théorèmes), la classe de fonctions effectivement calculable est décrite de façon non-formelle, par conséquent son équivalence avec les trois classes précitées ne peut pas être démontrée.

1.4 La logique comme langage

Déjà dans la démonstration de son théorème d'incomplétude, Gödel [28] utilise le fait que les **formules bien formées** peuvent être vues comme formant un langage. Turing donne une autre démonstration du théorème d'icnomplétude en introduisant la notion de machine de Turing qui est une forme d'automate. Ces idées furent approfondies et développées par la suite.

27. Alonzo Church, (Washington DC 1903 – Hudson OH,1995), est un mathématicien et logicien américain à qui l'on doit certains des fondements de l'informatique théorique. Dans sa thèse, il étudie les conséquences qu'aurait dans la formulation de la logique l'hypothèse que l'axiome du choix serait faux. De 1929 à 1967 il enseigne à l'université de Princeton, où échange avec von Neumann et encadre des étudiants brillants comme : Kleene, Rosser et Turing.

28. Alan Mathison Turing, (Londres 1912 – Wilmslow 1954). Mathématicien et cryptologue britannique. Ses travaux appliqués ont permis de casser le cryptage — effectué avec la redoutable machine « Enigma » — des communications de sous-marins allemands durant la deuxième guerre mondiale. Ses travaux théoriques ont permis de formaliser la notion d'algorithme en introduisant le concept de « machine » dont il est l'éponyme.

29. Les machines de Turing et leur application pour démontrer l'indécidabilité de certaines propositions seront présentées en chapitre 6.

30. Une source plus accessible où les articles originaux [14] et [67] sont reproduits (et commentés) est le recueil [19, pp. 108 et 119].

Kleene[31] a fondé la théorie de la récursion comme branche de la logique mathématique. Il a en outre introduit le concept d'expression régulière et de langage régulier [35] et il a montré l'équivalence, connue comme théorème de Kleene, entre langages réguliers et langages reconnus par des automates finis.

Les langages réguliers sont un cas particulier de la classe plus vaste des **langages formels** — des sous-ensembles particuliers de mots sur un alphabet fini spécifiés par un mécanisme fini (ex. les expression régulières, les automates, les grammaires formelles, etc.). L'article [48] est généralement cité comme le travail initiateur des la notion d'automate. Cependant, ce travail traite de ce que nous appelons aujourd'hui un réseau neuronal. Les **automates finis** déterministes ont été introduits par plusieurs auteurs dans [33, 49, 51] ; les automates non-déterministes dans [56].

Les **grammaires** génératives ont été développées par Chomsky[32] dans [11, 12] comme une tentative de formalisation des langages naturels. Si cette approche pour les langages naturels reste inachevée, elle trouve toute son utilité pour en théorie des langages formels, où il a établi l'**hiérarchie** — dite de Chomsky — établissant que chaque type spécifique de grammaire formelle engendre un type spécifique de langage qui est reconnu par un type spécifique d'automate.

Le **but de ce cours** est

— de rappeler une notion fondamentale pour ce cours, celle de graphe dirigé (considéré comme représentation d'une relation),
— de donner une introduction succincte des grammaires formelles et de l'hiérarchie de Chomsky selon leur expressivité,
— d'introduire les automates comme un système dynamique (chaîne de Markov déterministe) sur un ensemble fini d'états,
— pour arriver à la notion de machines de Turing et formuler la calculabilitié et l'*Entscheidungsproblem* comme un problème d'arrêt de trajectoires calculatoires d'une chaîne de Markov sur un graphe dirigé,
— et d'introduire les différentes classes de complexité algorithmique.

Comme application de ces notions, nous étudierons le problème de complexité algorithmique d'une suite aléatoire et présenterons la notion de **complexité de Kolmogorov**[33] [38] et son lien avec l'**entropie**.

31. Stephen Cole Kleene, (Hartford CT 1909 – Madison WI 1994), est un mathématicien et logicien américain. Il fut élève de Church et contribua de manière significative aux travaux de l'équipe.

32. Noam Chomsky, (né à Philadelphie PA, en 1928), est un linguiste et mathématicien, actuellement professeur émérite de linguistique au Massachusetts Institute of Technology, fondateur de la linguistique générative.

33. Andrei Nikolaevitch Kolmogorov, (Tambov 1903 – Moscou 1987), était un mathématicien soviétique qui a apporté des contributions significatives en mathématiques, notamment en théorie des probabilités, topologie, turbulence, mécanique classique, logique intuitionniste, théorie algorithmique de l'information et en analyse de la complexité des algorithmes.

2
Relations et graphes

Dans ce chapitre nous rappelons quelques notions élémentaires sur les relations et introduisons les structures combinatoires (graphes, digraphes, arbres) qui seront utiles dans la suite du cours. Les références de base pour ce chapitre sont les livres [44, 24, 43].

2.1 Rappels sur le produit cartésien

Si A et B sont deux ensembles, nous définissons le **produit cartésien** de A par B, l'ensemble des couples

$$A \times B = \{(a, b), a \in A, b \in B\},$$

où, pour $a, a' \in A$ et $b, b' \in B$,

$$[(a, b) = (a', b')] \Leftrightarrow [(a = a') \wedge (b = b')].$$

La cohérence de la définition impose $[A \times B = \emptyset] \Leftrightarrow [(A = \emptyset) \vee (B = \emptyset)]$. Il est évident que $|A \times B| = |A| \cdot |B| = |B \times A|$ mais $B \times A \neq A \times B$ en général. Le produit cartésien s'étend à un nombre arbitraire d'ensembles facteurs : si $(A_k)_{k=1,\ldots,n}$ sont des ensembles arbitraires,

$$A_1 \times \cdots \times A_n = \{(a_1, \ldots a_n) : a_i \in A_i, i = 1, \ldots, n\}, n \geq 1.$$

En particulier, si tous les A_i sont égaux à A, nous écrirons A^n pour le produit cartésien $\underbrace{A \times \cdots \times A}_{n \text{ termes}}$, pour $n \geq 1$, contenant les suites d'éléments de A de longueur n. Nous étendons la définition à $n = 0$ pour signifier $A^0 := \{()\}$ l'ensemble contenant la suite vide (de longueur 0). Nous noterons $A^+ = \cup_{n \geq 1} A^n$ et $A^* = \cup_{n \in \mathbb{N}} A^n = A^0 \cup A^+$ l'ensemble de suites d'éléments de

A de longueur arbitraire (finie). $A^{\mathbb{N}} := \{(a_1, a_2, \ldots) : a_i \in A, i \in \mathbb{N}\}$ pour l'ensemble des suites infinies d'éléments de A.

Dans la suite, $\mathbb{A}$ sera un ensemble dénombrable, le plus souvent fini. Nous l'appellerons alors **alphabet**; les éléments de $\mathbb{A}^*$ seront identifiés à des **mots** construits sur l'alphabet. Si $\alpha \in \mathbb{A}^*$, alors il existe un unique entier $n \geq 0$ tel que $\alpha \in \mathbb{A}^n$, le nombre n est alors appelé longueur du mot α, noté $|\alpha| = n$. Pour des mots, nous écrirons le plus souvent $\alpha = \alpha_1 \cdots \alpha_n$ au lieu de $\alpha = (\alpha_1, \ldots, \alpha_n)$. Si $|\alpha| = 0$, alors α désignera le mot vide qui sera noté ε ou $(\,)$.

Nous pouvons munir $\mathbb{A}^*$ d'une opération interne

$$\mathbb{A}^* \times \mathbb{A}^* \ni (\alpha, \beta) \mapsto \alpha\beta \in \mathbb{A}^*,$$

appelée **concaténation**. Si $\alpha = \alpha_1 \cdots \alpha_{|\alpha|}$ et $\beta = \beta_1 \cdots \beta_{|\beta|}$ sont deux mots de longueur $|\alpha|$ et $|\beta|$ respectivement, la concaténation $\alpha\beta = \gamma$ est un mot γ de longueur $|\alpha| + |\beta|$, dont les $|\alpha|$ premières lettres sont celles du mot α et les $|\beta|$ dernières lettres celles du mot β, i.e. $\gamma = \alpha_1 \cdots \alpha_{|\alpha|}\beta_1 \cdots \beta_{|\beta|}$. L'opération de concaténation peut être naturellement étendue sur $\mathbb{A}^* \times \mathbb{A}^{\mathbb{N}}$ (mais pas sur $\mathbb{A}^{\mathbb{N}} \times \mathbb{A}^*$ ou $\mathbb{A}^{\mathbb{N}} \times \mathbb{A}^{\mathbb{N}}$).

Définition 2.1.1. Si $\mathbb{A}$ est un alphabet dénombrable, l'ensemble $\mathbb{A}^*$ muni de l'opération de concaténation est un monoïde ayant le mot vide comme élément neutre, i.e. pour des mots $\alpha, \beta, \gamma \in \mathbb{A}^*$,

— $(\alpha\beta)\gamma = \alpha(\beta\gamma)$.

— $\varepsilon\alpha = \alpha\varepsilon = \alpha$.

L'ensemble $\mathbb{A}^*$ est appelé **clôture monoïdale** de $\mathbb{A}$.

Remarque 2.1.2. Si $\mathbb{X}$ est un ensemble arbitraire, la notation $\mathbb{X}^*$ désignera *toujours* l'ensemble $\mathbb{X}^* = \cup_{n\in\mathbb{N}}\mathbb{X}^n$, où la suite $(\mathbb{X}^n)_{n\in\mathbb{N}}$ est définie d'une certaine manière. En particulier, si $\mathbb{A}$ est un alphabet dénombrable, l'ensemble $\mathbb{A}^*$ est sa clôture monoïdale définie ci-dessus. En cohérence avec cette convention, $\mathbb{N}^*$ désigne l'ensemble des suites d'entiers de longueur (finie) arbitraire. L'ensemble $\{1, 2, \ldots\}$ d'entiers strictement positifs sera noté $\mathbb{N}_>$ dans ce cours.

2.2 Relations

Intuitivement une relation entre deux ensembles d'objets signifie que pour un caractère donné, certains éléments du premier ensemble sont en correspondance avec certains du second. Par exemple si F désigne l'ensemble de femmes et H celui des hommes, un élément $h \in H$ sera en relation avec un élément $f \in F$ si h est fils biologique de f. Alors la relation « être fils biologique de » est une relation de H vers F définie par les couples (h, f) qui la vérifient.

Définition 2.2.1. Soient $\mathbb{A}$ et $\mathbb{B}$ deux ensembles non-vides. On appelle **relation** de $\mathbb{A}$ vers $\mathbb{B}$ une partie $R \subseteq \mathbb{A} \times \mathbb{B}$. Si $(a, b) \in R$, nous dirons que

a est en relation avec b (nous écrirons parfois aRb). Si $R \subset \mathbb{A}^2$, la relation R est appelée relation sur $\mathbb{A}$. Si $|\mathbb{A}| \neq |\mathbb{B}|$, la relation est dite inhomogène ; dans le cas contraire, $\mathbb{B}$ peut-être identifié à $\mathbb{A}$ et alors la relation est dite homogène.

Dans ce cours, nous étudierons principalement des relations entre ensembles dénombrables (finis ou infinis). Nous disposerons alors de plusieurs *représentations équivalentes* d'une relation $R \subseteq \mathbb{A} \times \mathbb{B}$.

Exhaustive : par la liste de toutes les paires $(a, b) \in R$. Si les ensembles $\mathbb{A}$ et $\mathbb{B}$ sont considérés comme parties de $\mathbb{R}$, identifiés alors aux axes d'abscisses et d'ordonnées du plan $\mathbb{R}^2$, les couples $(a, b) \in R$ peuvent être vus comme les points du « graphe » de R. On parle alors de représentation **cartésienne** de R.

Matricielle : par une matrice booléenne $M := M(R) \in \mathbb{M}_{|\mathbb{A}|\times|\mathbb{B}|}(\{0, 1\})$, avec

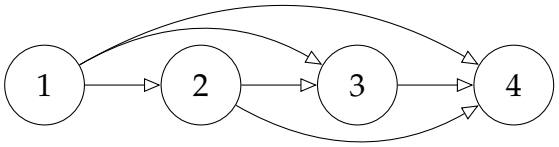
$$M(R)_{ab} = \begin{cases} 1 & \text{si } (a, b) \in R \\ 0 & \text{sinon.} \end{cases}$$

Le **domaine** de R, noté $\mathsf{dom}(R)$, est l'ensemble des indices des lignes non identiquement nulles ; l'**image** de R, notée $\mathsf{im}(R)$, est l'ensemble des indices des colonnes non identiquement nulles.

Sagittale : par un ensemble de sommets $\mathbb{A} \cup \mathbb{B}$ et des flèches [1] reliant $a \in \mathbb{A}$ à $b \in \mathbb{B}$ si, et seulement si, $(a, b) \in R$. Le domaine de R est l'ensemble de tous les sommets de $\mathbb{A}$ d'où émane une flèche, l'image de R est l'ensemble de sommets de $\mathbb{B}$ où arrive une flèche.

Exemple 2.2.2. Soient $\mathbb{A} = \mathbb{B} = \{1, 2, 3, 4\}$ et $R = \{(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)\}$. Ses représentations matricielle et sagittale sont :

$$M(R) = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} ;$$



Nous avons $\mathsf{dom}(R) = \{1, 2, 3\}$ et $\mathsf{im}(R) = \{2, 3, 4\}$. Vous aurez sûrement reconnu que $(a, b) \in R$ signifie que a « est strictement plus petit que » b.

Définition 2.2.3. Soit $R \subseteq \mathbb{A} \times \mathbb{B}$ une relation arbitraire.

1. On appelle **relation réciproque**, notée R^{-1}, la relation définie par
$$(a, b) \in R^{-1} \Leftrightarrow (b, a) \in R,$$
composée de couples réciproques de ceux appartenant à R.
2. On appelle **relation complémentaire**, notée R^c (ou parfois $\overline{R}$), la relation définie par
$$(a, b) \in R^c \Leftrightarrow (a, b) \notin R,$$
i.e. $R^c = A \times B \setminus R$.

1. De ces flèches provient l'adjectif sagittal, du latin *sagitta* n.f. flèche.

3. Si $\mathbb{B} = \mathbb{A}$, on appelle **relation identique**, notée $\mathbb{I}_{\mathbb{A}}$ la relation définie par
$$(a, a') \in \mathbb{I}_{\mathbb{A}} \Leftrightarrow (a = a'),$$
ou encore par la diagonale du produit cartésien : $\mathbb{I}_{\mathbb{A}} = \{(a, a), a \in \mathbb{A}\}$.

Remarque 2.2.4. Nous avons $\mathsf{dom}(R^{-1}) = \mathsf{im}(R)$ et $\mathsf{im}(R^{-1}) = \mathsf{dom}(R)$. En représentation saggitale, la relation R^{-1} s'obtient de R en renversant toutes les flèches. Par ailleurs, $\mathsf{dom}(\mathbb{I}_{\mathbb{A}}) = A = \mathsf{im}(\mathbb{I}_{\mathbb{A}})$ et $\mathbb{I}_{\mathbb{A}}^{-1} = \mathbb{I}_{\mathbb{A}}$.

Définition 2.2.5.
1. Si R est une relation sur $\mathbb{A}$ et $\mathbb{B} \subseteq \mathbb{A}$, on appelle **restriction** de R sur $\mathbb{B}$, la relation $R\restriction_{\mathbb{B}} = R \cap (\mathbb{B} \times \mathbb{B})$.
2. Si R et S sont deux relations de $\mathbb{A}$ vers $\mathbb{B}$, leur **disjonction** est la relation $R \cup S$, définie par la condition $(a, b) \in R \cup S \Leftrightarrow [(a, b) \in R] \vee [(a, b) \in S]$. De même, leur **conjonction** est la relation $R \cap S$, définie par la condition $(a, b) \in R \cap S \Leftrightarrow [(a, b) \in R] \wedge [(a, b) \in S]$.
3. Si R est une relation de $\mathbb{A}$ vers $\mathbb{B}$ et S une relation de $\mathbb{B}$ vers $\mathbb{D}$ alors leur **composition** est la relation $S \circ R$, définie par la condition
$$[(a, d) \in S \circ R] \Leftrightarrow [\exists b \in B : (a, b) \in R \wedge (b, d) \in S].$$

Il est facile de voir que $M(R \cup S) = M(R) \vee M(S)$ où le second membre est calculé par disjonction booléenne — élément par élément — des matrices représentant R et S. De même $M(R \cap S) = M(R) \wedge M(S)$.

Proposition 2.2.6. *Si R est une relation de $\mathbb{A}$ vers $\mathbb{B}$ et S une relation de $\mathbb{B}$ vers $\mathbb{D}$, alors*
$$M(S \circ R)_{a,d} = (M(R) \odot M(S))_{a,d} := \bigvee_{b \in B} M(R)_{a,b} \wedge M(S)_{b,d}.$$

Démonstration. Notons $X = M(R)$, $Y = M(S)$ et $Z = M(S \circ R)$. Alors, pour $a \in \mathbb{A}$ et $d \in \mathbb{D}$,
$$\begin{aligned} Z_{a,d} = 1 &\Leftrightarrow (a, c) \in S \circ R \\ &\Leftrightarrow [\exists b \in B : [(a, b) \in R] \wedge [(b, d) \in S]] \\ &\Leftrightarrow [\exists b \in B : [X_{a,b} = 1] \wedge [Y_{b,d} = 1]] \\ &\Leftrightarrow \bigvee_{b \in B} [X_{a,b} \wedge Y_{b,d} = 1]. \end{aligned}$$

□

Proposition 2.2.7. *Soient des relations $R \subseteq \mathbb{A} \times \mathbb{B}$, $S \subseteq \mathbb{B} \times \mathbb{D}$ et $T \subseteq \mathbb{D} \times \mathbb{E}$. Alors*
1. $T \circ (S \circ R) = (T \circ S) \circ R$ *(associativité),*
2. $\mathbb{I}_B \circ R = R = R \circ \mathbb{I}_{\mathbb{A}}$ *(unités gauche et droite),*
3. $(S \circ R)^{-1} = R^{-1} \circ S^{-1}$.

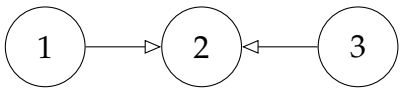
Démonstration. La démonstration est une application directe des définitions. À titre d'exemple, nous utilisont la représentation matricielle pour établir l'affirmation 3.

$$\begin{aligned} M((S\circ R)^{-1}) &= M(S\circ R)^t = (M(R)\odot M(S))^t \\ &= M(S)^t \odot M(R)^t = M(S^{-1})\odot M(R^{-1}) \\ &= M(R^{-1}\circ S^{-1}). \end{aligned}$$

□

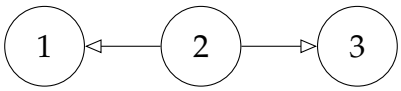
Exercice 2.2.8. Justifier toutes les étapes intermédiaires dans la démonstration de l'affirmation 3. de la proposition 2.2.7 présentée ci-dessus et compléter la démonstration des affirmations 1. et 2.

Exemple 2.2.9. Soient $\mathbb{A} = \{1,2,3\}$ et R la relation décrite en représentations sagittale et matricielle par



$$M(R) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

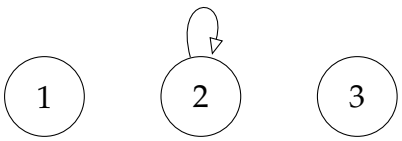
Alors R^{-1} est représentée dans les mêmes représentations par



$$M(R^{-1}) = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

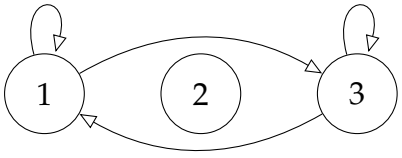
Nous aurons alors $M(R\circ R^{-1} = M(R^{-1})\odot M(R)$, i.e.

$$M(R\circ R^{-1}) = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \odot \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$



et $M(R^{-1}\circ R) = M(R)\odot M(R^{-1})$, i.e.

$$M(R^{-1}\circ R) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \odot \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix};$$



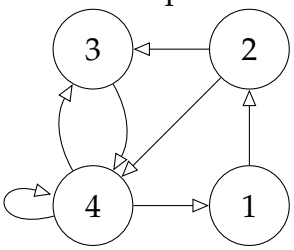
Définition 2.2.10. Soit $R \subseteq \mathbb{A}^2$ une relation sur A. La relation est dite

- **réflexive** si $\forall a \in \mathbb{A} \Rightarrow (a,a) \in R$;
- **anti-réflexive** si $\forall a \in \mathbb{A} \Rightarrow (a,a) \in R^c$;
- **symétrique** si $a,b \in \mathbb{A} : (a,b) \in R \Rightarrow (b,a) \in R$ (i.e. $R = R^{-1}$);
- **anti-symétrique** si $[(a,b) \in R] \wedge [(b,a) \in R] \Rightarrow a = b$ (i.e. $R \cap R^{-1} \subseteq \mathbb{I}_{\mathbb{A}}$) ou encore $[a \neq b] \Rightarrow [(a,b) \in R^c] \vee [(b,a) \in R^c]$.

Remarque 2.2.11. L'anti-réflexivité n'est pas la négation de réflexivité. R est réflexive si dans sa représentation sagittale, chaque sommet a une boucle ; R est anti-réflexive si aucun sommet n'a de boucle. L'anti-symétrie n'est pas la négation de la symétrie. R est symétrique si toutes les arêtes sont bi-directionnelles ; R est anti-symétrique si aucune arête n'est bi-directionnelle.

Définition 2.2.12. Soient $R \subseteq \mathbb{A}^2$ une relation sur $\mathbb{A}$, $a, b \in \mathbb{A}$ et $n \in \mathbb{N}$. On appelle **R-chemin** de longueur n de a à b toute suite $\mathbf{x} = (x_0, \ldots, x_n) \in \mathbb{A}^{n+1}$ de $n+1$ éléments de $\mathbb{A}$ tels que $x_0 = a, x_n = b$ et $(x_i, x_{i+1}) \in R$, pour tout $i = 0, \ldots, n-1$. Si R est donnée par sa représentation sagittale, un R-chemin de longueur n est une trajectoire de n arêtes consécutives sur la représentation de R. Si $a = b$ et $\mathbf{x}$ est un R-chemin de a à b, alors $\mathbf{x}$ est un **cycle**. Un R-chemin de a à b dont tous les sommets (à l'exception peut-être de a et b) sont distincts est appelé **sans recoupement**. Un chemin sans recoupement qui passe par tous les sommets d'un graphe est appelé **hamiltonien**.

Exemple 2.2.13. Soit R la relation de représentation sagittale



Le chemin 124344 est de longueur 5. Le chemin 41234 est un cycle hamiltonien de longueur 4.

Théorème 2.2.14. *Soient $R \subseteq \mathbb{A}^2$ une relation sur $\mathbb{A}$ et $a, b \in \mathbb{A}$. S'il existe un R-chemin fini de a à b alors il existe un R-chemin sans-recoupement de a à b.*

Démonstration. Supposons que $\mathbf{x}$ est un R-chemin avec recoupement de a à b, i.e. $\mathbf{x} = (a \equiv x_0, \ldots x_n \equiv b)$ avec $n > 1$. Puisque $\mathbf{x}$ est avec recoupement, il existe i, j avec $0 \leq i < j \leq n-1$ tels que $x_i = x_j$. Le chemin $\mathbf{x}$ se décompose alors en $\mathbf{x} = \boldsymbol{\zeta\kappa\xi}$ où $\boldsymbol{\zeta} = (a \equiv x_0, \ldots, x_{i-1}, x_i)$, $\boldsymbol{\kappa} = (x_i, \ldots, x_j)$ et $\boldsymbol{\xi} = (x_i \equiv x_j, x_{j+1}, \ldots x_n \equiv b)$. Il est clair que $\boldsymbol{\zeta\xi}$ est un R-chemin de a à b qui ne contient pas le cycle $\boldsymbol{\kappa}$. Puisque $i < j$ la longueur du cycle $\boldsymbol{\kappa}$ est strictement positif ; le chemin $\boldsymbol{\zeta\xi}$ est donc strictement plus court que $\mathbf{x}$. Si $\boldsymbol{\zeta\xi}$ est sans recoupement, le théorème est montré. Sinon, on recommence ; comme la longueur initiale du chemin est finie, l'effacement des cycles intermédiaires s'arrêtera au bout d'un temps fini. □

Si R est une relation sur $\mathbb{A}$, nous pouvons la composer avec elle-même pour former $R \circ R := R^2$ (remarquer que R^2 peut être vide) et, par récurrence, définir R^n pour tout $n \geq 1$. Pour des raisons pratiques, nous définissons aussi $R^0 = \mathbb{I}_{\mathbb{A}}$. Nous avons $(a, b) \in R^n \Leftrightarrow \exists R$-chemin de longueur n de a à b et nous pouvons alors définir $R^+ = \cup_{n \geq 1} R^n$ et $R^* = \cup_{n \geq 0} R^n = \mathbb{I}_{\mathbb{A}} \cup R^+$.

2.3 Graphes dirigés

Définition 2.3.1. On appelle **graphe dirigé ou digraphe** le quadruplet $\mathbb{G} = (\mathbb{G}^0, \mathbb{G}^1, s, t)$, où $\mathbb{G}^0$ est un ensemble dénombrable de sommets, $\mathbb{G}^1$ un ensemble dénombrable d'arêtes et s et t deux fonctions $\mathbb{G}^1 \to \mathbb{G}^0$ associant à chaque arête $\alpha \in \mathbb{G}^1$ son sommet source $s(\alpha)$ (d'où elle émane) et son sommet terminus $t(\alpha)$ (où elle arrive). Pour tout $x \in \mathbb{G}^0$ on définit le **degré sortant** de x, l'entier $d_x^- = \mathsf{card}\{\alpha \in \mathbb{G}^1 : s(\alpha) = x\}$ et le **degré entrant** de x, l'entier $d_x^+ = \mathsf{card}\{\alpha \in \mathbb{G}^1 : t(\alpha) = x\}$.

Remarque 2.3.2. Soient $\alpha, \beta \in \mathbb{G}^1$ des arêtes et $x, y \in \mathbb{G}^0$ des sommets.
- Si $s(\alpha) = t(\alpha) = x \in \mathbb{G}^0$ on dit que l'arête est une **boucle** sur le sommet x.
- Si pour deux sommets $x, y \in \mathbb{G}^0$ (éventuellement identiques) il existe deux arêtes distinctes α et β ayant $s(\alpha) = s(\beta) = x$ et $t(\alpha) = t(\beta) = y$, alors on dit que le sommet x est relié à y par des **arêtes multiples**.
- Si $s(\alpha) = t(\beta)$ et $s(\beta) = t(\alpha)$ alors les arêtes sont appelées **opposées** et on note $\alpha = \overline{\beta}$.

Si l'application $(s, t) : \mathbb{G}^1 \to \mathbb{G}^0 \times \mathbb{G}^0$ est injective, alors l'ensemble des arêtes $\mathbb{G}^1$ peut être identifié à une partie $R := (s, t)(\mathbb{G}^1) \subseteq \mathbb{G}^0 \times \mathbb{G}^0$ du produit cartésien des sommets. $\mathbb{G}^1$ définit alors une relation sur $\mathbb{G}^0$. Dans ce cas, les fonctions s et t deviennent superflues car l'ensemble des arêtes est identifiée avec cette relation. Réciproquement, toute relation R sur $\mathbb{G}^0$ définit un graphe dirigé, $\mathbb{G}(R)$, sans arêtes multiples, appelé graphe de la relation R. Si la relation R est symétrique, toutes les arêtes sont bi-directionnelles ; on peut alors parler de graphe (non dirigé) de R en identifiant les paires d'arêtes $(\alpha, \overline{\alpha})$ en position tête-bêche entre deux sommets donnés avec une arête non-dirigée, notée encore α, entre les même sommets.

Définition 2.3.3. Soit $\mathbb{G} := (\mathbb{G}^0, \mathbb{G}^1, s, t)$ un graphe dirigé. Pour $n \geq 1$, nous définissons l'ensemble de **trajectoires** de longueur n du graphe [2]

$$\mathbb{G}^n := \{\alpha = \alpha_1 \cdots \alpha_n \in (\mathbb{G}^1)^n : s(\alpha_{i+1}) = t(\alpha_i), \text{ pour } i = 1, \ldots, n-1\}$$

et $\mathbb{G}^* = \cup_{n \geq 0} \mathbb{G}^n$ l'ensemble de trajectoires de longueur arbitraire [3]. Les fonctions s et t — initialement définies sur $\mathbb{G}^1$ — sont naturellement étendues sur $\mathbb{G}^*$:
- si $\alpha = v \in \mathbb{G}^0$ (i.e. $|\alpha| = 0$), alors $s(\alpha) = t(\alpha) = v$;
- Si $|\alpha| \geq 2$ et $\alpha = \alpha_1 \cdots \alpha_{|\alpha|}$, avec $\alpha_i \in \mathbb{G}^1$ pour $i = 1 \ldots |\alpha|$, alors $s(\alpha) = s(\alpha_1)$ et $t(\alpha) = t(\alpha_{|\alpha|})$.

Si $v \in \mathbb{G}^0$, on note $\mathbb{G}_v^* := \{\alpha \in \mathbb{G}^0 : s(\alpha) = v\}$, les trajectoires qui émanent du sommet v.

2. L'ensemble de trajectoires de longueur 0 sera identifié avec l'ensemble de sommets $\mathbb{G}^0$.

3. Noter qu'en général $\mathbb{G}^n \neq (\mathbb{G}^1)^n$.

Remarque 2.3.4. Étant donné que la construction des ensembles $\mathbb{G}^n$ et $\mathbb{G}^*$ à partir du graphe $\mathbb{G} = (\mathbb{G}^0, \mathbb{G}^1, s, t)$ est canonique, lorsque nous utiliserons dorénavant le symbole $\mathbb{G}$, nous sous-entendrons que toute la suite $(\mathbb{G}^n)_{n\geq 0}$ est donnée.

Remarque 2.3.5. Si $\mathbb{G}$ est un digraphe, $\mathbb{G}^*$ *n'est pas un monoïde* car si $\alpha, \beta \in \mathbb{G}^*$, le produit par concaténation $\alpha\beta$ n'est défini que si (α, β) est une **paire composable**, i.e. $t(\alpha) = s(\beta)$. De même, l'unité n'est pas unique; chaque sommet $v \in \mathbb{G}^0$ joue le rôle d'une unité. Un ensemble avec ces propriétés est appelé **semi-groupoïde**.

Définition 2.3.6. Soit R une relation sur $\mathbb{A}$; on note $\mathbb{G} := \mathbb{G}(R)$ le graphe dirigé la représentant. L'ensemble $\mathbb{A}$ (ou de manière équivalente le digraphe $\mathbb{G}$) est dit R**-connexe** si

$$\forall (a, b) \in \mathbb{G}^0 \times \mathbb{G}^0 \Leftrightarrow \exists \alpha \in \mathbb{G}^* : s(\alpha) = a; t(\alpha) = b.$$

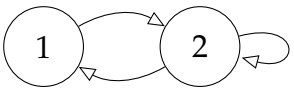
Exercice 2.3.7. Considérer la relation R dont le digraphe (représentation sagittale) $\mathbb{G} := \mathbb{G}(R)$ est celui de l'exemple 2.2.2.

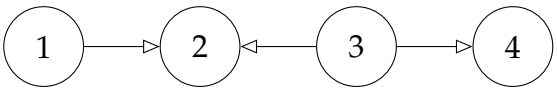
1. Déterminer $\mathbb{G}^*$.
2. $\mathbb{G}$ est-il R-connexe ?

Définition 2.3.8. Soit R une relation sur $\mathbb{A}$. La relation est dite

1. **transtitive** si $[(a, b) \in R \wedge (b, c) \in R] \Rightarrow [(a, c) \in R]$ ou encore $R^2 \subseteq R$,
2. **circulaire (ou cyclique)** si $[(a, b) \in R \wedge (b, c) \in R] \Rightarrow [(c, a) \in R]$ ou encore $R^2 \subseteq R^{-1}$.

Exemple 2.3.9. Soient R et S les relations définies par leurs digraphes :

$\mathbb{G}(R) =$ 

$\mathbb{G}(S) =$ 

1. R n'est ni transitive ni circulaire car $(1, 2) \in R$ et $(2, 1) \in R$ mais $(1, 1) \notin R$.
2. S est transitive et circulaire car $S^2 = \emptyset$ donc $S^2 \subseteq S$ et $S^2 \subseteq S^{-1}$.

Il est rappelé qu'une relation R sur $\mathbb{A}$
— antisymétrique, i.e. $[(a, b) \in R] \wedge [(b, a) \in R] \Rightarrow [a = b]$ (ou $R \cap R^{-1} \subseteq \mathbb{I}_{\mathbb{A}}$) et
— transitive, i.e. $[(a, b) \in R] \wedge [(b, c) \in R] \Rightarrow [(a, c) \in R]$ (ou $R^2 \subseteq R$)
est

1. un **ordre partiel** sur $\mathbb{A}$ si elle est réflexive, i.e. $\forall a, (a, a) \in R$ (ou $\mathbb{I}_{\mathbb{A}} \subseteq R$),
2. un **ordre strict** sur $\mathbb{A}$ si elle est anti-réflexive, i.e. $\forall a, (a, a) \notin R$ (ou $\mathbb{I}_{\mathbb{A}} \cap R = \emptyset$).

Théorème 2.3.10. *Soit R une relation sur $\mathbb{A}$. Alors les trois propriétés suivantes sont équivalentes :*

1. *R est transitive,*
2. *$\forall n \geq 1 : R^n \subseteq R$,*
3. *$R^+ = R$.*

Démonstration. $2 \Rightarrow 1$: La transitivité de R découle immédiatement en prenant $n = 2$.

$1 \Rightarrow 2$: Supposons R transitive et montrons par récurrence que pour tout $n \geq 1 : R^n \subseteq R$. En effet, l'inclusion est triviale pour $n = 1$; supposons-la vraie pour un $n > 1$ et considérons $\alpha = ax_1 \dots x_n b$ un R-chemin de longueur $n+1$ de a à b. Mais alors $\beta = ax_1 \dots x_n$ est un R-chemin de longueur n de a à x_n ce qui signifie que $(a, x_n) \in R^n \subseteq R$ (par l'hypothèse de récurrence). Par ailleurs $(x_n, b) \in R$ et R est transitive. Par conséquent $R^{n+1} \subseteq R$. Ceci montre que $\forall n \geq 1 : R^n \subseteq R$].

$2 \Rightarrow 3$: Puisque $R^+ = R \cup R^2 \cup R^2 \cup \dots$, il s'ensuit que $R \subseteq R^+$. Par ailleurs $R^n \subseteq R$, pour tout $n \geq 1$; par conséquent $R^+ \subseteq R$.

$3 \Rightarrow 1$: Immédiate car $R^2 \subseteq R^+ = R$.

□

Si R est une relation arbitraire sur $\mathbb{A}$, en général elle ne possède pas de propriété particulière telle que symétrie, réflexivité, transitivité, circularité, etc. En notant $\mathcal{P}$ une quelconque de ces propriétés, nous souhaitons adjoindre à R les couples de $\mathbb{A}^2$ qui sont indispensables pour qu'elle acquière la propriété $\mathcal{P}$.

Définition 2.3.11. Une relation S sur $\mathbb{A}$ est dite la **$\mathcal{P}$-clôture** de R si

1. S a la propriété $\mathcal{P}$,
2. $R \subseteq S$ et
3. pour toute relation T sur $\mathbb{A}$, on a :

$$[T \text{ a la propriété } \mathcal{P}] \wedge [R \subseteq T] \Rightarrow [S \subseteq T].$$

Remarque 2.3.12. La $\mathcal{P}$-clôture de R est la « plus petite » relation ayant la propriété $\mathcal{P}$. Une relation a la propriété $\mathcal{P}$ si, et seulement si, R est la $\mathcal{P}$-clôture de R.

Théorème 2.3.13. *Si R est une relation sur $\mathbb{A}$, alors sa clôture*

1. *réflexive est $R \cup \mathbb{I}_{\mathbb{A}}$,*
2. *symétrique est $R \cup R^{-1}$ et*
3. *transitive est R^+.*

Démonstration. La preuve des affirmations 1 et 2 est laissée en exercice.

La remarque liminaire pour montrer 3 est que R^+ est une relation transitive car la composition de R-chemins et un R-chemin. Supposons que T est une relation transitive arbitraire contenant R et considérons, pour un $n \geq 1$, un R-chemin $\alpha = x_0 \dots x_n$. L'inclusion $R \subseteq T$ implique que pour tous les $i = 0, \dots, n-1$ on a $(x_i, x_{i+1}) \in T$. Puisque T est transitive, nous aurons

que $(x_0, x_n) \in T$. Ceci montre que $R^n \subseteq T$, pour tout $n \geq 1$, par conséquent $R^+ \subseteq T$. Donc R^+ est la plus petite relation transitive contenant R ; elle est appelée la **clôture transitive** de R. □

Remarque 2.3.14. $R^* = \mathbb{I}_{\mathbb{A}} \cup R^+$ est la **relation d'accessibilité**[4] de R. Si R est symétrique, alors R^* est une équivalence. En effet,

— R^* est réflexive car R l'est,
— R^* est symétrique car R l'est,
— R^* est transitive car $R^+ \subseteq R^*$.

La partition de $\mathbb{A}$ en classes d'équivalence de R^* est l'ensemble des composantes R-connexes de $\mathbb{A}$.

Relations usuelles

Éléments	Ensembles	Signification
Réflexive		
$\forall a \in \mathbb{A}, aRa$	$\mathbb{I}_{\mathbb{A}} \subseteq R$	chaque sommet a une boucle
Antiréflexive		
$\forall a \in \mathbb{A}, aR^c a$	$\mathbb{I}_{\mathbb{A}} \cap R = \emptyset$	aucun sommet n'a de boucle
Symétrique		
$aRb \Leftrightarrow bRa$	$R = R^{-1}$	toutes les arêtes sont bi-directionnelles
Anti-symétrique		
$aRb \wedge bRa \Rightarrow a = b$	$R \cap R^{-1} \subseteq \mathbb{I}_{\mathbb{A}}$	les seules arêtes bi-directionnelles sont les boucles
Transitive		
$aRb \wedge bRc \Rightarrow aRc$	$R^2 \subseteq R$	s'il existe de chemin indirect de a vers c il existe aussi un chemin direct
Cyclique		
$aRb \wedge bRc \Rightarrow cRa$	$R^2 \subseteq R^{-1}$	on peut fermer tout chemin de a vers c par une arête (c, a)
Acyclique		
$aRb \wedge bRc \Rightarrow cR^c a$	$R^+ \cap \mathbb{I}_{\mathbb{A}} = \emptyset$	ne contient aucun cycle.

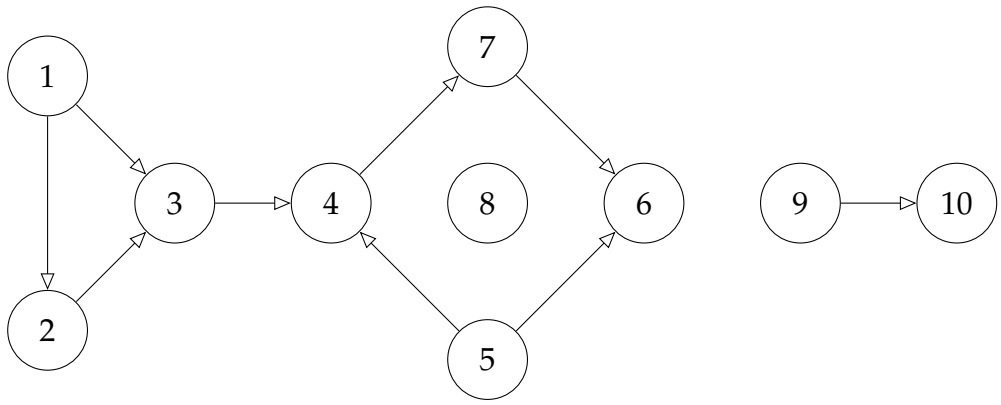
2.4 Arbres

Définition 2.4.1. Une relation R sur $\mathbb{A}$ est **acyclique** si R ne contient aucun cycle, i.e. sa clôture transitive est anti-réflexive.

Soit $\mathbb{G} := \mathbb{G}(R)$ le digraphe d'une relation acyclique R sur un ensemble $\mathbb{A}$ (i.e. $\mathbb{A} = \mathbb{G}^0$). On appelle **base** de $\mathbb{A}$, notée $\mathbb{B}$, l'ensemble des sommets $a \in \mathbb{A} : d_a^- = 0$ et **feuillage** de A, noté $\mathbb{F}$, l'ensemble des sommets $a \in \mathbb{A} : d_a^+ = 0$ (noter que si card$\mathbb{A}$ est fini, ces ensembles ne sont pas vides).

4. Nous avons rencontré déjà cette notion dans le contexte des chaînes de Markov (cf. cours *Probabilités pour la théorie de l'information*).

Exemple 2.4.2. Soit R une relation dont le digraphe est donné par

$\mathbb{G}(R) =$ 

Alors R est acyclique avec $\mathbb{B} = \{1, 5, 8, 9\}$ et $\mathbb{F} = \{6, 8, 10\}$. Noter que $\mathbb{B} \cap \mathbb{F}$ n'est pas nécessairement vide.

Soit $\mathbb{G}$ le digraphe d'une relation R. Si $(a, b) \in R$, il s'ensuit qu'il existe une arête $\alpha \in \mathbb{G}^1$ ayant $s(\alpha) = a$ et $t(\alpha) = b$; il est donc intuitivement clair pourquoi nous disons alors que a est un **successeur** de b ou que b est un **prédécesseur** de a. On note $\mathsf{succ}(a) = \{b \in \mathbb{A} : (a, b) \in R\}$ et $\mathsf{pred}(a) = \{b \in \mathbb{A} : (b, a) \in R\}$. Mais nous avons vu que les fonctions s et t peuvent être étendues sur $\mathbb{G}^*$; nous disons que a est un **ancêtre** de b ou qui b est un **descendant** de a si $(a, b) \in R^+$.

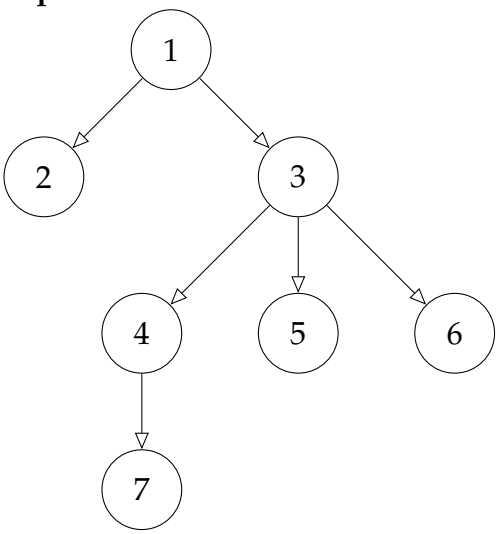
Il est facile de voir qu'une relation d'ordre strict est acyclique. Réciproquement si R est une relation d'ordre strict alors R^+ est acyclique (pourquoi ?) Pour vérifier donc qu'une relation est acyclique, il suffit de vérifier que sa clôture transitive est anti-réflexive.

Définition 2.4.3. Une relation T sur $\mathbb{A}$ est un **arbre enraciné** s'il existe un $r \in \mathbb{A}$ tel que

— pour tout $a \in \mathbb{A} \setminus \{r\}$ il existe un *unique* T-chemin de r vers a et
— $(r, r) \notin T^+$.

L'élément r s'appelle **racine** de l'arbre et un élément quelconque a **nœud** de l'arbre. On note (r, T) pour l'arbre enraciné à r.

Exemple 2.4.4. Voici un arbre ! Il a $\mathbb{A} = \{1, \ldots, 7\}$ et $r = 1$.



Théorème 2.4.5. *Si (r, T) est un arbre sur $\mathbb{A}$, alors*

1. *la relation T est acyclique,*
2. *r est l'unique racine,*
3. *$\mathsf{pred}(r) = \emptyset$ et*
4. *pour tout $a \in \mathbb{A} \setminus \{r\}$, on a $\mathsf{card}\,\mathsf{pred}(a) = 1$.*

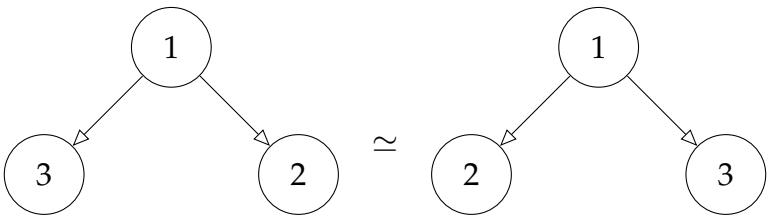
Démonstration. Nous montrons toutes les affirmations par l'absurde.

1. Supposons que α_1 soit un T-cycle de a vers a (avec a quelconque). Comme $(r, r) \notin T^+$, il s'ensuit que $a \neq r$; par conséquent, il existe un unique T-chemin $\alpha_2 \in T^+$ de r vers a. Dans ce cas, α_2 et $\alpha_2\alpha_1$ sont deux T-chemins distincts de r vers a ce qui contredit la définition.
2. Supposons qu'il existe une autre racine $r' \neq r$. Il existe alors un chemin α de r à r' (car r est racine) et un chemin β de r' à r (car r' est racine). Alors $\alpha\beta$ est un cycle de r vers r, en contradiction avec le fait $(r, r) \notin T^+$.
3. Si $a \in \mathsf{pred}(r)$, il existe un chemin α (de longueur 1) de a vers r et un T-chemin de β de r vers a (car r est une racine). Alors $\beta\alpha$ est un cycle de r vers r, ce qui est impossible.
4. Supposons que $b, c \in \mathsf{pred}(a)$ et $b \neq c$. Alors $\beta_1 = (b, a)$ est un T-chemin de longueur 1 de b vers a et $\beta_2 = (c, a)$ un T-chemin de longueur 1 de c vers a. Mais il existe aussi un T-chemin α_1 de r vers b et un T-chemin α_2 de r vers c. Alors $\alpha_1\beta_1$ et $\alpha_2\beta_2$ sont deux chemins distincts de r vers a, en contradiction avec l'unicité imposée par la définition.

□

Étant donné qu'un arbre T est une relation sur $\mathbb{A}$, nous pouvons définir toutes les notions précédentes telles que base, feuillage, ancêtres, descendants, etc. Mais T est acyclique et son unique racine n'a pas de prédécesseur. Les nœuds de l'arbre peuvent être regroupés en niveaux. Le niveau 0, note T_0, contient uniquement la racine. Les autres niveaux sont définis par récurrence ; pour $n \geq 1$, nous définissons $T_n = \{a \in \mathbb{A} : \mathsf{pred}(a) \subseteq T_{n-1}\}$. Un arbre peut avoir un nombre infini de niveaux mais s'il est fini, on appelle **hauteur de l'arbre** le plus grand n tel que $T_n \neq \emptyset$.

Les nœuds d'un arbre à un niveau donné ne sont pas structurés. Les digraphes



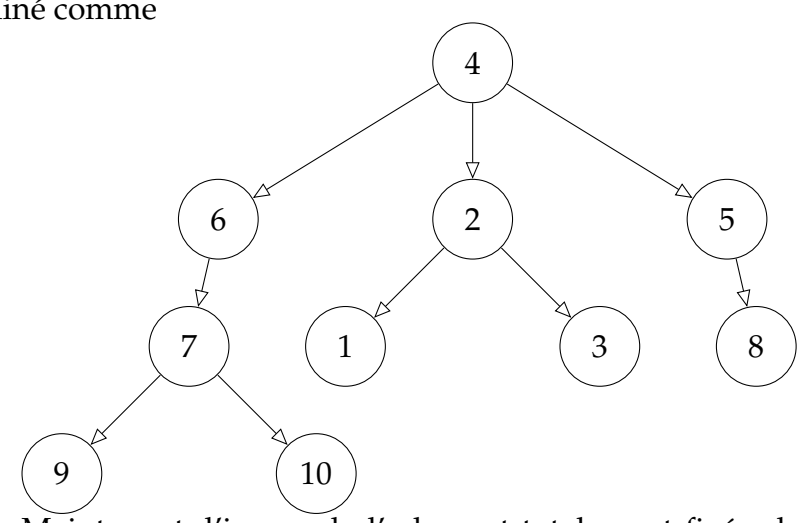
représentent le même arbre. Il est parfois commode de les ordonner ; les deux arbres précédents seront alors considérés comme différents. La structure ainsi obtenue est appelée **arbre ordonné**. Une représentation de T en

tant qu'arbre ordonné sera sous forme d'ensemble de listes ordonnées de couples ; chaque couple sera de la forme (parent, enfant) et l'ordre dans la liste désignera un ordre dans les enfants de chaque nœud.

Exemple 2.4.6. Soit T l'arbre *ordonné* représenté par

$$T = \{((4,6),(4,2),(4,5)),((5,8)),((6,7)),((2,1),(2,3)),((7,9),(7,10))\}.$$

Alors son digraphe (en tant qu'arbre ordonné) est alors uniquement déterminé comme



Maintenant, l'image de l'arbre est totalement figée ; les sous-arbres ne peuvent plus pivoter autour des arêtes qui le relient à des sommets du niveau inférieur. (Le niveau inférieur d'un niveau donné de l'arbre, se situe immédiatement au dessus dans la convention de dessiner les arbres vers le bas).

Définition 2.4.7. Soient T un arbre sur $\mathbb{A}$ et $v \in \mathbb{G}^0(T)$ un nœud quelconque de T.

1. On définit

   $$B = \{x \in \mathbb{A} : (v,x) \in T^*\} = \{v\} \cup \{\text{tous les descendants de } v\}.$$

   Alors $T(v) = T \cap B^2$ est un arbre sur B de racine v. Il est appelé **sous-arbre** de T dominé par v.
2. Soit $d \in \mathbb{N}_>$. On dit que T est un
   — d-arbre si $\operatorname{card}\operatorname{succ}(v) \leq d$ et
   — d-arbre complet si $\operatorname{card}\operatorname{succ}(v) = d$.

Le cas $d = 2$ des arbres binaires est particulièrement intéressant en informatique. On peut aussi donner une définition récursive d'un arbre binaire.

Définition 2.4.8. Un arbre T sur $\mathbb{A}$ est binaire si
— ou bien il est vide,
— ou alors il est égal à un triplet ordonné $T = (r, T_g, T_d)$, où r est sa racine et T_g, T_d sont deux arbres binaires, appelés sous-arbres respectivement gauche ou droit de T.

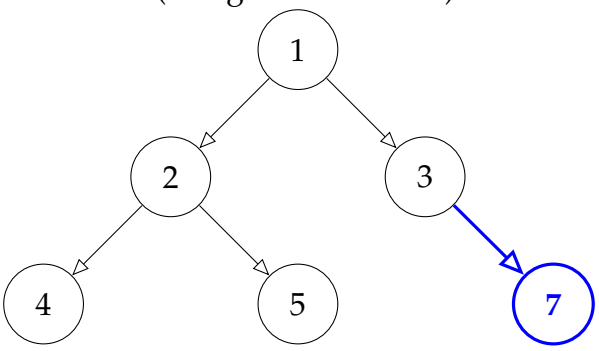
Supposons que T est un d-arbre (non-nécessairement complet) sur un ensemble $\mathbb{A}$ ordonné. Cet arbre peut être obtenu d'un d-arbre ordonné complet $\hat{T}$ en effaçant certaines branches (sous-arbres dominés). On note $\iota : \mathbb{G}^0(T) \hookrightarrow \mathbb{G}^0(\hat{T})$ l'application d'inclusion des sommets de T dans ceux de $\hat{T}$. Maintenant, l'arbre $\hat{T}$ étant complet et ordonné, chaque $v \in \mathbb{G}^0(\hat{T})$ a d successeurs ordonnés, notés $s_{v1}, \dots s_{vd}$. L'arbre T est obtenu de l'arbre $\hat{T}$ en effaçant tous les sommets $v \in \mathbb{G}^0(\hat{T})$ pour lesquels $\iota^{-1}(v) = \emptyset$ ainsi que tous les successeurs de v mais en gardant la même ossature. L'arbre T est alors appelé **arbre positionnel** (cf. exemple 2.4.10).

Définition 2.4.9. Soit T un arbre sur $\mathbb{A}$. Sa clôture symétrique S s'appelle **arbre non-dirigé** sur $\mathbb{A}$.

Exemple 2.4.10. Soit l'arbre binaire ordonné complet $\hat{T}$ sur $\mathbb{A} = \{1, \dots, 7\}$, donné en description exhaustive ordonnée par

$$\hat{T} = \{((1,2),(1,3)),((2,4),(2,5)),((3,6),(3,7))\}.$$

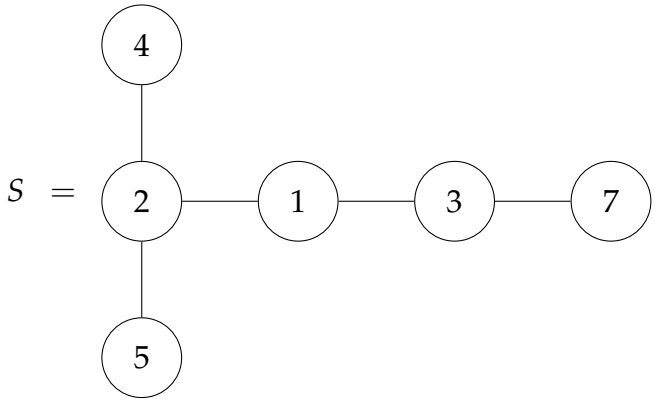
Supposons que le sommet 6 manque de l'arbre T (ainsi que l'arête le reliant à son parent 3 qui devrait être une arête « gauche »). Cependant, nous gardons mémoire du fantôme de l'arête manquante et nous représentons l'arête (3,7) comme une arête « droite » (cf. figure ci-dessous) dans l'arbre T.



Ce même arbre T peut aussi être décrit sous sa forme récursive

$$T = (1, \underbrace{(2,(4,\emptyset,\emptyset),(5,\emptyset,\emptyset))}_{T_g}, \underbrace{(3,\emptyset,(7,\emptyset,\emptyset))}_{T_d}).$$

Sa clôture symétrique est l'arbre non-dirigé



Nous finissons ce chapitre introductif avec une autre caractérisation des arbres non-dirigés. Soit R une relation sur $\mathbb{A}$ et $\alpha = a_0 \dots a_n$, $a_i \in \mathbb{A}$ un R-chemin arbitraire. On dit que $\alpha = a_0, \dots, a_n$ est un R**-chemin simple** si pour tout $i, j \in \{1, \dots, n-1\}$, nous avons $(a_i \neq a_j) \vee (a_{i-1} \neq a_{j+1})$. Pour mieux comprendre la notion, il est plus facile d'appréhender sa négation : un R-chemin est non-simple s'il existe $i, j \in \{1, \dots, n-1\}$ tels que $(a_i = a_j) \wedge (a_{i-1} = a_{j+1})$, i.e. s'il existe une arête qui est traversée dans les deux sens.

Si R est une relation symétrique sur $\mathbb{A}$, elle est acyclique si, et seulement si, elle est vide. Nous introduisons donc la notion moins contraignante : R est **simplement acyclique** si elle ne contient aucun cycle simple. Une relation simplement acyclique est automatiquement anti-réflexive car tout cycle de longueur 1 est simple. Nous sommes maintenant en mesure de donner une autre caractérisation d'un arbre non-dirigé.

Théorème 2.4.11. *Soit R une relation symétrique sur un ensemble $\mathbb{A}$ fini avec* $\mathsf{card}\mathbb{A} > 1$. *Les deux conditions suivantes sont équivalentes.*

1. *R est un arbre non-dirigé sur $\mathbb{A}$.*
2. *R est connexe et simplement acyclique.*

Démonstration. Sera rédigée ultérieurement. □

2.5 Exercices

Relations et digraphes

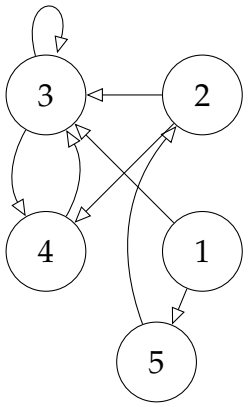
1. Que peut-on conclure sur A et B si $(A \times B) \cap (B \times A) \neq \emptyset$?
2. Soient A, B, C ensembles, parties d'un univers E. Comparer
   — $A \times (B \cup C)$ et $(A \times B) \cup (A \times C)$.
   — $A \times (B \cap C)$ et $(A \times B) \cap (A \times C)$.
   — $A \times (B \setminus C)$ et $(A \times B) \setminus (A \times C)$.
3. Monter que $(X \subseteq A) \wedge (Y \subseteq B) \Rightarrow [X \times Y = (A \times Y) \cap (X \times B)]$.
4. Soient A, B deux parties d'un univers E. Notons $\mathbb{I}_E = \{(x, x) : x \in E\}$. Montrer que
$$[A \cap B = \emptyset] \Leftrightarrow [(A \times B) \cap \mathbb{I}_E = \emptyset].$$
5. Soit R une relation de $\mathbb{A}$ vers $\mathbb{B}$ (finis) et $M := M(R)$ la matrice booléenne qui la représente. Calculer $M(R^{-1})$, $M(R^c)$ et $M(\mathbb{I}_{\mathbb{A}})$.
6. Pour chacune des relations R, S et T définies sur $\mathbb{R}$ par
   — $xRy \Leftrightarrow (y = |x|)$,
   — $xSy \Leftrightarrow (|x| + |y| = 1)$ et
   — $xTy \Leftrightarrow (y|x| = 1)$,
   déterminer son domaine et son image et donner sa représentation cartésienne et celle de sa réciproque.

7. Soient R et S les relations sur $\mathbb{A} = \{1,2,3,4\}$ dont les matrices sont

$$M(R) := \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad M(S) := \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}.$$

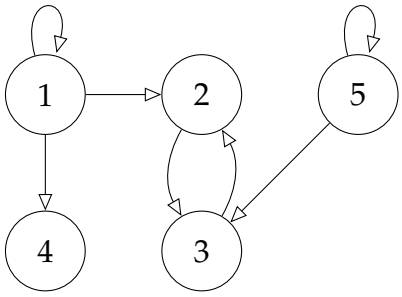
   Représenter leurs digraphes et déterminer leurs domaines et images.

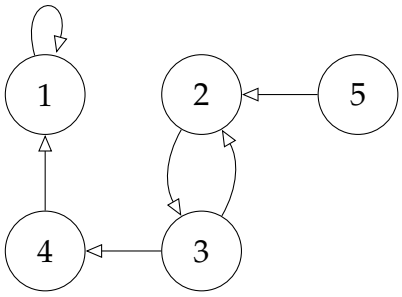
8. Représenter matriciellement la relation R décrite par le digraphe



   et déterminer son domaine et son image.

9. Soient $\mathbb{A} = \{1,2,\ldots,5\}$ et R et S des relations sur $\mathbb{A}$ décrites respectivement par le digraphe gauche et droit





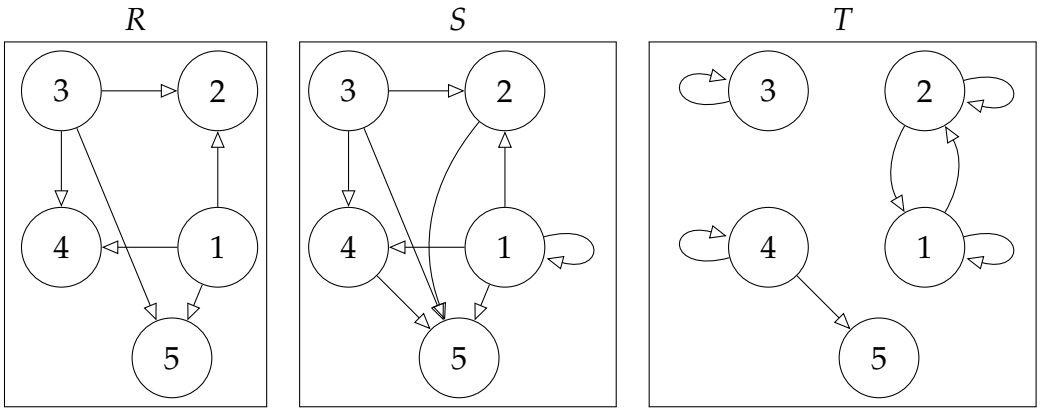
   Déterminer les digraphes des relations $R \cap S$ et $R \cup S$.

10. Soient R et S des relations de $\mathbb{A}$ vers $\mathbb{B}$.

    (a) Montrer que
    — $R \subseteq S \Rightarrow R^{-1} \subseteq S^{-1}$,
    — $(R \cup S)^{-1} = R^{-1} \cup S^{-1}$,
    — $R \subseteq S \Rightarrow S^c \subseteq R^c$.

    (b) Comparer
    — $\mathsf{dom}(R \cap S)$ et $\mathsf{dom}(R) \cap \mathsf{dom}(S)$,
    — $\mathsf{im}(R \cap S)$ et $\mathsf{im}(R) \cap \mathsf{im}(S)$.

11. Soient R, S et T des relations sur $\mathbb{A} = \{1,2,\ldots,5\}$ définies par leurs digraphes



Sont-elles, réflexives, anti-réflexives, symétriques, anti-symétriques, transitives, circulaires ?

12. Le graphe de Herschel est le graphe biparti à 11 sommets (5 de couleur bleue et 6 de couleur rouge) et à 18 arêtes bi-directionnelles — chacune d'elles reliant des sommets de couleurs différentes. Le graphe est homéomorphe à la sphère $\mathbb{S}^2$, donc il est planaire. La figure 2.1 donne les représentations dans l'espace et dans le plan de ce graphe. Montrer que ce graphe n'admet pas de circuit hamiltonien.

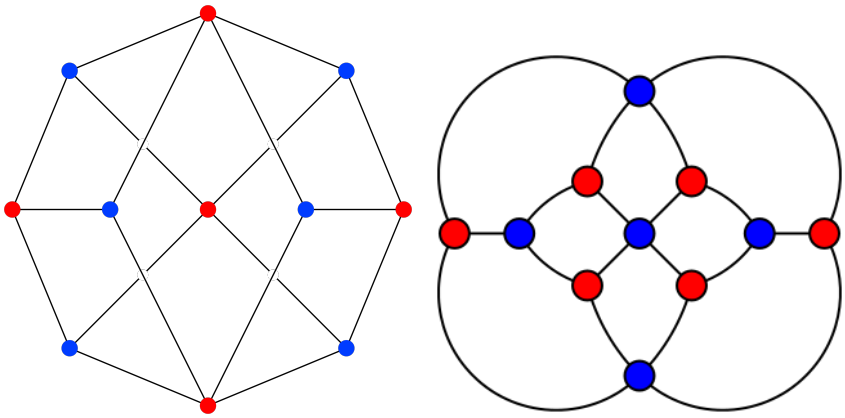


FIGURE 2.1 – Les représentations spatiale et planaire du graphe de Herschel.

13. Montrer que
    (a) $(a,b) \in R^+ \Leftrightarrow \exists R$-chemin de longueur strictement positive de a à b,
    (b) $(a,b) \in R^* \Leftrightarrow \exists R$-chemin de longueur positive de a à b ou encore
    (c) $(a,b) \in R^* \Leftrightarrow [[(a=b)] \vee [(a,b) \in R^+]]$.
14. [Extrait du contrôle du 8 mars 2022]. Le but de cet exercice est de montrer le

    Théorème : Soient $\mathbb{A}$ un alphabet avec $|\mathbb{A}| = p$, pour un $p \in \mathbb{N}_>$ et R une relation sur $\mathbb{A}$. Alors

$$R^+ = R \cup R^2 \cup \ldots \cup R^p.$$

(a) Donner un argument pour justifier qu'il suffise de montrer l'inclusion $R^+ \subseteq R \cup R^2 \cup \ldots \cup R^p$.

(b) Soient $a, b \in \mathbb{A}$ et α un R-chemin de a à b. En invoquant un résultat du cours, expliquer *en une phrase* pourquoi on peut, sans perte de généralité, supposer que α est sans recoupement.

(c) Considérer deux sommets $a, b \in \mathbb{A}$ tels que aR^+b. En distinguant deux cas ($a \neq b$ et $a = b$) obtenir une majoration sur la longueur $|\alpha|$ du chemin. Conclure.

Arbres, clôture

15. Parmi les relations R définies sur $\mathbb{A}$, repérer les arbres :

(a) $\mathbb{A} = \{1, \ldots, 5\}$ et $R = \{(1,2), (2,4), (1,3), (2,5), (3,4)\}$,
(b) $\mathbb{A} = \{1, \ldots, 4\}$ et $R = \{(1,2), (3,4), (1,3), (4,1)\}$,
(c) $\mathbb{A} = \{1, \ldots, 6\}$ et $R = \{(2,3), (3,4), (4,6), (3,5), (3,1)\}$,
(d) $\mathbb{A} = \{1, \ldots, 6\}$ et $R = \{(1,2), (1,3), (1,4), (1,5), (6,5)\}$.

16. Soit T un arbre sur $\mathbb{A}$. Monter que

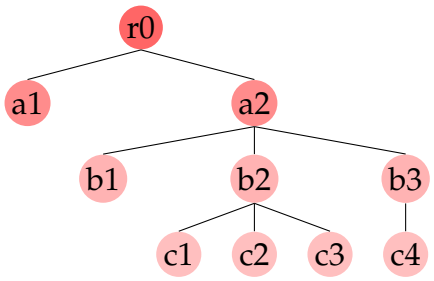
$$\forall a, b, c \in T : (aTb) \wedge (bTc) \Rightarrow (aT^c c).$$

17. Soit R une relation de $\mathbb{A}$. Montrer que si R^+ est un ordre strict, alors R est acyclique.

18. Représenter l'arbre ordonné décrit par l'ensemble des listes de couples :

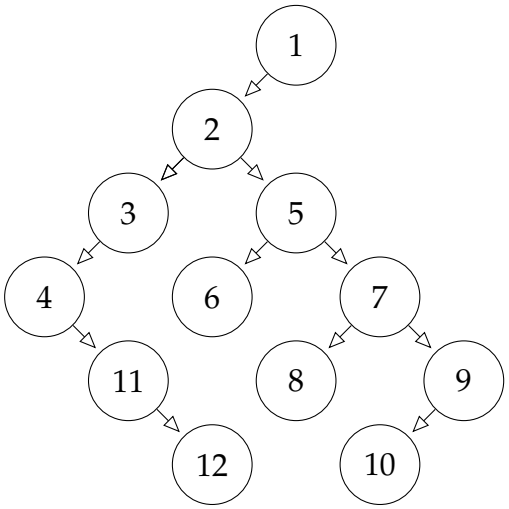
$$T = \{((3,8)), ((4,1),(4,7)), ((2,4),(2,3),(2,5)), ((5,9),(5,6))\}.$$

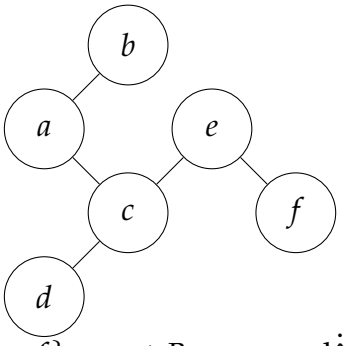
19. Soit T l'arbre ordonné ci-dessous :



Représenter son arbre binaire positionnel $B(T)$.

20. Reconstruire l'arbre T à partir de son arbre binaire positionnel $B(T)$ donné ci-dessous :



21. Soit l'arbre non-dirigé R représenté par 

Représenter tous les arbres sur $\mathbb{A} = \{a, \ldots, f\}$ ayant R comme clôture symétrique.
22. Soit R une relation sur $\mathbb{A}$ et soient $a, b \in \mathbb{A}$. Montrer que

$$(a \neq b) \wedge (aR^+b) \Rightarrow (\exists R - \text{chemin simple de } a \text{ vers } b).$$

23. Donner un exemple de relation R sur A telle que, pour un $a \in \mathbb{A}$,
— aR^+a et
— il n'existe pas de cycle simple en a.

3

Langages et grammaires formels

Les notions développées dans ce chaptre ont été introduites dans [12] — initialement pour décrire des langages naturels — et développées ultérieurement dans le cadre des langages formels ; nous suivons essentiellement [31]. Des références plus facilement accessibles sont [50, 63].

3.1 Motivation

Si $\mathbb{A}$ est un ensemble fini, certains éléments de sa clôture monoïdale $\mathbb{A}^*$ peuvent avoir une signification particulière. Par exemple si $\mathbb{A} = \{a, \ldots, z, \text{␣}\}$, où ␣ est le caractère « blanc », une classe bien particulière de $\mathbb{A}^*$ contient toutes les phrases possibles et imaginables du français, une autre toutes les phrases de l'anglais, etc. Ainsi, nous appellerons généralement **langage** toute partie de $\mathbb{A}^*$.

Deux questions vont nous préoccuper dans ce cours.

Comment engendrer le langage à partir d'un nombre restreint de règles ? Il n'est pas nécessaire d'avoir appris toutes les phrases possibles et imaginables du français ; avec les connaissances acquises pendant la période de son apprentissage, nous pouvons construire *n'importe quelle phrase* grammaticalement correcte en français. Nous disposons donc d'un système générateur, appelé **grammaire**, qui nous permet d'engendrer le langage.

Comment reconnaître qu'une phrase donnée appartient au langage ? Si vous voyez la phrase *"Zij $\omega(\cdot)$ een lineaire afbeelding van een reële Hilbertruimte $\mathcal{K}$ naar de ruimte van stochasten op een kansruimte met eindige momenten"*, votre système de reconnaissance linguistique du français la rejette immédiatement ; sans l'ombre d'un doute elle n'est pas du français. Nous appelons **automate** (ou **machine**) un dispositif in-

formatique (ou un algorithme) faisant automatiquement cette reconnaissance. Si on présente une phrase en entrée de l'automate, après un nombre fini d'étapes intermédiaires, il s'arrête en donnant la réponse « oui » ou « non » selon que la phrase appartient ou non au langage.

Dans ce cours, nous ne nous intéresserons pas aux langages naturels (français, anglais, musique, etc.) mais aux langages formels (programmes informatiques, expressions mathématiques, etc.) qui ont une structure syntaxique plus stricte et peuvent de ce fait être étudiés plus facilement. Ce chapitre traitera du problème de génération par une grammaire et le chapitre 4 est une application des grammaires à la logique et la construction d'une preuve mathématique. Le chapitre 5 traite le problème de reconnaissance d'un langage par un automate. Nous verrons qu'il existe une hiérarchie de grammaires, l'**hiérarchie de Chomsky** [12]; pour chaque type de grammaire dans cette hiérarchie correspondra un type d'automate permettant de reconnaître les phrases produites par la grammaire.

3.2 Grammaires formelles

Nous avons déjà introduit la notion de clôture monoïdale $\mathbb{A}^*$ d'un alphabet $\mathbb{A}$ dans le paragraphe §2.1. On complète la définition par la

Proposition 3.2.1. *L'ensemble $\mathbb{A}^*$ (resp. $\mathbb{A}^+$), muni de l'opération interne de concaténation est*
- *un monoïde (resp. semigroupe),*
- *possédant la propriété de simplification bilatère.*
- *En outre, $|\alpha\beta| = |\alpha| + |\beta|$, pour tout couple $\alpha, \beta \in \mathbb{A}^*$.*

Démonstration. Exercice. □

On rappelle que l'unité du monoïde $\mathbb{A}^*$ est le mot vide, noté ε et que la propriété de simplification bilatère, pour $\alpha, \beta, \gamma, \delta \in \mathbb{A}^*$, signifie que

$$[\alpha\beta = \alpha\gamma] \Rightarrow [\beta = \gamma] \text{ et } [\alpha\beta = \gamma\beta] \Rightarrow [\alpha = \gamma].$$

3.2.1 Aspects combinatoires de $\mathbb{A}^*$

Si S est un semigroupe (resp. monoïde) et $\mathbb{A} \subseteq S$ alors $\mathbb{A}$ est contenu dans au moins un sous-semigroupe (resp. sous-monoïde) de S (en l'occurrence dans S lui-même). Par conséquent, $\mathbb{A}^+$ peut-être identifié avec le semigroupe

$$\cap\{\mathbb{B} : \mathbb{B} \text{ sous-semigroupe de } S, \mathbb{A} \subseteq \mathbb{B}\} \neq \emptyset.$$

Il est donc le plus petit semi-groupe contenant $\mathbb{A}$.

Si $S = \mathbb{A}^+$ (reps. $S = \mathbb{A}^*$), alors $\mathbb{A}$ est dit **ensemble générateur** de S.

Définition 3.2.2. Soit $\mathbb{A}$ un ensemble générateur du semi-groupe (resp. monoïde) S. Alors S est dit **libre** sur $\mathbb{A}$ si chaque élément de S a une représentation unique comme produit d'éléments de $\mathbb{A}$.

Remarque 3.2.3. L'ensemble[1] $\mathbb{A}^*$ est libre sur $\mathbb{A}$. Ainsi, si $\alpha, \beta \in \mathbb{A}^*$, avec $\alpha = a_1 \cdots a_m$, $a_i \in \mathbb{A}$ pour $i = 1, \ldots, m$ et $\beta = b_1 \cdots b_n$, $b_j \in \mathbb{A}$ pour $j = 1, \ldots, n$, nous avons

$$[\alpha = \beta] \Leftrightarrow [(m = n) \wedge (\forall i \in \{1, \ldots, n\}, a_i = b_i)].$$

Théorème 3.2.4. *Soient $\alpha, \beta, \gamma, \delta \in \mathbb{A}^*$. Si $\alpha\beta = \gamma\delta$, alors*

1. *si $|\alpha| \geq |\gamma|$, alors $\exists! \zeta \in \mathbb{A}^*$ tel que $\alpha = \gamma\zeta$ et $\delta = \zeta\beta$;*
2. *si $|\alpha| = |\gamma|$, alors $\alpha = \gamma$ et $\delta = \beta$;*
3. *si $|\alpha| \leq |\gamma|$, alors $\exists! \zeta \in \mathbb{A}^*$ tel que $\gamma = \alpha\zeta$ et $\beta = \zeta\delta$;*

Démonstration. Exercice ! □

Définition 3.2.5. Soit $\alpha, \beta \in \mathbb{A}^*$. On écrit $\alpha \preceq \beta$ s'il existe $\gamma \in \mathbb{A}^*$ tel que $\alpha\gamma = \beta$. On dit alors que α est un **préfixe** de β ou que γ est un **suffixe** de β.

3.2.2 Grammaires

Définition 3.2.6. Soient

- $\mathbb{A}$ un alphabet fini non-vide et $\mathbb{A}_t, \mathbb{A}_n$ des parties complémentaires dans $\mathbb{A}$, appelées respectivement **symboles terminaux** et **non-terminaux** de $\mathbb{A} = \mathbb{A}_t \sqcup \mathbb{A}_n$,
- une partie *finie non-vide* $\Pi \subset \mathbb{A}^*\mathbb{A}_n\mathbb{A}^* \times \mathbb{A}^*$, appelée **productions**,
- S est un élément particulier de $\mathbb{A}_n$ appelé **axiome**.

Le quadruplet $\Gamma = (\mathbb{A}_t, \mathbb{A}_n, \Pi, S)$ est alors appelée **grammaire**.

Remarque 3.2.7. Il est donc évident que Π est une relation sur $\mathbb{A}^*$. Si $(\alpha, \beta) \in \Pi$, nous écrivons aussi $\alpha \to \beta$ au lieu de $\alpha\Pi\beta$ ou de $(\alpha, \beta) \in \Pi$.

Notation 3.2.8. Habituellement, les éléments de $\mathbb{A}_n$ sont notés par des lettres majuscules tandis que ceux de $\mathbb{A}_t$ par des lettres miniscules, des chiffres, d'opérations logique ou mathématiques, etc.

Définition 3.2.9. Soit $\Gamma = (\mathbb{A}_t, \mathbb{A}_n, \Pi, S)$ une grammaire avec $\mathbb{A} = \mathbb{A}_t \sqcup \mathbb{A}_n$. Cette grammaire induit sur $\mathbb{A}^*$ un digraphe $\mathbb{G} := \mathbb{G}(\Gamma) = (\mathbb{G}^0, \mathbb{G}^1, s, t)$, dit digraphe de **dérivation directe**, où $\mathbb{G}^0 = \mathbb{A}$ et $\mathbb{G}^1$ un ensemble d'arêtes dirigées, notées $\alpha \blacktriangleright \beta$. La présence d'une arête $\alpha \blacktriangleright \beta$ signifie

$$[\alpha \blacktriangleright \beta] \Leftrightarrow [\exists \gamma_1, \gamma_2, \eta, \zeta \in \mathbb{A}^* : \alpha = \gamma_1\zeta\gamma_2, \beta = \gamma_1\eta\gamma_2, (\zeta, \eta) \in \Pi].$$

On note, comme d'habitude, $\blacktriangleright^+ = \cup_{n \geq 1} \blacktriangleright^n$ et $\blacktriangleright^* = \cup_{n \geq 0} \blacktriangleright^n$ les chemins de longueur arbitraire sur le digraphe.

1. Il est souligné que l'ensemble $\mathbb{A}^*$ ne contient que de l'information de bas niveau (combinatoire). Des informations géométriques ou analytiques peuvent être adjointes à $\mathbb{A}^*$ pour obtenir des objets plus compliqués tels que des groupes ou des systèmes de fonctions itérées.

Remarque 3.2.10. Les productions induisent une arête entre les mots α et β dont α contient nécessairement au moins un symbole non terminal. Le terme « production » sous-entend que si le mot α est un sous-mot d'un mot autorisé (i.e. accessible à partir de S par les règles grammaticales) $\zeta = \zeta_1\alpha\zeta_2$ et (α, β) une production, alors $\zeta_1\beta\zeta_2$ sera aussi un mot autorisé par la grammaire. Par exemple, supposons que $\mathbb{A}_n = \{A, B, C\}$, $\mathbb{A}_t = \{x, y, z\}$ et que $(zBx, zCCxx) \in \Pi$. Si le mot $\zeta = xyAzBxyz$ est autorisé, alors le mot $\zeta' = xyAzCCxxyz$ est aussi autorisé (i.e. accessible à partir de S). L'ensemble des trajectoires $\mathbb{G}_S^*$ du digraphe $\mathbb{G} = \mathbb{G}(\blacktriangleright)$ émanant de l'axiome S contiendra tous les mots qui peuvent être produits par la grammaire à partir de l'axiome.

Remarque 3.2.11. Il est souligné que le digraphe des dérivations directes $\mathbb{G} := \mathbb{G}(\Gamma)$ provenant d'une grammaire Γ peut avoir des arêtes mutliples. Par exemple si $\pi_1 = aS \to ac$ et $\pi_2 = Sb \to cb$ sont des productions, alors $aSb \blacktriangleright_{\pi_1} acb$ et $aSb \blacktriangleright_{\pi_2} acb$ sont deux arêtes distinctes du graphe $\mathbb{G}(\blacktriangleright)$ entre les sommets aSb et acb.

Remarque 3.2.12. Le digraphe $\mathbb{G}(\blacktriangleright)$ a comme sommets les mots de $\mathbb{A}^*$. Le mot β s'obtient à partir du mot α ($\alpha \neq \beta$), s'il existe un chemin ω de longueur $L = |\omega|$, i.e. $\omega = \omega_1 \cdots \omega_L \in \mathbb{G}^+$ tel que $s(\omega) = \alpha$, $t(\omega) = \beta$ et $\omega_i \blacktriangleright \omega_{i+1}$, pour $i = 1, \ldots, L-1$. Les arêtes qui composent ω sont indexées par les productions qui les engendrent.

Définition 3.2.13. Soit $\Gamma(\mathbb{A}_t, \mathbb{A}_n, \Pi, S)$ une grammaire avec $\mathbb{A} = \mathbb{A}_t \sqcup \mathbb{A}_n$. On appelle **langage** engendré par Γ l'ensemble

$$\mathcal{L}(\Gamma) = \{\alpha \in \mathbb{A}_t^* : S \blacktriangleright^* \alpha\} = \bigcup_{\omega \in \mathbb{G}_S^*} \{t(\omega)\} \cap \mathbb{A}_t^*.$$

Exemple 3.2.14. Soient $\mathbb{A}_t = \{0, 1\}$, $\mathbb{A}_n = \{S, A\}$ et

$$\Pi = \{(S, 0A1), (0A, 00A), (A1, A11), (A, AA), (A, \varepsilon)\}.$$

Une suite possible de dérivations est la suivante

$$\begin{aligned} S \blacktriangleright 0A1 \blacktriangleright 00A1 \blacktriangleright 00AA1 \blacktriangleright 00AA11 \\ \blacktriangleright 00A11 \blacktriangleright 000A11 \blacktriangleright 00011. \end{aligned}$$

Il est facile de montrer que $\mathcal{L}(\Gamma) = \{0^m 1^n, m, n \geq 1\}$.

Notation 3.2.15. Pour noter les productions, nous utilisons aussi la **forme de Backus-Naur** (BNF). Les production de l'exemple 3.2.14 sont ainsi notées :

— $S \to 0A1$,
— $0A \to 00A$,
— $A1 \to A11$,
— $A \to AA|\varepsilon$.

3.3 Hiérarchie de Chomsky

Il existe une classification de grammaires selon leur puissance d'expression telle que définie par leurs productions. Cette hiérarchie est donnée dans la table 3.1 ci-dessous.

N^o	Type	Productions
0	non-contrainte	$a \longrightarrow \beta, \alpha \in \mathbb{A}^*\mathbb{A}_n\mathbb{A}^*, \beta \in \mathbb{A}^*$
1	contextuelle	$\alpha A\gamma \longrightarrow \alpha\beta\gamma, A \in \mathbb{A}_n, \alpha, \beta, \gamma \in \mathbb{A}^*$
2	non-contextuelle ou algébrique	$A \longrightarrow \alpha, A \in \mathbb{A}_n, \alpha \in \mathbb{A}^*$
3	régulière ou linéaire	$A \longrightarrow \alpha B \vert B\alpha \vert \alpha, A, B \in \mathbb{A}_n, \alpha \in \mathbb{A}_t^*$

TABLE 3.1 – L'hiérarchie des grammaires selon Chomsky.

Exemple 3.3.1. Soit la grammaire Γ dont les productions sont

$$S \longrightarrow aSa|bSb|\varepsilon.$$

Un exemple de dérivation est

$$S \blacktriangleright aSa \blacktriangleright a^2Sa^2 \blacktriangleright a^2bSba^2 \blacktriangleright a^2baSaba^2 \blacktriangleright \ldots$$

Il est facile de montrer que $\mathcal{L}(\Gamma) = \{\alpha\overline{\alpha}, \alpha \in \{a,b\}^*\}$, où $\overline{\alpha}$ est le mot renversé de α.

La classification de Chomsky est très utile car elle induit une correspondance — présentée dans la table 3.2 — entre les grammaires, les langages qui en découlent et les automates qui réconnaissent ces langages.

N^o	Langage	Automate
0	récursivement énumérable	machine de Turing (TM)
1	contextuel	machine de Turing à entrée bornée (LBA)
2	non-contextuel	automate à pile (PDA)
3	régulier	automate (déterministe ou non-déterministe) fini (FA)

TABLE 3.2 – Les langages produits par les grammaires et les automates qui les reconnaissent.

Les chapitres suivants seront consacrés à l'étude de certaines classes d'automates. Ici on donne un avant goût de ces objets conceptuels. En général, un **automate déterministe** est un dispositif (une application) qui à chaque état interne et à chaque donnée en entrée il décide d'une action, fonction uniquement de l'état interne et de l'entrée. Il est dit **non-déterministe** si résultat est choisi parmi une collection finie d'actions possibles (l'action est une « fonction multivaluée » de l'état interne et de l'entrée).

Selon le mode d'évolution de l'automate, on distingue

Machine de Turing (TM) : un dispositif avec un nombre fini d'états internes et une entrée finie mais non bornée,

Automate à entrée bornée (LBA) : où l'entrée cette fois-ci est bornée. L'acronyme LBA provient du nom anglais de l'automate : *linear bounded automaton*.

Automate à pile (PDA) : où l'entrée est finie mais il existe un ruban auxiliaire non-borné permettant de stocker les résultats intermédiaires sous forme de pile de type « dernier venu-premier sorti ». L'acronyme PDA provient du nom anglais de l'automate : *push down automaton* ; la discipline de service dernier venu premier sorti est aussi connue sous son acronyme anglais LIFO pour *last in first out*.

Automate fini (FA) : où il n'y a qu'un nombre fini d'états intermédiaires et une entrée de taille finie.

On a parlé d'hiérarchie pour les grammaires, les langages et les automates de reconnaissance. Cela signifie que les grammaires de type $n+1$ sont des cas particuliers des grammaires de type n, pour $n = 0, 1, 2$. La même inclusion s'applique pour les langages et les automates. En particulier une machine de Turing est un ordinateur universel qui peut simuler tous les automates de type plus élevé. Un automate qui se trouve dans un état initial s_0 et à qui on présente un mot α comme entrée, il exécute certaines opérations et soit il s'arrête en retournant un résultat $r := r(s_0, \alpha)$, soit il ne s'arête pas. On dit qu'un ensemble $E \subseteq U$, (où U désigne l'univers de référence) est **récursivement défini** s'il existe un algorithme fini (i.e. un automate), tel que pour toute entrée $x \in U$, l'automate s'arête en un temps fini et répond « oui » si $x \in E$ et « non » si $x \notin E$. L'ensemble E est dit **récursivement énumérable** si l'automate s'arrête en un temps fini pour les $x \in E$ en répondant « oui », tandis que si $x \notin E$, soit l'automate s'arrête en un temps fini en rejetant x, soit il ne s'arrête pas.

3.4 Arbres de dérivation pour des grammaires de type 2 ou 3

Les grammaires non-contextuelles de type 2 (et par conséquent les grammaires régulières de type 3 qui en sont un cas particulier) admettent une représentation de leur suites de dérivations en arbre ordonné, appelé **arbre des dérivations** (*parsing-tree* en anglais). Pour illustrer cette représentation, revenons à la grammaire de l'exemple 3.3.1 dont les productions sont :

$$S \longrightarrow aSa|bSb|\varepsilon.$$

Considérons la suite des dérivations

$$S \blacktriangleright aSa \blacktriangleright a^2Sa^2 \blacktriangleright a^2bSba^2 \blacktriangleright a^2b\varepsilon ba^2a = a^2bba^2.$$

L'arbre correspondant est représenté dans la figure 3.1. Le mot du langage généré se lit en parcourant les feuilles de l'arbre dans l'ordre $a^2b\varepsilon ba^2 = a^2b\varepsilon ba^2$.

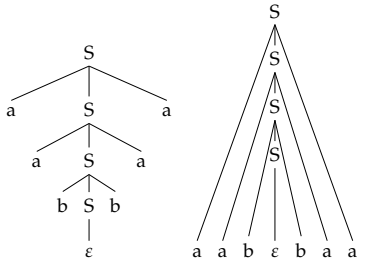


FIGURE 3.1 – Le schéma de gauche représente l'arbre (ordonné) correspondant à la suite des dérivations $S \blacktriangleright aSa \blacktriangleright a^2Sa^2 \blacktriangleright a^2bSba^2 \blacktriangleright a^2b\varepsilon ba^2$. Le schéma de droite, représente la même information mais en mettant tous les nœuds de la frontière de l'arbre sur le même niveau pour rendre le mot terminal (palindromique) a^2bba^2 plus lisible.

Pour l'exemple précédent, la suite des dérivations est unique. Ceci n'est cependant pas toujours le cas comme le montre la remarque 3.4.1.

Remarque 3.4.1. Pour certaines grammaires, on peut parvenir au même mot terminal (en partant de S) en suivant des trajectoires de $G^*(\blacktriangleright)$ différentes.

1. Considérer la grammaire $\Gamma_1 = (\{a\}, \{S\}, \Pi, S)$, ayant comme ensemble de productions $\Pi_1 := S \longrightarrow aS|Sa|\varepsilon$. Une suite de dérivations possibles est
$$S \blacktriangleright aS \blacktriangleright aSa \blacktriangleright aaSa \blacktriangleright aaa.$$
Mais une autre suite des dérivations permet de parvenir au même résultat :
$$S \blacktriangleright Sa \blacktriangleright Saa \blacktriangleright aaa.$$
Cette dernière suite des dérivations s'appelle **gauche**, car à chaque étape, c'est le symbole non-terminal le plus à gauche qui est remplacé. De la même façon nous avons des dérivations droites. Il est évident que $\mathcal{L}(\Gamma_1) = \{a^n, n \in \mathbb{N}\}$. Mais ce langage peut être engendré aussi par la grammaire Γ_1' ayant comme productions $\Pi_1' := S \longrightarrow Sa|\varepsilon$ qui engendre le langage en suivant une unique trajectoire pour chaque mot du langage.
2. Une situation plus problématique est illustrée par la grammaire $\Gamma_2 = (\emptyset, \{S\}, \Pi, S)$ avec $\Pi_2 := S \longrightarrow S|\varepsilon$. Il est évident que $\mathcal{L}(\Gamma_2) = \{\varepsilon\}$ (le langage trivial). Cependant, il y a une infinité de trajectoires de S à ε. On parle alors de **grammaire ambiguë** (voir définition 3.4.2). Bien sûr, le langage trivial peut être engendré par la grammaire non-ambiguë $\Pi_2' := S \longrightarrow \varepsilon$.

Définition 3.4.2. Une grammaire indépendante du contexte Γ est dite **ambiguë** s'il existe de mots de $\mathcal{L}(\Gamma)$ qui s'obtiennent par des dérivations gauches distinctes. Un langage est dit **intrinsèquement ambigu** s'il n'est engendré que par des grammaires ambiguës.

Les grammaires utilisés en informatique pour les compilateurs doivent être impérativement non-ambiguës. Considérons en effet la grammaire suivante avec

un ensemble de mots réservés : $\mathcal{R} = \{\text{if}, \text{then}, \text{else}, \ldots\}$
un ensemble d'instructions impératives : $\mathcal{I} = \{i_1, i_2, \ldots\}$,
un ensemble de conditions booléennes : $\mathcal{C} = \{c_1, c_2, \ldots\}$,
un ensemble de symboles terminaux : $\mathbb{A}_t = \mathcal{R} \sqcup \mathcal{I} \sqcup \mathcal{C}$,
un ensemble de symboles non-terminaux : $\mathbb{A}_n = \{S, C, I\}$,
un ensemble de productions :

$$\begin{aligned} S &\longrightarrow \text{ if } C \text{ then } S | \text{ if } C \text{ then } S \text{ else } S | I \\ C &\longrightarrow c_1 | c_2 | \cdots \\ I &\longrightarrow i_1 | i_2 | \cdots \end{aligned}$$

Dans la figure 3.2, on représente le début de deux trajectoires distinctes qui peuvent mener aux même mots du langage.

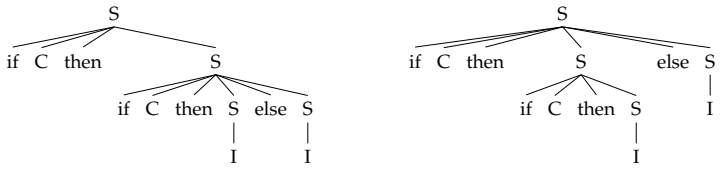


FIGURE 3.2 – L'expression intermédiaire « if C then if C then I else I » peut être obtenue par deux trajectoires distinctes qui peuvent mener au même mot du langage, par exemple le mot « if c_1 then if c_2 then i_1 else i_2 ». Or la trajectoire calculatoire qui correspond à l'arbre de gauche est analysée (parsed) par le compilateur comme « if C then if C then I else I endif endif » tandis que celle de l'arbre de droite comme « if C then if C then I endif else I endif ».

3.5 Exercices

Langages formels, grammaires

24. Soit la grammaire non-contextuelle $\Gamma = \{\mathbb{A}_t, \mathbb{A}_n, \Pi, S)$ avec $\mathbb{A}_t = \{a, b, c\}$, $\mathbb{A}_n = \{S, A, B\}$ et productions

$$\begin{aligned} S &\to aAB \\ A &\to Bba \\ B &\to bB | c. \end{aligned}$$

Le mot *acbabc* appartient-il au langage $\mathcal{L}(\Gamma)$? (Justifier votre réponse).

25. Décrire le langage $L(\Gamma)$ engendré par les grammaires suivantes :
    (a) $S \to aS \mid bA$ et $A \to bA \mid c$,
    (b) $S \to aaS \mid aa$,

(c) $S \to aaS \mid a \mid b$,
(d) $S \to aA$ et $A \to bS \mid a$.

26. Pour chacune des grammaires de l'exercice précédent, préciser son type.

27. Soit Γ la grammaire décrite par les productions :
— $S \to A \mid AC$,
— $A \to a \mid b \mid c$,
— $C \to AC \mid BC \mid A \mid B$,
— $B \to 0 \mid 1 \mid \cdots, \mid 9$.
Repérer les affirmations vraies parmi les suivantes :
(a) $ab037 \in L(\Gamma)$,
(b) $S \blacktriangleright^* 2a3b$,
(c) $aaaa \in L(\Gamma)$,
(d) $S \blacktriangleright a$,
(e) $S \blacktriangleright^* ab$,
(f) $BC \blacktriangleright 2$,
(g) $BC \blacktriangleright^+ 2$,
(h) $C \blacktriangleright^* 2abc$,
(i) $C \blacktriangleright^5 3b4$.

28. Écrire une grammaire génératrice pour les langages suivants :
(a) $L = \{a^m b^n ; m \geq 1, n \geq 1\}$,
(b) $L = \{a^m b^n ; m \geq 0, n \geq 3\}$.

29. Soit $\Gamma = (\mathbb{A}_t, \mathbb{A}_n, \Pi, S)$, avec $\mathbb{A} = \mathbb{A}_t \sqcup \mathbb{A}_n$, une grammaire non contextuelle, i.e. les productions sont de la forme $A \to \alpha$, avec $A \in \mathbb{A}_n$ et $\alpha \in \mathbb{A}^*$. Le but de cet exercice est de montrer le
Théorème : Soient $\alpha^{(1)}, \alpha^{(2)}, \beta \in \mathbb{A}^*$ et $k \in \mathbb{N}_>$. Si $\alpha^{(1)}\alpha^{(2)} \blacktriangleright^k \beta$, alors il existe des mots $\beta^{(1)}, \beta^{(2)} \in \mathbb{A}^*$ et des entiers $k_1, k_2 \in \mathbb{N}$, tels que $k = k_1 + k_2$, $\alpha^{(1)} \blacktriangleright^{k_1} \beta^{(1)}$, $\alpha^{(2)} \blacktriangleright^{k_2} \beta^{(2)}$ et $\beta = \beta^{(1)}\beta^{(2)}$.
(a) Pour $\alpha, \beta \in \mathbb{A}^*$, rappeler la définition de $\alpha \blacktriangleright \beta$.
(b) Si $k = 1$ montrer que la conclusion du théorème est correcte. *Suggestion :* Considérer deux cas pour la dérivation directe $\alpha = \alpha^{(1)}\alpha^{(2)} \blacktriangleright \beta$ en se servant de la définition rappelée en question 1.
(c) En supposant que la conclusion du théorème soit vraie pour un certain $k \geq 1$, montrer qu'elle reste vraie pour $k + 1$.

4
Logique et langages

> THE design of the following treatise is to investigate the fundamental laws of those operations of the mind by which reasoning is performed ; to give expression to them in the symbolical language of a Calculus, and upon this foundation to establish the science of Logic and construct its method ; to make that method itself the basis of a general method for the application of the mathematical doctrine of Probabilities ; and, finally, to collect from the various elements of truth brought to view in the course of these inquiries some probable intimations concerning the nature and constitution of the human mind.
>
> George BOOLE, *An investigation of the laws of thought*, Cambridge [5].

La logique mathématique, initialement développée comme une branche de l'algèbre par Boole [5] au 19e siècle, a fasciné plusieurs mathématiciens durant le 20e siècle (Frege, Gödel, Russel, Whitehead, ...) ; elle est cependant quelque peu délaissée dans les cursus standard de mathématiques malgré son importance fondamentale dans toute activité scientifique. La logique est la discipline qui s'intéresse aux lois qui gouvernent l'inférence correcte et fiable.

Le présent cours ayant d'autres objectifs, se limite à un survol des notions de logique nécessaires pour l'introduction rigoureuse de la notion d'algorithme comme opération effective de calcul et de sa complexité déterminant son efficacité.

Ce chapitre a pour but

— de rappeler brièvement des notions de logique bi-valuée (booléenne) propositionnelle et de la logique du premier ordre et d'appliquer les constructions vues aux chapitres précédentes (grammaires et langages) pour décrire la logique comme un langage et
— de préparer le terrain pour les chapitres suivants sur les automates.

Il est donc d'emblée clair que nous n'aborderons pas dans ce chapitre des généralisations comme la logique modale, la logique floue, etc.

4.1 Algèbres de Boole, ensembles partiellement ordonnés, treillis

Définition 4.1.1. Soit $\mathbb{X}$ un ensemble arbitraire, contenant au moins deux symboles distincts $\mathbf{0}$ et $\mathbf{1}$, muni
- — d'une opération unaire $\mathbb{X} \ni x \mapsto \overline{x} \in \mathbb{X}$,
- — de deux opérations binaires, notées respectivement $+$ et $\times$, définies par

$$\mathbb{X} \times \mathbb{X} \ni (x, y) \mapsto x + y \in \mathbb{X} \text{ et } \mathbb{X} \times \mathbb{X} \ni (x, y) \mapsto x \cdot y \in \mathbb{X},$$

vérifiant, pour $x, y, z \in \mathbb{X}$, les propriétés suivantes :

complémentation : $x + \overline{x} = \mathbf{1}$ et $x \cdot \overline{x} = \mathbf{0}$;
neutre : $x + \mathbf{0} = x$ et $x \cdot \mathbf{1} = x$;
commutativité : $x + y = y + x$ et $x \cdot y = y \cdot x$;
associativité : $x + (y + z) = (x + y) + z$ et $x \cdot (y \cdot z) = (x \cdot y) \cdot z$;
distributivité : $x \cdot (y + z) = x \cdot y + x \cdot z$ et $x + (y \cdot z) = (x + y) \cdot (x + z)$.

La structure $(\mathbb{X}, +, \cdot, \mathbf{0}, \mathbf{1})$ est appélée [1] **algèbre de Boole**.

Exemple 4.1.2. La structure $\mathbb{B}_2 := (\{0,1\}, +, \cdot)$, dont les opérations sont définies par

$+$	0	1
0	0	1
1	1	1

et

$\cdot$	0	1
0	0	0
1	0	1

est une algèbre de Boole.

Plus généralement, la structure $\mathbb{B}_2^n := (\{0,1\}^n, +, \cdot, \mathbf{0} = 0^n, \mathbf{1} = 1^n)$, pour $n \geq 1$, avec les opérations $+$ et $\cdot$ définies composante par composante à partir de celles définies sur $\mathbb{B}_2$, est une algèbre de Boole.

Exemple 4.1.3. Soit Ω un ensemble fini. La structure

$$\mathbb{B} := (\mathcal{P}(\Omega), + = \cup, \cdot = \cap, \mathbf{0} = \emptyset, \mathbf{1} = \Omega)$$

est une algèbre de Boole.

Théorème 4.1.4. *Dans une algèbre de Boole $(\mathbb{X}, +, \cdot, \mathbf{0}, \mathbf{1})$, pour tout $x, y \in \mathbb{X}$,*

1. $\overline{\mathbf{0}} = \mathbf{1}$ *et* $\overline{\mathbf{1}} = \mathbf{0}$;
2. $x + x = x$ *et* $xx = x$;
3. *si* $x + y = \mathbf{1}$ *et* $xy = \mathbf{0}$, *alors* $y = \overline{x}$;
4. $\overline{x + y} = \overline{x}\,\overline{y}$ *et* $\overline{xy} = \overline{x} + \overline{y}$;

1. Comme pour la multiplication numérique, on omet souvent le symbole $\cdot$.

5. *l'opération $x \mapsto \overline{x}$ est une involution, i.e. $\overline{\overline{x}} = x$.*

Définition 4.1.5. Le couple $(\mathbb{X}, \preceq)$ où $\mathbb{X}$ un ensemble arbitraire et $\preceq$ une relation d'ordre partiel sur $\mathbb{X}$ est dit **ensemble partiellement ordonné**, très souvent désigné par le néologisme anglais **poset** (pour *partially ordered set*) [2].

Lorsque $\mathbb{X}$ est un ensemble fini, une manière pratique de représenter le poset $(\mathbb{X}, \preceq)$ est par son **diagramme de Hasse** défini par les règles suivantes :

— les éléménts de $\mathbb{X}$ sont représentés comme points du plan ;
— lorsque $x \prec y$ dans $\mathbb{X}$, l'ordonnée du représentant de x est plus petite que celle de y ;
— si $x \prec y$ dans $\mathbb{X}$ et il n'y a pas de $z \in \mathbb{X}$ tel que $x \prec z \prec y$, alors les représentants de x et y sont reliés par une arête.

La figure 4.1 représente quelques exemples de diagrammes de Hasse de posets.

Définition 4.1.6.
— Si un poset contient un élément distingué **0** tel que pour tout $x \in \mathbb{X}$ on a $\mathbf{0} \preceq x$, alors cet élément est unique et est appelé l'élément minimal de $\mathbb{X}$.
— Par dualité, si un poset contient un élément distingué **1** tel que pour tout $x \in \mathbb{X}$ on a $x \preceq \mathbf{1}$, alors cet élément est unique et est appelé l'élément maximal de $\mathbb{X}$.
— Dans un poset avec élément minimal, si pour un élément $x \in \mathbb{X}$ vérifiant $\mathbf{0} \prec x$, il n'existe pas $y \in \mathbb{X}$ tel que $\mathbf{0} \prec y \prec x$, alors x est un **atome**.
— Si $X \subseteq \mathbb{X}$ et $u \in \mathbb{X}$ est un élément tel que $\forall x : x \in X \Rightarrow x \preceq u$, alors u est un **majorant** de X. Un majorant s de X, tel que $s \preceq u$ pour tous les majorants de X, est appelé **borne supérieure** de X, notée $\sup X$. Les minorants et la borne inférieure $\inf X$ sont définis dualement. Noter que si $\sup X$ et $\inf X$ existent, elles sont uniques.

Définition 4.1.7. Un **treillis** est un poset $(\mathbb{X}, \preceq)$ tel que tout couple $\{x, y\}$ d'éléments de $\mathbb{X}$ possède une borne supérieure $\sup\{x, y\} \in \mathbb{X}$, notée $s := x \vee y$, et une borne inférieure $\inf\{x, y\} \in \mathbb{X}$, notée $i := x \wedge y$.

Un treillis est **complet**, si toute partie finie $X \subseteq \mathbb{X}$ possède une borne supérieure et une borne inférieure (dans $\mathbb{X}$). Le treillis est **σ-complet** lorsque pour toute partie dénombrable $X \subseteq \mathbb{X}$, les bornes $\sup X$ et $\inf X$ existent (dans $\mathbb{X}$).

Le treillis est **atomique** si tout élément de $\mathbb{X}$ est la borne supérieure d'un ensemble d'atomes.

Parmi les posets de la figure 4.1, les diagrammes (a)–(c) représentent des treillis. Par contre (d) et (e) ne représentent pas de treillis.

2. L'acronyme *epo* qui pourrait le désigner en français a une connotation ... quelque peu avillie.

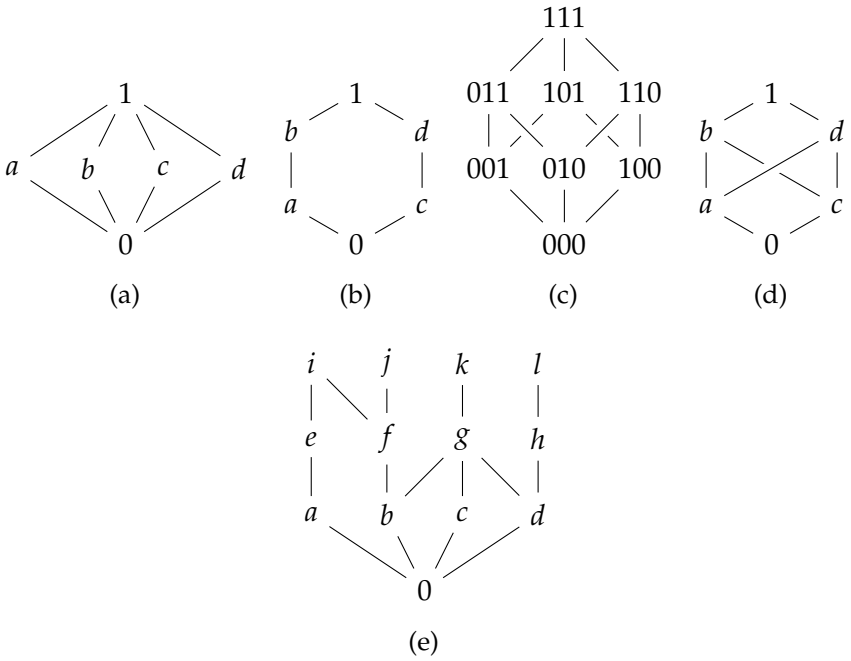


FIGURE 4.1 – Quelques exemples de posets représentés par leurs diagrammes de Hasse. Les posets (a)–(d) ont un élément minimal et un maximal. Le poset (e) a un élément minimal mais pas d'élément maximal ; les éléments i, j, k, l sont des majorants de certaines parties du poset.

Lemme 4.1.8. *Soient $\mathbb{X}$ un treillis et $x, y \in \mathbb{X}$ tels que $x \preceq y$. Alors $\forall z : z \in \mathbb{X}$, nous avons*

$$x \wedge z \leq y \wedge z.$$

Théorème 4.1.9. *Dans un treillis $\mathbb{X}$, les operations $\vee$ et $\wedge$ vérifient, pour tous $x, y, z \in \mathbb{X}$, les propriétés suivantes :*

idempotence	$x \wedge x = x, x \vee x = x,$
commutativité	$x \wedge y = y \wedge x, x \vee y = x \vee y,$
associativité	$x \wedge (y \wedge z) = (x \wedge y) \wedge z, x \vee (y \vee z) = (x \vee y) \vee z,$
absorption	$x \wedge (x \vee y) = x \vee (y \wedge y) = x,$
cohérence	$x \preceq y \iff x \wedge y = x \iff x \vee y = y,$
inégalité modulaire	$x \preceq z \implies x \vee (y \wedge z) \preceq (x \vee y) \wedge z,$
inégalités distributives	$x \wedge (y \vee z) \succeq (x \wedge y) \vee (x \vee z)$ *et*
	$x \vee (y \wedge z) \preceq (x \vee y) \wedge (x \vee z).$

En outre, les quatre premières conditions (idempotence, commutativité, associativité, absorption) caractérisent totalement le treillis.

Définition 4.1.10. Soit $\mathbb{X}$ un treillis.

— Le treillis est **distributif** si les inégalités distributives sont des égalités, i.e.

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z) \text{ et } x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z), \forall x, y, z \in \mathbb{X}.$$

— Le treillis est **complémenté** si pour tout $x \in \mathbb{X}$, il existe $\overline{x} \in \mathbb{X}$ tel que $x \vee \overline{x} = \mathbf{1}$ et $x \wedge \overline{x} = \mathbf{0}$.

Remarque 4.1.11. Un treillis distributif complémenté $(\mathbb{X}, \wedge, \vee, \mathbf{0}, \mathbf{1})$ est une **algèbre de Boole**. Si en outre il est σ-complet, alors il est une **tribu**[3]. Ainsi, une **probabilité classique**, n'est qu'une fonction σ-additive normalisée sur un treillis distributif complémenté.

Exemple 4.1.12. Soit $\mathbb{V}$ un espace vectoriel (euclidien ou hilbertien) de dimension finie et $\mathcal{F} = \mathcal{S}(\mathbb{V})$ l'ensemble de sous-espaces vectoriels de $\mathbb{V}$. La relation « être sous-espace vectoriel de » est un ordre partiel, noté $\subseteq$, qui rend $(\mathcal{S}(\mathbb{V}), \subseteq)$ un poset. En notant $A \vee B = \mathsf{vect}(A \cup B)$ et $A \wedge B = A \cap B$, pour tout $A, B \in \mathcal{S}$, le couple $(\mathcal{S}, \subseteq)$ devient un treillis complémenté par l'involution d'orthogonalité $A \mapsto A^{\perp}$ avec $O = \{0\}$ et $I = \mathbb{V}$. Ce treillis $(\mathcal{S}, \vee, \wedge, \{0\}, \mathbb{V})$ n'est pas distributif (donc il ne constitue pas une algèbre de Boole). Il s'agit d'un treillis modulaire orthocomplémenté qui est σ-complet lorsque l'espace vecoriel $\mathbb{V}$ est de Banach. Les mesures des probabilité (fonctions σ-additives normalisées) sur ces treillis correspondent aux **probabilités quantiques**[4].

4.2 Fonctions booléennes

Définition 4.2.1. Soit $\mathbb{X}$ un ensemble arbitraire. Une application $f : \mathbb{X} \to \mathbb{B}_2$ est appelée **fonction booléenne.**

Il est évident que $\mathsf{card}\{f : \mathbb{X} \to \mathbb{B}_2\} = 2^{\mathsf{card}\mathbb{X}} = \mathsf{card}\mathcal{P}(\mathbb{X})$ car les fonctions booléennes coïncident avec les indicatrices des parties de $\mathbb{X}$.

Un cas particulier important est celui où $\mathbb{X} = \mathbb{B}_2^n$ pour un $n \geq 1$ fixé. Dans ce cas, une fonction booléenne f donne une valeur 0 ou 1 (interprétée comme « fausse » ou « vraie ») sur les n bits de son argument. En identifiant cette suite de n bits avec la représentation binaire d'un entier, f n'est finalement qu'une application $f : \{0, \ldots, 2^n - 1\} \to \{0, 1\}$. La figure 4.2 représente la réalisation de f par un circuit électronique du type « boîte noire ».

Exemple 4.2.2. (Fonctions booléennes $\mathbb{B}_2^n \to \mathbb{B}_2$).

$n = 1$ **:** il existe 4 fonctions booléennes (f_i), $i = 0, \ldots, 3$.

b_0	$f_0(b_0)$	$f_1(b_0)$	$f_2(b_0)$	$f_3(b_0)$
0	0	0	1	1
1	0	1	0	1
	0	ID	NOT	**1**

Les fonctions f_0 et f_3 sont les fonctions constantes. La fonction f_1 est la fonction identité tandis que f_2 est la fonction de renversement du bit d'entrée, connue sous le nom de NOT.

3. Voir cours de probabilités.
4. Voir cours *Mathematical foundations of quantum mechanics.*

$n = 2$: il existe 16 fonctions booléennes (f_i), $i = 0, \ldots, 15$. Ces fonctions sont représentées dans le tableau ci-dessous.

b_1	b_0	$f_0(b_1b_0)$	$f_1(b_1b_0)$	$f_2(b_1b_0)$	$f_3(b_1b_0)$	$f_4(b_1b_0)$	$f_5(b_1b_0)$	$f_6(b_1b_0)$	$f_7(b_1b_0)$	$f_8(b_1b_0)$	$f_9(b_1b_0)$	$f_{10}(b_1b_0)$	$f_{11}(b_1b_0)$	$f_{12}(b_1b_0)$	$f_{13}(b_1b_0)$	$f_{14}(b_1b_0)$	$f_{15}(b_1b_0)$
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

On reconnaît aisément certaines fonctions : $f_1 = \text{AND}$, $f_7 = \text{OR}$, $f_6 = \text{XOR}$, $f_{14} = \text{NAND}$, etc.

Soit $(\mathbb{X}, +, \cdot)$ une algèbre de Boole arbitraire finie et $(\mathbb{X}, \vee, \wedge)$ le treillis distributif complémenté correspondant. On note $\mathsf{At}(\mathbb{X})$, l'ensemble des atomes du treillis. Pour tout x de l'algèbre (resp. du treillis) il existe un entier $r := r(x)$ et des atomes $\{a_1, \ldots, a_r\} \in \mathsf{At}(\mathbb{X})$ qui permettent d'écrire x de manière unique (à des permutations d'indices près) comme $x = a_1 + a_2 + \ldots + a_{r(x)}$ (resp. $x = a_1 \vee a_2 \vee \ldots \vee a_{r(x)}$). L'application définie par $\mathbb{X} \ni x \mapsto \beta(x) := \{a_1, \ldots, a_r\}$ établit une bijection entre $\mathbb{X}$ et $\mathcal{P}(\mathsf{At}(\mathbb{X}))$. Par conséquent, toute algèbre de Boole finie a 2^N éléments, où $N = \mathsf{card}(\mathsf{At}(\mathbb{X}))$.

Les nombres utilisé sur un ordinateur constituent une partie finie des rationnels qui peuvent être mis en bijection avec une partie finie de $\mathbb{N}$. Par conséquent, toute application $F : \mathbb{N} \to \mathbb{N}$ peut-être réalisée *in fine* par une collection $(f_0, \ldots, f_m)$ d'applications $\mathbb{B}_2^n \to \mathbb{B}_2$.

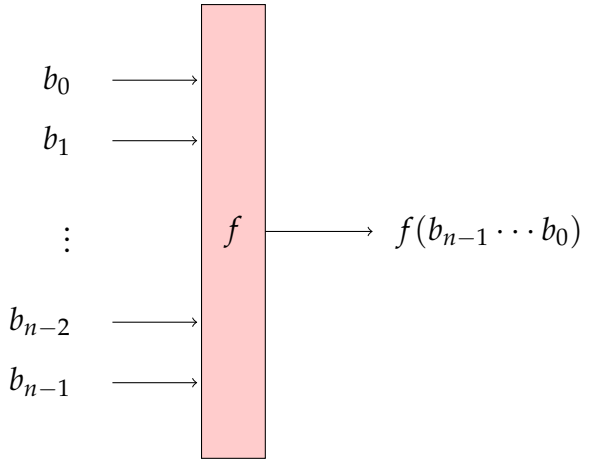


FIGURE 4.2 – Réalisation du type « boîte noire » de le fonction booléenne f.

Étant donné que toute expression booléenne peut être convertie en une forme disjonctive canonique, il s'ensuit que toute fonction booléenne f peut être décomposée en une disjonction de conjonctions. Par conséquent, la « boîte noire » qui représente f dans la figure 4.2 peut-être exprimée par une combinaison des fonctions AND, OR et NOT. On peut même exprimer toute fonction booléenne en utilisant uniquement de combinaisons [5] de la fonction

5. Ce résultat était connu dès 1888 par Peirce, *A Boolian algebra with one constant*, [Peirce CP4.12] in *The Collected Papers of Charles Sanders Peirce*, Harvard Unversity Press (1933).

NAND [60].

4.3 La logique comme langage

4.3.1 Logique propositionnelle

Le but de ce paragraphe est de montrer que les formules logiques syntaxiquement correctes, désignées ci-dessous par l'acronyme anglais wwf (pour *well-formed formulæ*) constituent un langage de cardinal dénombrable. Des présentations plus complètes peuvent être trouvées dans [1, 17, 18, 64].

On considère un alphabet $\mathbb{A}$ dénombrable décomposable en trois parties disjointes : une partie finie $\mathbb{A}_l = \{\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow\}$ contenant les **connecteurs logiques**, une partie finie $\mathbb{A}_s = \{(,)\}$ contenant les **séparateurs** — servant à lever l'ambiguïté de certaines formules — une partie finie $\mathbb{A}_c = \{\top, \bot\}$ représentant les constantes logiques de tautologie et de contradiction et une partie $\mathbb{A}_{\mathrm{pr}}$ dénombrable (finie ou infinie) non-vide contenant les **variables propositionnelles**. Les variables propositionnelles sont habituellement notées par les lettres minuscules. On note $\mathbb{A} = \mathbb{A}_l \sqcup \mathbb{A}_s \sqcup \mathbb{A}_{\mathrm{pr}} \sqcup \mathbb{A}_c$ et on considère sa clôture monoïdale $\mathbb{A}^*$. On identifie les éléments de $\mathbb{A}_{\mathrm{pr}}$ avec les mots de longueur 1 de $\mathbb{A}^*_{\mathrm{pr}}$ (pour $(p) \in \mathbb{A}^*$, on écrira donc p au lieu de (p)).

Définition 4.3.1. L'ensemble $\mathcal{F} := \mathcal{F}(\mathbb{A}_{\mathrm{pr}})$ des **formules propositionnelles bien formées** sur $\mathbb{A}_{\mathrm{pr}}$ est le langage qui coïncide avec la plus petite partie de $\mathbb{A}^*$ qui

— contient $\mathbb{A}_{\mathrm{pr}} \sqcup \mathbb{A}_c$,
— si $p \in \mathcal{F}$ alors $\neg p \in \mathcal{F}$,
— si $p, q \in \mathcal{F}$ alors les mots $(p \vee q), (p \wedge q), (p \Rightarrow q)$ et $(p \Leftrightarrow q)$ appartiennent aussi à $\mathcal{F}$.

Ainsi, p ou $\neg(p \Rightarrow q)$ sont des formules tandis que $(p \vee q \vee r)$ ou $(p \Rightarrow q \wedge r)$ ne le sont pas. On peut décrire $\mathcal{F}$ par une construction récursive des mots qui le composent :

Théorème 4.3.2. *On introduit une suite croissante $(\mathcal{F}_n)$ de parties de $\mathbb{A}^*$ comme suit :*

— $\mathcal{F}_0 = \mathbb{A}_{pr}$ *et*
— *récursivement pour $n > 0$ par*

$$\mathcal{F}_n = \mathcal{F}_{n-1} \cup \{\neg\phi : \phi \in \mathcal{F}_{n-1}\} \cup \{(\phi\lambda\psi) : \phi, \psi \in \mathcal{F}_{n-1}; \lambda \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}\}.$$

Alors $\mathcal{F} = \cup_{n\in\mathbb{N}} \mathcal{F}_n$.

Remarque 4.3.3. Le langage $\mathcal{F}$ est dénombrable. Il peut aussi être décrit par la forme Backus-Naur de sa grammaire en définissant comme ensemble des symboles terminaux $\mathbb{A}_t = \mathbb{A}_l \sqcup \mathbb{A}_s \sqcup \mathbb{A}_c \sqcup \mathbb{A}_{\mathrm{pr}}$, comme ensemble des

symboles non-terminaux $\mathbb{A}_n = \{S, V, C\}$ et les productions

$$\begin{aligned} C &\to \wedge | \vee | \Rightarrow | \Leftrightarrow \\ V &\to p|q|r|\dots \\ S &\to V|\neg(S)|(S)C(S)|\top|\bot. \end{aligned}$$

On appelle **hauteur** d'une formule (mot) $\phi \in \mathcal{F}$, et on note $h(\phi)$, le plus petit entier n tel que $\phi \in \mathcal{F}_n$. On peut aussi construire l'**arbre de dérivation** d'une formule ϕ comme l'arbre à $h(\phi)$ générations. Par exemple, la formule $\phi = ((p \vee q) \wedge (r \wedge s))$ et bien formée car elle peut être analysée selon son arbre de dérivation comme l'arbre à gauche de la figure 4.3. Une autre analyse possible est selon l'**arbre de décomposition** de la formule (à droite de la figure 4.3).

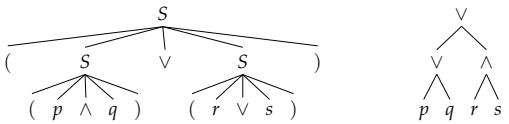


FIGURE 4.3 – La formule ϕ est bien formée car elle peut être analysée sans ambiguïté, soit en suivant son arbre de dérivation (à gauche), soit son arbre de décomposition (à droite).

Un résultat simple à montrer — mais important par ses implications — est que pour toute formule de $\mathcal{F}$, le nombre de parenthèses ouvrantes est égal au nombre de parenthèses fermantes. Ceci permet d'établir que pour chaque formule, il existe un schéma unique de lecture. Une formule peut être simplifiée en enlevant éventuellement le couple de parenthèses externes. Certains couples de parenthèses internes peuvent être aussi enlevés si des règles de préséance ou de distribution [6] sont en force. En logique booléenne, on peut ramener toute formule bien formée en une **forme normale disjonctive (FND)** ou **forme normale conjonctive (FNC)**. La forme normale disjonctive est une disjonction d'une ou plusieurs conjonctions et chaque variable apparaît au plus une fois dans chaque conjonction. Par exemple les formules $p, p \vee q, (p \wedge q) \vee r, (p \wedge \neg q \wedge \neg r) \vee (s \wedge t)$ sont des FND. Par contre, la formule (légale) $\neg(p \vee q)$ ne l'est pas car la négation s'applique sur toute la parenthèse. En logique booléenne, la FND de la dernière formule est $\neg p \wedge \neg q$. Dualement, on peut définir la forme normale conjonctive d'une formule comme la conjonction d'une ou plusieurs disjonctions ; chaque variable apparaît au plus une fois dans chaque disjonction.

Le théorème 4.3.2 permet la construction récursive de formules syntaxiquement bien formées ; l'arbre de dérivation permet de vérifier si une formule est syntaxiquement bien formée. Cependant, ces deux procédés ne

6. Il n'est pas souhaitable de fixer des règles de distribution à ce stade. Nous envisageons en effet des logiques qui **ne vérifient pas nécessairement la distribution** $p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$ ou sa duale.

nous disent rien sur le sens des formules. Pour une interprétation sémantique des formules nous devons fixer les valeurs de vérité des variables propositionnelles.

Définition 4.3.4. Une **valuation** est une application $v : \mathcal{F}_0 := \mathbb{A}_{\mathrm{pr}} \to \{0,1\}$.

Pour toute valuation v sur $\mathbb{A}_{\mathrm{pr}}$, on peut définir par récurrence une unique extension $\mathbf{v} := \mathbf{v}_v : \mathcal{F} \to \{0,1\}$. Il suffit en effet de définir, pour tout $p \in \mathbb{A}_{\mathrm{pr}}$, $\mathbf{v}(p) = v(p)$ et ensuite l'étendre uniquement aux opérations élémentaires qui interviennent dans la construction de $\mathcal{F}$:

— $\mathbf{v}(\neg\phi) = 0$ si, et seulement si, $\mathbf{v}(\phi) = 1$,
— $\mathbf{v}(\phi \wedge \psi) = 1$ si, et seulement si, $\mathbf{v}(\phi) = 1$ et $\mathbf{v}(\psi) = 1$,
— $\mathbf{v}(\phi \vee \psi) = 0$ si, et seulement si, $\mathbf{v}(\phi) = 0$ et $\mathbf{v}(\psi) = 0$,
— $\mathbf{v}(\phi \Rightarrow \psi) = 0$ si, et seulement si, $\mathbf{v}(\phi) = 1$ et $\mathbf{v}(\psi) = 0$,
— $\mathbf{v}(\phi \Leftrightarrow \psi) = 1$ si, et seulement si, $\mathbf{v}(\phi) = \mathbf{v}(\psi)$,

pour toutes les formules $\phi, \psi \in \mathcal{F}$.

La définition de l'extension ci-dessus équivaut aux tables de vérité suivantes :

$v(p)$	$\mathbf{v}(\neg p)$
0	1
1	0

$v(p)$	$v(q)$	$\mathbf{v}(p \wedge q)$	$\mathbf{v}(p \vee q)$	$\mathbf{v}(p \Rightarrow q)$	$\mathbf{v}(p \Leftrightarrow q)$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

Par abus de notation, très souvent on écrira v au lieu de $\mathbf{v}$ pour cette extension et on vérifie immédiatement qu'elle peut également être définie par les relations fonctionnelles :

$$\begin{aligned} v(\neg p) &= 1 \oplus v(p) \\ v(p \wedge q) &= v(p)v(q) \\ v(p \vee q) &= v(p) \oplus v(q) \oplus v(p)v(q) \\ v(p \Rightarrow q) &= 1 \oplus v(p) \oplus v(p)v(q) \\ v(p \Leftrightarrow q) &= 1 \oplus v(p) \oplus v(q), \end{aligned}$$

où $\oplus$ désigne l'addition modulo 2.

Si $\mathcal{F}$ est un langage propositionnel et $v : \mathcal{F} \to \{0,1\}$ une valuation, alors v induit une **sémantique** (une interprétation) sur les formules. En particulier $v^{-1}(1)$ est la partie du langage contenant les formules vraies.

Définition 4.3.5. Soit $\phi \in \mathcal{F}$ une formule fixée.

1. Un **modèle** pour ϕ est une valuation v telle que $\mathbf{v}_v(\phi) = 1$.
2. ϕ est **satisfaisable** si elle admet au moins un modèle ; sinon elle est dite **insatisfaisable**.

3. ϕ est une **tautologie** si $\mathbf{v}_v(\phi) = 1$ pour toute valuation v. On écrit alors $\models \phi$.

Exemple 4.3.6. Pour des propositions p et q, nous avons :
— $\models \neg p \vee p$,
— $\neg p \wedge p$ est insatisfaisable,
— La valuation $v(p) = 1$ et $v(q) = 0$ est un modèle pour $p \vee (\neg q \Rightarrow p)$, mais la valuation w, définie par $w(p) = w(q) = 1$, l'est aussi.

Définition 4.3.7. Soient Φ une collection de formules de $\mathcal{F}$ et $\psi \in \mathcal{F}$. On dit que ψ est une **conséquence logique** de Φ si tout modèle pour (toutes les formules de) Φ est aussi un modèle pour ψ. On note alors $\Phi \models \psi$.

Remarque 4.3.8. Si $\Phi = \{\phi_1, \ldots, \phi_n\} \subseteq \mathcal{F}$, la conséquence logique $\Phi \models \psi$ signifie que $\phi_1 \wedge \ldots \wedge \phi_n \Rightarrow \psi$ est une tautologie. De manière équivalente, la formule $\phi_1 \wedge \ldots \wedge \phi_n \Rightarrow \neg\psi$ est instatisfaisable (déduction par réfutation). Noter la différence entre $\Rightarrow$ qui est un connecteur entre éléments de $\mathcal{F}$ et $\models$ qui fait intervenir la sémantique.

Exemple 4.3.9. $p, p \Rightarrow q \models q$, car toute valuation v vérifie $v((p \wedge (p \Rightarrow q)) \Rightarrow q) = 1$.

Définition 4.3.10. Soient $\phi, \psi \in \mathcal{F}$. On dit que ϕ et ψ sont **logiquement équivalentes**, et on note $\phi \equiv \psi$, si $\phi \models \psi$ et $\psi \models \phi$.

Proposition 4.3.11. *Pour p, q, r, s propositions, nous avons les équivalences logiques suivantes.*

$$
\begin{aligned}
\textit{implication} \quad & p \Rightarrow q \equiv \neg p \vee q, \\
\textit{équivalence entre connecteurs} \quad & p \Leftrightarrow q \equiv (\neg p \vee q) \wedge (\neg q \vee p) \\
& \equiv (p \wedge q) \vee (\neg p \wedge \neg q), \\
\textit{double négation} \quad & \neg\neg p \equiv p, \\
\textit{règles de Morgan} \quad & \neg(p \wedge q) \equiv \neg p \vee \neg q \ \textit{ et } \ \neg(p \vee q) \equiv \neg p \wedge \neg q, \\
\textit{idempotence} \quad & p \vee p \equiv p \wedge p \equiv p, \\
\textit{commutativité} \quad & p \vee q \equiv q \vee p \ \textit{ et } \ p \wedge q \equiv q \wedge p, \\
\textit{associativité} \quad & p \vee (q \vee r) \equiv (p \vee q) \vee r \ \textit{ et} \\
& p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r, \\
\textit{contradiction} \quad & \neg p \wedge p \equiv \bot, \\
\textit{tertium non datur} \quad & \neg p \vee p \equiv \top, \\
\textit{domination} \quad & p \vee \top \equiv \top \ \textit{ et } \ p \wedge \bot \equiv \bot, \\
\textit{éléments neutres} \quad & p \vee \bot \equiv p \ \textit{ et } \ p \wedge \top \equiv p, \\
\textit{distributivité} \quad & p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r) \ \textit{ et} \\
& p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r), \\
\textit{absorption} \quad & p \wedge (p \vee q) \equiv p \ \textit{ et } \ p \vee (p \wedge q) \equiv p.
\end{aligned}
$$

Dans l'approche précédente, la déduction logique utilise la sémantique des formules. Cependant, si on veut procéder formellement (sur ordinateur par exemple), la déduction doit être uniquement syntaxique. On parle alors d'**approche formelle de la logique propositionnelle**. Celle-ci utilise

les axiomes i.e. les propositions primitives non démontrables,
les théorèmes, i.e. les propositions obtenues à partir des axiomes ou de théorèmes déjà démontrés à l'aide de règles syntaxiques,
les règles d'inférence, i.e. un schéma de raisonnement permettant d'obtenir de nouveaux théorèmes à partir de **prémisses**[7] (d'autres théorèmes ou d'hypothèses).

Si $p_1, \ldots, p_n$ sont des prémisses et T le théorème qui en découle syntaxiquement on note $p_1, \ldots, p_n \vdash T$. On dit que T est le **séquent** (conséquence syntaxique) des **prémisses** $p_1, \ldots, p_n$. Par exemple la déduction

p_1	« il pleut »
p_2	« s'il pleut je prends mon parapluie »
T	« je prends mon parapluie »,

est notée $p_1, p_2 \vdash T$.

Remarque 4.3.12. Il faut bien distinguer la signification des symboles $\models$ et $\vdash$. On écrit $\{\phi_1, \ldots, \phi_n\} \models \psi$ si pour toute valuation v, nous avons $v(\phi_1 \wedge \ldots \wedge \phi_n \Rightarrow \psi) = 1$. On écrit $p_1, \ldots, p_n \vdash T$ pour signifier que les premisses (hypothèses) $p_1, \ldots, p_n$ entraînent comme conséquence syntaxique le théorème T.

Les **règles d'inférence** permettant de faire les déductions syntaxiques sont :

combinaison	$A, B \vdash A \wedge B$
simplificaion	$A \wedge B \vdash B$
addition	$A \vdash A \vee B$
modus ponens	$A, A \Rightarrow B \vdash B$
modus tollens	$\neg B, A \Rightarrow B \vdash \neg A$
syllogisme hypothétique	$A \Rightarrow B, B \Rightarrow C \vdash A \Rightarrow C$
syllogisme disjonctif	$A \vee B, \neg B \vdash A$
règle des cas	$A \Rightarrow B, \neg A \Rightarrow B \vdash B$
élimination de l'équivalence	$A \Leftrightarrow B \vdash A \Rightarrow B$
introduction de l'équivalence	$A \Rightarrow B, B \Rightarrow A \vdash A \Leftrightarrow B$
incohérence	$A, \neg A \vdash B$.

7. PRÉMISSE, subst. fém., du latin *præmissa*, Chacune des deux propositions, majeure et mineure, d'un syllogisme, d'un raisonnement. En anglais on dit PREMISS (UK) ou PREMIS (US), pluriel premisses ou premises. À ne confondre
— ni avec PRÉMICES, subst. fém. pl., du latin *primitiæ*, A. Premiers fruits de la terre, premiers nés d'un troupeau, B. Au fig. Début, commencement, première manifestation d'un phénomène,
— ni avec le nom pluriel anglais PREMISES (s'orthographiant comme le pluriel US de PREMIS), signifiant locaux, bâtiment ou plus généralement lieux.

Théorème 4.3.13. *Si $A, P \vdash B$ alors $A \vdash P \Rightarrow B$.*

Il est utile de rappeler ici que les règles d'inférence ci-dessus sont celles de la logique propositionnelle classique. D'autres logiques sont possibles, par exemple la logique de Łukasziewicz, dont la sémantique est définie par une valuation floue (à valeurs dans $[0,1]$) [30], ou encore la logique modale [4, 10], faisant usage de deux connecteurs logiques supplémentaires par rapport à la logique booléenne, l'opérateur de nécessité $\Box$ et l'opérateur de possibilité $\Diamond$.

4.3.2 Logique du premier ordre (ou calcul des prédicats)

Comme pour la logique propositionnelle, on distingue les aspects syntaxiques des aspects sémantiques de la logique du premier ordre [8]. Syntaxiqement, la logique du premier ordre est un langage plus expressif et plus flexible que la logique propositionnelle, comportant — outre les variables et les connecteurs logiques — des quantificateurs logiques (universel et existentiel), des connexions par des relations et des évaluations par des fonctions. Un exemple de formule de la logique du premier ordre est la formule (écrite sous forme métamathématique) comme

$$\forall x_1 \forall x_2 (x_1 \leq x_2 \Leftrightarrow \exists x_3 (x_1 + x_3) = x_2).$$

En stricte syntaxe de la logique du premier ordre, cette formule sera remplacée par

$$\forall x_1 \forall x_2 (\mathsf{leq}(x_1, x_2) \Leftrightarrow \exists x_3 (\mathsf{eq}(\mathsf{add}(x_1, x_3), x_2)),$$

où l'on fait usage d'une fonction à deux arguments add qui exécute l'addition de ses arguments, et deux relations : leq qui est la relation « plus petit que » et eq qui est la relation identité.

La logique du premier ordre est un langage sur l'alphabet

$$\mathbb{A} = (\mathbb{A}_v \sqcup \mathbb{A}_l \sqcup \mathbb{A}_s \sqcup \mathbb{A}_q) \sqcup (\mathbb{A}_c \sqcup \sqcup_{n\geq 1} \mathcal{R}_n \sqcup \sqcup_{n\geq 1} \Phi_n),$$

décomposé en

une partie ubiquitaire i.e. constitutive de tout langage, décomposée en quatre sous-alphabets
- — **des variables** regroupées dans l'ensemble dénombrable $\mathbb{A}_v = \{v_1, v_2, \ldots\}$,
- — **des connecteurs logiques** $\mathbb{A}_l$ comme dans §4.3.1,
- — **des séparateurs** $\mathbb{A}_s$ comme dans §4.3.1,
- — **des quantificateurs** universel et existentiel , $\mathbb{A}_q = \{\forall, \exists\}$;

8. Pour l'information du lecteur, mentionnons un exemple de problème relevant de la logique du second ordre. Soit $\mathbb{X}$ un ensemble dénombrablement infini. Si on veut exprimer que $\mathbb{X}$ est bien ordonné, nous devons énoncer une proposition de la forme toute partie non-vide $B \subseteq \mathbb{X}$ admet un minimum. Or le quantificateur universel $\forall B \in \mathcal{P}(\mathbb{X}) \setminus \{\emptyset\}$ porte sur un ensemble non-dénombrable. Des propositions de ce type ne seront pas étudiées dans ce cours.

une partie spécifique composée
- de l'ensemble dénombrable $\mathbb{A}_c$ **des constantes**,
- **une suite d'ensembles de relations** $(\mathcal{R}_n)_{n\geq 1}$ où $\mathcal{R}_n$ est un ensemble dénombrable de **relations** n-aires,
- **une suite d'ensembles de fonctions** $(\Phi_n)_{n\geq 1}$ où Φ_n est un ensemble dénombrable de **symboles fonctionnels** à n arguments.

Dans la grande majorité des cas, l'ensemble $\mathbb{A}_c$ contient un très petit nombre de constantes et les ensembles $\mathcal{R}_n$ et Φ_n sont vides pour la plupart des entiers n.

Définition 4.3.14.
1. Les **termes** $\mathcal{T} := \mathcal{T}(\mathbb{A})$ entrant dans la logique du premier ordre est un langage sur $\mathbb{A}$ défini comme

$$\mathcal{T} := \mathcal{T}(\mathbb{A}) = \cup_{n\in\mathbb{N}}\mathcal{T}_k(\mathbb{A}),$$

où $(\mathcal{T}_k)_{k\geq 0}$ est une suite croissante définie par la récurrence suivante :
— $\mathcal{T}_0 = \mathbb{A}_v \sqcup \mathbb{A}_c$ et, pour $k \geq 0$,
— $\mathcal{T}_{k+1} = \mathcal{T}_k \sqcup \cup_{n\geq 1}\{ft_1t_2\cdots t_n; f \in \Phi_n, t_1 \in \mathcal{T}_k, \ldots, t_n \in \mathcal{T}_k\}$.
2. Un mot α de $\mathbb{A}^*$ est un **formule atomique** si, et seulement si, il existe un entier $n \geq 1$, un symbole de relation n-aire $R \in \mathcal{R}_n$ et n termes $t_1, \ldots, t_n$ de $\mathcal{T}$ tels que $\alpha = Rt_1\cdots t_n$. L'ensemble de formules atomiques est noté $\mathcal{A}(\mathbb{A})$.
3. On définit l'ensemble $\mathcal{F}$ de **formules** de la logique comme

$$\mathcal{F} := \mathcal{F}(\mathbb{A}) = \cup_{k\geq 0}\mathcal{F}_k,$$

où $(\mathcal{F}_k)_{k\geq 0}$ est une suite croissante définie par la récurrence suivante :
— $\mathcal{F}_0 = \mathcal{A}(\mathbb{A})$ et
— pour $k \geq 0$,

$$\begin{aligned}\mathcal{F}_{k+1} = \mathcal{F}_k &\sqcup \{\neg F; F \in \mathcal{F}_k\}\\ &\sqcup \{(F\lambda G), F \in \mathcal{F}_k, G \in \mathcal{F}_k, \lambda \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}\\ &\sqcup \{\forall v_m F, F \in \mathcal{F}_k, m \in \mathbb{N}\} \sqcup \{\exists v_m F, F \in \mathcal{F}_k, m \in \mathbb{N}\}.\end{aligned}$$

Souvent on élargit la famille de fonctions en incluant les fonctions Φ_0 à zéro entrée ; il s'agit bien sûr des constantes ; on peut ainsi absorber le sous-alphabet des constantes $\mathbb{A}_c$ dans Φ_0 et considérer donc la famille $(\Phi_n)_{n\in\mathbb{N}}$.

Maintenant, pour chaque $n \in \mathbb{N}$, l'ensemble Φ_n est dénombrable (fini ou infini). On peut donc énumérer ses éléments : $\Phi_n = \{f_{n,1}, f_{n,2}, \ldots\}$, avec $f_{n,l}$ fonction de n arguements. On peut faire de même pour les relations n-aires $\mathcal{R}_n = \{r_{n,1}, r_{n,2}, \ldots\}$ pour $n \geq 1$. On peut ainsi caractériser les différents ensembles $(\mathcal{T}, \mathcal{A}, \mathcal{F})$ ci-dessus en présentant les productions de leurs grammaires respectives (sous BNF).

Le langage des termes $\mathcal{T}$ peut aussi être décrit en termes de sa gram-

maire

$$\begin{aligned}
V &\to v_1|v_2|\cdots\\
F_0 &\to c_1|c_2|\cdots\\
F_1 &\to f_{1,1}|f_{1,2}|\cdots\\
F_2 &\to f_{2,1}|f_{2,2}|\cdots\\
&\vdots\\
T &\to V|F_0|F_1(T)|F_2(T,T)|F_3(T,T,T)|\cdots.
\end{aligned}$$

Comme le langage des termes est défini en notation polonaise inversée (qui rend l'utilisation de parenthèses superflue), il est en général difficile pour un humain de vérifier aisément qu'un mot $\tau \in \mathbb{A}^*$ qui semble provenir de la grammaire ci-dessus est effectivement un terme. Il y a cependant un algorithme de vérification très simple à mettre en œuvre pour conclure.

Définition 4.3.15. Soit s un symbole dans $\mathbb{T} := \mathbb{A}_c \sqcup \mathbb{A}_v \sqcup (\sqcup_{n\in\mathbb{N}}\Phi_n)$. On définit le **poids** $\mathsf{wt}(s)$ du symbole s par la formule

$$\mathsf{wt}(s) = \begin{cases} n-1 & \text{si } s \in \Phi_n \\ -1 & \text{si } s \in \mathbb{A}_c \sqcup \mathbb{A}_v. \end{cases}$$

Le poids s'étend naturellement aux mots $\sigma \in \mathbb{T}^*$ par

$$\mathbb{T}^* \ni \sigma = s_1 s_2 \cdots s_n \mapsto \mathsf{wt}(\sigma) = \sum_{i=1}^{n} \mathsf{wt}(s_i).$$

Lemme 4.3.16. *Un mot $\tau \in \mathbb{T}^*$ sera un terme si $\mathsf{wt}(\tau) = -1$ et tout segment initial $\tau\restriction_k$ (pour $1 \le k < |\tau|$) de τ a un poids $\mathsf{wt}(\tau\restriction_k) \ge 0$ pour tout $1 \le k < |\tau|$.*

Exemple 4.3.17. Soient $f \in \Phi_1$, $g \in \Phi_3$, $x, y, z \in \mathbb{A}_v$ et $c, d \in \mathbb{A}_c$. Alors

$$\tau = ggffxgzycfdffgfcgyfzffdfcfc \in \mathcal{T}.$$

Comme le langage $\mathcal{T}$ est dénombrable, ses éléments sont énumérables : $\mathcal{T} = \{t_1, t_2, \ldots\}$. En utilisant la dénombrabilité de $\mathcal{T}$, on peut décrire la grammaire du langage de formules atomiques $\mathcal{A}$ comme suit :

$$\begin{aligned}
T &\to t_1|t_2|\cdots\\
R_0 &\to r_{0,1}|r_{0,2}|\cdots\\
R_1 &\to r_{1,1}|r_{1,2}|\cdots\\
R_2 &\to r_{2,1}|r_{2,2}|\cdots\\
&\vdots\\
A &\to T = T|R_1(T)|R_2(T,T)|R_3(T,T,T)|\cdots.
\end{aligned}$$

Comme le langage des formules atomiques est dénombrable, ses éléments peuvent être énumérés $\mathcal{A} = \{a_1, a_2, \ldots\}$. On conclut que la grammaire du langage $\mathcal{F}$ est décrite par les productions :

$$V \to v_1 | v_2 | \cdots$$
$$A \to a_1 | a_2 | \cdots$$
$$S \to A | \neg(S) | | (S) \wedge (S) | (S) \vee (S) | (S) \Rightarrow (S) | (S) \Leftrightarrow (S) | \exists V(S) | \forall V(S).$$

Théorème 4.3.18. *Pour toute formule $\phi \in \mathcal{F}$, un et un seul des cinq cas suivants se présente :*

1. $\phi \in \mathcal{A}$,
2. $\exists! \psi \in \mathcal{F} : \phi = \neg\psi$,
3. $\exists! (\chi, \psi) \in \mathcal{F}^2 : \phi = \chi \lambda \psi$, *avec* $\lambda \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}$,
4. $\exists! k \in \mathbb{N}, \exists! \psi \in \mathcal{F} : \phi = \forall v_k \psi$ *et*
5. $\exists! k \in \mathbb{N}, \exists! \psi \in \mathcal{F} : \phi = \exists v_k \psi$.

Définition 4.3.19. (Variables libres et liées). Soit $\phi \in \mathcal{F}$.

— Si ϕ est une formule atomique, alors toutes les occurrences de v_k dans ϕ sont libres.
— Si $\phi = \neg\psi$, les occurrences libres de v_k dans ϕ le sont dans ψ.
— Si $\phi = \chi \lambda \psi$, avec $\lambda \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}$, les occurrences libres de v_k dans ϕ le sont aussi dans χ et dans ψ.
— Si $\phi = \forall v_l \psi$ ou $\phi = \exists v_l \psi$, pour tous les $k \neq l$, les occurrences libres de v_k dans ϕ le sont aussi dans ψ. Aucune des occurrences de v_l dans ϕ n'est libre. Les occurrences non-libres d'une variable sont appelées **liées**. Les variables **libres** dans ϕ sont les variables qui admettent au moins une occurrence libre dans ϕ. Une formule qui ne contient aucune variables libre est appelée **close**.
— Soient $\psi[v_1, \ldots, v_k]$ une formule avec k variables distinctes libres et $t_1, \ldots, t_k \in \mathcal{T}$. On note $\psi[v_1 \to t_1, \ldots, v_k \to t_k]$ la formule où chaque occurrence de v_j est remplacée par t_j, pour $j = 1, \ldots, k$. Cette opération est appelée **substitution**.

Exemple 4.3.20. [L'énoncé du théorème de Fermat]. En utilisant une notation méta-linguistique non-canonique, on écrirait la méta-formule

$$F = \neg(\exists w \exists x \exists y \exists z((x+1)^{w+3} + (y+1)^{w+3} = (z+1)^{w+3})),$$

où les variables w, x, y, z balaient l'ensemble $\mathbb{N}$. Cette expression donne lieu en une formule wff, écrite dans l'alphabet $\mathbb{A}_v = \mathbb{N}$, $\mathbb{A}_c = \{1, 3\}$, $\Phi_2 = \{+, \times\}$, $\mathcal{R}_2 = \{I\}$. On doit d'abord exprimer l'exponentiation comme une multiplication. Afin de simplifier l'écriture, on suppose qu'il existe une fonction supplémentaire $\exp \in \Phi_2$ dont le premier argument serait la base et le deuxième l'exposant. On doit alors écrire

$$\neg(\exists w \exists x \exists y \exists z(I + \exp + x1 + w3 \exp + y1 + w3 \exp + z1 + w3)).$$

La formule F est dite close ; elle ne contient que des **variables liées** par des quantificateurs. Le théorème de Fermat peut aussi s'exprimer à l'aide de la méta-formule $G(w)$, avec

$$G(w) = \neg(\exists x \exists y \exists z((x+1)^{w+3} + (y+1)^{w+3} = (z+1)^{w+3})),$$

par $F = \forall w G(w)$; la formule $G(w)$ contient une **variable libre**, w, qui n'est pas sujette à un quantificateur ; la formule F est sa clôture universelle (car elle fait intervenir le quantificateur universel) de $G(w)$. Si t est un terme, on note $G[w \to t]$ la formule qui s'obtient en substituant la variable libre w par t. Par exemple, si $t = v \times v$, alors

$$G[w \to t] = \neg(\exists x \exists y \exists z((x+1)^{v^2+3} + (y+1)^{v^2+3} = (z+1)^{v^2+3})),$$

Étant donné que l'écriture canonique imposée par le langage $\mathcal{F}$ est assez fastidieuse et difficile à lire pour un humain, nous utiliserons très souvent les méta-formules qui mélangent la notation logique avec la notation mathématique usuelle.

4.4 Structures, modèles, théories, déductions

Une **structure** est un multiplet, composé d'un ensemble de base, de relations et d'opérations internes et (éventuellement) d'éléments distingués. Par exemple $\langle \mathbb{Z}, + \rangle$ ou $\langle \mathbb{Z}, +, 0 \rangle$ désigne une structure qu'on appelle groupe abélien (additif) des entiers naturels avec élément neutre 0. De même, $\langle \mathbb{R}, \leq, +, \times \rangle$ ou $\langle \mathbb{R}, \leq, +, \times, 0, 1 \rangle$ désigne le corps ordonné des réels[9]. Les formules introduites dans le paragraphe précédent permettront de décrire les propriétés de ces structures.

Si l'on veut signifier que 0 est l'élément neutre de la structure définie par le groupe additif $\langle \mathbb{Z}, +, 0 \rangle$, on introduit la constante z (pour zéro), la fonction $f \in \Phi_2$ correspondant à l'addition en notation polonaise et $I \in \mathcal{R}_2$ la relation identité. On aura alors que la formule

$$\phi = \forall v(Ifvzv \wedge Ifzvv)$$

est satisfaite dans la structure étudiée.

Définition 4.4.1. Soit le langage des formules $\mathcal{L} := \mathcal{F}(\mathbb{A})$.

1. On appelle réalisation de ce langage, ou $\mathcal{L}$-structure, un multiplet $\mathcal{M}$ composé
   — d'un ensemble non-vide $\mathbb{M}$, appelé **ensemble de base** de la réalisation,
   — pour chaque constante $c \in \mathbb{A}_c$, d'un élément $\overline{c} \in \mathbb{M}$, appelé **interprétation** de c dans la réalisation,

9. La deuxième écriture précise que ce corps possède un élément distingué (0) le neutre pour l'addition et un autre élément (1) le neutre pour la multiplication.

— pour chaque fonction $f \in \Phi_k$ ($k \geq 1$), d'une application $\overline{f} : \mathbb{M}^k \to \mathbb{M}$ appelée **interprétation** de f dans la réalisation,
— pour chaque relation $R \in \mathcal{R}_k$ ($k \geq 1$), d'une partie $\overline{R} \subseteq \mathbb{M}^k \times \mathbb{M}^k$ (i.e. d'une relation sur $\mathbb{M}^k$) appelée **interprétation** de R dans la réalisation.

2. Si $F := F[v_0, \ldots, v_{n-1}]$ est une formule ouverte de $\mathcal{L}$, $\mathcal{M}$ une $\mathcal{L}$-structure et $m_0, \ldots, m_{n-1} \in \mathbb{M}$, on dira que F est **satisfaite** dans la structure $\mathcal{M}$ lorsque les variables sont interprétées par les m_i, i.e. $\overline{v}_i = m_i$ pour tout $i = 0, \ldots, n-1$. On note

$$\langle \mathcal{M}; v_0 \to m_0, \ldots, v_{n-1} \to m_{n-1} \rangle \vDash F.$$

3. Une **théorie** T est un ensemble distingué de formules de $\mathcal{L}$. On dit qu'une $\mathcal{L}$-structure $\mathcal{M}$ est un **modèle** de T ou T est satisfaite par $\mathcal{M}$, noté $\mathcal{M} \vDash T$ si toute formule de T est sastisfaite par $\mathcal{M}$.

Le moment est arrivé de se poser la question : qu'est-ce une preuve mathématique ? C'est déduire une proposition F d'un ensemble de propositions préalablement admises (axiomes, théorèmes déjà démontrés, hypothèses) en suivant une suite de règles de déduction formelle (i.e. des productions de ce langage) et des axiomes logiques.

Les **règles de déduction** formelle utilisées en logique du premier ordre sont :

1. le *modus ponendo ponens* [10] qui consiste à inférer des propositions $F \Rightarrow G$ et de F le séquent (conséquence syntaxique) G, noté

$$F \Rightarrow G, F \vdash G;$$

2. la règle de généralisation : si on sait démontrer $F[v]$ sans hypothèse particulière sur v, alors on aura $\forall v F[v]$.

Les **axiomes logiques** sont

1. les tautologies (comme, par exemple, $\neg(\neg F) \Leftrightarrow F$, $F \vee G \Rightarrow F$) et
2. les axiomes des quantificateurs
   — $\neg \forall v F[v] \Leftrightarrow \exists v \neg F[v]$,
   — si v n'a pas d'occurrence libre dans F, on a

$$\forall v(F \Rightarrow G[v]) \Rightarrow (F \Rightarrow \forall v G[v]).$$

   — $\forall v F[v] \Rightarrow F[v \to t]$, pour tout terme t.

Définition 4.4.2. Soient T une théorie et F une formule du langage $\mathcal{L}$. Une **démonstration formelle** de F dans T est une suite finie de formules $(F_0, F_1, \ldots, F_n)$ de $\mathcal{L}$ se terminant par F (i.e. $F_n = F$) et telle que chaque F_i pour $i = 0, \ldots, n$ satisfait l'une au moins de conditions :

1. $F_i \in T$,
2. F_i est un axiome logique,

10. MODUS PONENDO PONENS, loc. lat. qui affirme en affirmant.

3. F_i se déduit d'une ou de deux formules qui le précèdent par l'une des deux règles de déduction.

S'il existe une démonstration de F dans T, on dit que F est **démontrable** dans T et on note $T \vdash F$.

4.5 Intermède : arithmétiques de Robinson et de Peano

Gödel a montré l'incomplétude d'une théorie contenant l'arithmétique des entiers naturels, i.e. celle découlant des axiomes de Peano. Ultérieurement, l'incomplétude même d'une arithmétique plus rudimentaire, découlant des axiomes de Robinson [65], a été démontrée. L'alphabet des arithmétiques de Peano et de Robinson — au delà des parties $\mathbb{A}_l$, $\mathbb{A}_s$ et $\mathbb{A}_q$ communes à toute logique — comporte les parties suivantes :

— un ensemble dénombrable de variables $\mathbb{A}_v$,
— un singleton contenant l'unique constante $\mathbb{A}_c = \{0\}$,
— une fonction $S \in \Phi_1$, la fonction « successeur de »,
— deux fonctions $+, \times \in \Phi_2$, l'addition et la multiplication,
— une relation $I \in \mathcal{R}_2$, l'identité.

Dans un souci de lisibilité de formules, on utilisera dans la suite une écriture plus proche des formules habituelles. Par exemple, nous écrirons $a + b$ au lieu de $+ab$ ou $s = t$ au lieu de Ist.

Les termes de la théorie sont des expressions construites à partir de la constante 0 et les variables ; ils sont obtenus par application de la fonction unaire S et des fonctions binaires $+$ et $\times$. Par exemple $SSS0$, $(SS0 + x)$, $(S0 \times (Sx + y))$ sont des termes. Les termes ne contenant aucune variable sont appelés fermés. Les termes numériques de l'arithmétique sont les termes construits par application successive de la fonction S sur la constante 0. Ils sont notés par un surlignement $\overline{n} = S \cdots S0$ (pour les distinguer des variables n).

Axiome 4.5.1 (Arithmétique de Robinson (RA))**.** *L'arithmétique est décrite par*

1. $\forall x(0 \neq Sx)$ *(0 n'est pas un successeur),*
2. $\forall x \forall y((Sx = Sy) \Rightarrow (x = y))$ *(injectivité de la fonction S),*
3. $\forall x((x \neq 0) \Rightarrow \exists y(x = Sy))$ *(il n'existe pas de pseudo-zeros, i.e. des objets différents de 0 qui ne sont pas successeurs),*
4. $\forall x(x + 0 = x)$ *(neutralité de 0 pour l'addition),*
5. $\forall x \forall y(x + Sy = S(x + y))$,
6. $\forall x(x \times 0 = x)$,
7. $\forall x \forall y(x \times Sy = (x \times y) + x)$.

Le fait que l'affirmation $\forall x(0 \neq Sx)$ ne soit pas démontrable dans RA, montre que cette dernière est une théorie très rudimentaire. En ajoutant cependant cette propriété comme axiome, on peut établir une relation d'ordre sur RA.

L'arithmétique de Peano (PA) est obtenue de RA en ajoutant l'axiome de récurrence :

$$(\phi(0) \wedge \forall x(\phi(x) \Rightarrow \phi(Sx))) \Rightarrow \forall x \phi(x).$$

On peut cependant montrer que cet axiome rend alors superflue l'affirmation $\forall x((x \neq 0) \Rightarrow \exists y(x = Sy))$ (qui devient un execice !) On peut alors enlever 3 de la liste des affirmations de PA.

4.6 Le premier théorème d'incomplétude et les idées essentielles de sa démonstration

On présente ci-dessous les idées essentielles [11] de la preuve du premier théorème d'incomplétude. La preuve de Gödel est très technique et difficile à lire. Une des difficultés est due au fait que nous utilisons aujourd'hui une notation différente pour plusieurs formules de l'article. Mais l'idée de base de la preuve est que le langage de l'arithmétique est dénombrable, on peut donc numéroter toutes les formules du langage. La deuxième difficulté de l'article de Gödel est s'assurer que la traduction des formules en des entiers naturels est cohérente et que le passage des entiers à des formules se fait de manière canonique.

Nous omettons cette deuxième partie difficile et fastidieuse (mais essentielle) et nous supposons, pour simplifier, que les fonctions propositionnelles à une variable entière w sont rangées dans un ordre lexicographique que nous numérotons : $(P_n[w], n \in \mathbb{N})$, pour tout $w \in \mathbb{N}$. En admettant que P_n est syntaxiquement correcte, la proposition $P_n[w]$ sera une affirmation entre les entiers n et w. Les mots qui constituent la preuve d'une proposition sont aussi des propositions rangées $(\Pi_x, x \in \mathbb{N})$.

Soit $Q[w] = \neg\exists x(\Pi_x \text{ démontre } P_w[w])$. $Q[w]$ est une fonction propositionnelle qui ne dépend que de w. Mais toutes les fonctions sont énumérées, $P_0[w], P_1[w], \ldots$. Par conséquent, $Q[w]$ doit être l'une des fonctions de la liste, i.e. $Q[w] = P_k[w]$ pour un certain k. Examiner cette égalité pour $w = k$:

$$P_k[k] = Q[k] = \neg\exists x(\Pi_x \text{ démontre } P_k[k]).$$

S'il existait une démonstration de $P_k[k]$ dans la théorie, alors $P_k[k]$ serait fausse. Mais l'arithmétique est cohérente donc on n'y peut pas démontrer des propositions fausses. Contradiction [12]. Si $P_k[k]$ est fausse, alors $\neg P_k[k] = \neg Q[k]$ est vraie, ce qui signifie qu'il existe une preuve de $P_k[k]$ et nous arrivons encore une fois à une contradiction.

11. Kurt Gödel a renié les deux premières traductions anglaises de son article [28] car il pensait qu'elles trahissaient sa pensée. L'auteur de ces lignes se garde donc bien de prétendre qu'elles constituent la transcription fidèle de la pensée de Gödel ....

12. Contradiction qui ressemble au paradoxe du philosophe crétois Épiménide (7e s. av. notre ère) qui énonçait « les crétois mentent toujours » ou au paradoxe d'Euboulide (4e s. av. notre ère) qui de manière encore plus paradoxale énonçait : « cette proposition est fausse ».

4.7 Notes

L'argument de contradiction utilisé dans la preuve du premier théorème de Gödel nous le retrouverons dans la preuve d'indécidabilté d'une machine de Turing. En fait, le théorème de Turing est une autre démonstration du théorème de Gödel.

La logique mathématique est un domaine fascinant. À part les livres [1, 17, 18, 65] déjà mentionnés, il est instructif de consulter l'excellent livre de vulgarisation scientifique [54] écrit par un physicien théoricien et mathématicien de renom et excellent pédagogue, Sir Roger Penrose [13]. Dans un registre plus ludique, on peut consulter la bande dessinée [25] dont le scénario — écrit conjointement par un mathématicien-romancier et un informaticien théoricien — relate les développements historiques des fondements des mathématiques, de la logique et des débuts de l'informatique théorique.

La théorie des treillis est développée dans [3] et les relations de treillis avec les probabilités classiques ou quantiques dans [71, 57, 55].

Une revue très pédagogique des différentes variétés de logiques non-standard est l'article [21].

13. Roger Penrose (né à Colchester en 1931) est un mathématicien, physicien mathématicien et philosophe des sciences britannique, récompensé par le prix Nobel de Physique en 2020 pour ses travaux sur la formation des trous noirs. Il a été anobli (*knighted*, acquis le titre de Sir) en 1994 par la Reine d'Angleterre « *for services to science* ».

5

Automates à nombre fini d'états et langages réguliers

Nous utilisons tous d'automates dans la vie de tous les jours (feux de circulation, programmation d'un lave-linge ou lave-vaisselle, tourniquets de métro — non rennais ! — etc.). Certains d'entre eux sont tellement rudimentaires qu'ils n'ont pas d'entrée, ils se contentent de répéter indéfiniment le même cycle (ex. feux de circulation) ; d'autres ont besoin d'une entrée, par exemple l'automate qui régit l'ouverture du tourniquet a besoin de votre ticket de métro en entrée : il vérifie que votre ticket est valide et si oui commande l'ouverture du tourniquet, sinon il émet un son (de rejet).

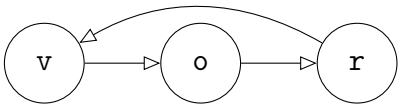
Nous allons nous intéresser à des automates plus élaborés qui permettent de vérifier si un mot appartient à un langage donné. Dans ce chapitre nous étudions les automates finis qui sont les automates non-triviaux les plus simples. Cette étude nous permettra d'aborder progressivement les automates plus généraux tels que les machines de Turing qui seront étudiées au chapitre suivant. Le point culminant de notre étude des automates sera de montrer que l'*Entscheidungsproblem*, posé par Hilbert, n'est pas décidable par une machine de Turing. Ce résultat de non-décidabilité par un automate de Turing est une autre démonstration, plus transparente, du premier théorème d'incomplétude de Gödel, énoncé en §1.2.1 et dont une démonstration est esquissée en §4.6.

Cependant, même si l'intuition de Hilbert était fausse, elle a été très fructueuse car elle a initié toute une ligne de pensée : commencée avec les travaux de Church [13] sur le λ-calcul a eu comme point culminant la notion de « machine » introduite par Turing [67]. Ces deux approches, développées indépendamment, mais ultérieurement démontrées équivalentes et connues sous le nom de « thèse de Church-Turing », servent aujourd'hui de modèle théorique de tout algorithme qui peut être exécuté sur un ordinateur

pour donner des résultats en temps fini. Tout algorithme que nous pouvons concevoir peut être en fait modélisé par une machine de Turing. Les machines de Turing générales seront étudiées au chapitre suivant. Ici, nous introduisons la notion d'automate fini (instantiation la plus simple d'une machine de Turing).

5.1 Automates finis déterministes

Avant de donner la définition d'un automate fini général, commençons par l'automate rudimentaire sans entrée qui décrit le fonctionnement d'un feu rouge. Nous avons un espace fini d'états $\mathbb{X} = \{\mathtt{v},\mathtt{o},\mathtt{r}\}$ et une application de transition $\delta : \mathbb{X} \to \mathbb{X}$ définie de *manière équivalente*[1] soit par le vecteur $(\delta(x))_{x\in\mathbb{X}}$ soit par une matrice stochastique déterministe $D \in \mathcal{M}_{|\mathbb{X}|\times|\mathbb{X}|}(\{0,1\})$ définie par $D_{x,y} = \mathbb{1}_{\{y\}}(\delta(x)), x, y \in \mathbb{X}$. Si la suite $(X_n)_{n\in\mathbb{N}}$ décrit l'état du système et si le système commence de manière déterministe avec l'état initial $X_0 = x_0 \in \mathbb{X}$, son évolution est une chaîne de Markov triviale i.e. déterministe appelée **système dynamique** sur $\mathbb{X}$. On a $X_n = \delta(X_{n-1}), n \geq 1$ ou de manière équivalente $X_n = \delta^n(x_0)$ où $\delta^n := \underbrace{\delta \circ \cdots \circ \delta}_{n \text{ termes}}$ (avec la convention $\delta^0 = \mathrm{id}$. En termes probabilistes[2], nous avons la situation triviale $\mathbb{P}_{x_0}(X_n = \delta^n(x_0)) = 1$, pour tout $n \geq 0$ Étant donné que l'ensemble $\mathbb{X}$ est fini et $\delta : \mathbb{X} \to \mathbb{X}$, alors le système dynamique (X_n) devient *nécessairement* périodique à partir d'un certain instant (en l'occurrence il l'est dès le début dans cet exemple très simple). Par ailleurs, étant donné que l'évolution est donnée par la matrice D, on peut la décrire de manière équivalente par le digraphe $\mathbb{G} = \mathbb{G}(D)$ associé à la relation D.

$\mathbb{G}(D) =$ 

Chaque élément de l'ensemble $G^*_{x_0}$ des trajectoires émanant de x_0 est appelé **trajectoire calculatoire** de l'automate initiée à x_0.

Considérons maintenant le cas d'automates déterministes finis avec entrée. De nouveau l'ensemble fini $\mathbb{X}$ désignera l'ensemble des états de l'automate ; mais maintenant nous avons aussi un ensemble fini $\mathbb{A}$ qui représente l'alphabet des mots de l'entrée. La fonction de transition sera maintenant une application $\delta : \mathbb{X} \times \mathbb{A} \to \mathbb{X}$. Au lieu de considérer une seule fonction δ définie sur $\mathbb{X} \times \mathbb{A}$, nous pouvons de manière équivalente considérer une famille $(\delta_a)_{a\in\mathbb{A}}$ — indexée par $\mathbb{A}$ — de fonctions $\delta_a : \mathbb{X} \to \mathbb{X}$ définies pour tout $x \in \mathbb{X}$ et tout $a \in \mathbb{A}$ par $\delta_a(x) = \delta(x, a)$. Maintenant, pour chaque

1. Ceci est un résultat très général : toute fonction mesurable f sur un espace mesurable $(\mathbb{X}, \mathcal{X})$ (qui est un espace standard de Borel) est *équivalente* à un noyau stochastique déterministe $K : \mathbb{X} \times \mathcal{X} \to \{0,1\}$ défini par $K(x, A) = \mathbb{1}_A(f(x)) = \varepsilon_{f(x)}(A)$ pour tout $x \in \mathbb{X}$ et tout $A \in \mathcal{X}$ où ε désigne la masse de Dirac. Cette écriture s'appelle *représentation spectrale* de f.

2. Cf. cours *Probabilités pour la théorie de l'information*.

$a \in \mathbb{A}$ la fonction δ_a est équivalente à une matrice stochastique déterministe D_a qui induit une relation sur $\mathbb{X}$, dont toutes les arêtes sont relatives à la lettre $a \in \mathbb{A}$. Chaque $a \in \mathbb{A}$ donne donc naissance à un digraphe $\mathbb{G}(D_a)$ ayant d'ensembles d'arêtes $\mathbb{G}^1(D_a)$ différents mais le même ensemble de sommets $\mathbb{G}^0(D_a) = \mathbb{X}$. En superposant tous les digraphes $\mathbb{G}(D_a)$ ainsi obtenus les arêtes aquèrent une étiquette qui les différencie ; certains sommets peuvent alors être reliés par des arêtes multiples différenciées par leurs étiquettes ; au lieu de les dessiner séparément, on dessine une seule arête dirigée portant toutes les étiquettes correspondantes.

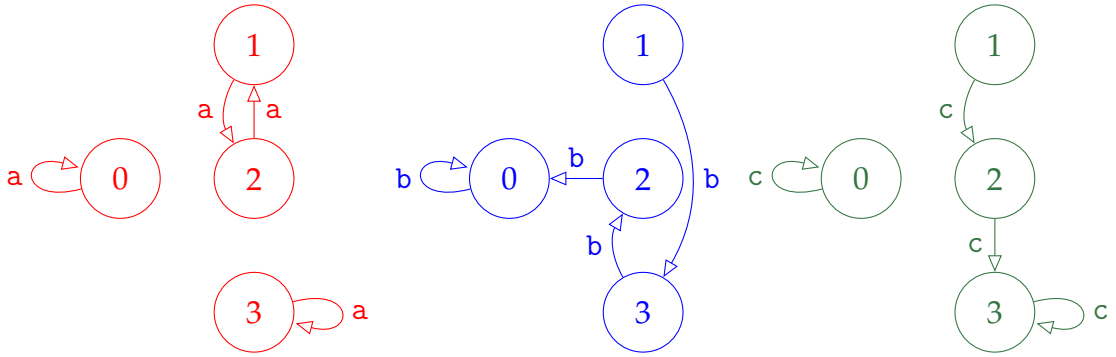
Exemple 5.1.1. Soient $\mathbb{X} = \{0,1,2,3\}$, $\mathbb{A} = \{\mathtt{a},\mathtt{b},\mathtt{c}\}$ et δ définie par

x	$\delta_a(x)$	$\delta_b(x)$	$\delta_c(x)$
0	0	0	0
1	2	3	2
2	1	0	3
3	3	2	2

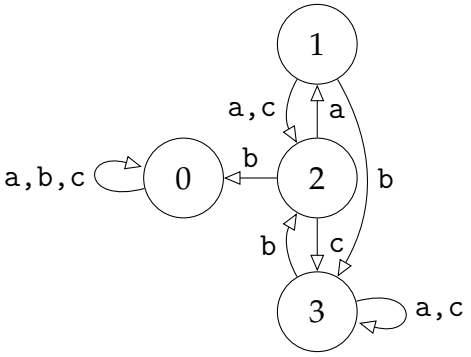
ou encore, de manière équivalente, par les matrices stochastiques déterministes

$$D_a = \begin{pmatrix} 1&0&0&0\\ 0&0&1&0\\ 0&1&0&0\\ 0&0&0&1 \end{pmatrix} ; D_b = \begin{pmatrix} 1&0&0&0\\ 0&0&0&1\\ 1&0&0&0\\ 0&0&1&0 \end{pmatrix} ; D_c = \begin{pmatrix} 1&0&0&0\\ 0&0&1&0\\ 0&0&0&1\\ 0&0&1&0 \end{pmatrix}.$$

Chacune des matrices ci-dessus définit une relation sur $\mathbb{X}$ — avec $\mathrm{dom}(D_l) = \mathbb{X}$ pour tout $l \in \mathbb{A}$ — qui induit un digraphe $\mathbb{G}(D_l)_{l\in\mathbb{A}}$.



En superposant les trois digraphes, nous obtenons le digraphe avec des étiquettes multiples sur les arêtes :



Il est évident que la donnée de la fonction de transition δ est équivalente à la donnée du digraphe avec des arêtes multi-étiquetées obtenus par la fonction de transition δ.

Définition 5.1.2. Un **automate fini déterministe** est la donnée $M = (\mathbb{X}, \mathbb{A}, \delta, x_0, \mathbb{F})$, où $\mathbb{X}$ est un ensemble fini d'états, $\mathbb{F} \subseteq \mathbb{X}$ est l'ensemble des états finaux d'acceptation, $\mathbb{A}$ est l'alphabet fini des messages d'entrée, $x_0 \in \mathbb{X}$ est l'état initial, $\delta = (\delta_a)_{a\in\mathbb{A}}$, avec $\delta_a : \mathbb{X} \to \mathbb{X}$ et $\mathrm{dom}(\delta_a) = \mathbb{X}$, est la famille des fonctions de transition. L'ensemble d'automates finis déterministes est noté AFD.
Le digraphe $\mathbb{G}(M)$ de l'automate sera le digraphe avec des arêtes étiquetées provenant des relations $(D_a)_{a\in\mathbb{A}}$ définies par la famille δ. Le **langage** reconnu par M est l'ensemble des mots engendrant des trajectoires calculatoires de $\mathbb{G}^*(M)$ qui rendent accessibles les états de $\mathbb{F}$ depuis l'état initial x_0, i.e.

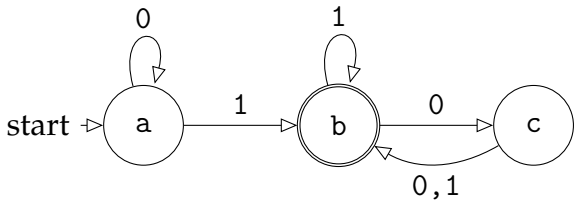
$$\mathcal{L}(M) = \{\boldsymbol{\alpha} \in \mathbb{A}^* : \Delta_{\boldsymbol{\alpha}}(x_0) \in \mathbb{F}\},$$

où $\Delta_{\boldsymbol{\alpha}} = \delta_{\alpha_{|\alpha|}} \circ \cdots \circ \delta_{\alpha_1}$.

Exercice 5.1.3. Soit $M = (\mathbb{X}, \mathbb{A}, \delta, x_0, \mathbb{F})$ un AFD. Pour tout $a \in \mathbb{A}$, on note $D_a \in \mathcal{M}_{|\mathbb{X}|\times|\mathbb{X}|}(\{0,1\})$ la matrice stochastique déterministe codant l'application δ_a.

1. Soit $\boldsymbol{\alpha} = a_1 \cdots a_n$ un mot en entrée de l'automate. Déterminer la matrice $D[\boldsymbol{\alpha}]$ qui code l'application de transfert $\Delta_{\boldsymbol{\alpha}}$.
2. Montrer que pour tout α, la matrice $D[\alpha]$ est une matrice stochastique déterministe. Calculer ses éléments de matrice $D[\boldsymbol{\alpha}]_{x,y}$ pour $x, y \in \mathbb{X}$.
3. Pour $B \subseteq \mathbb{X}$, noter $\mathbf{v}_B \in \{0,1\}^{\mathbb{X}}$ le vecteur (colonne) ayant comme coordonnées $v_B(x) = \mathbb{1}_B(x), x \in \mathbb{X}$.
4. Monter que le mot $\boldsymbol{\alpha}$ est accepté par l'automate si, et seulement si, $\mathbf{v}^t_{\{x_0\}} D[\boldsymbol{\alpha}] \mathbf{v}_{\mathbb{F}} \geq 1$, où $\mathbf{v}^t_B$ désigne le vecteur transposé (ligne) de $\mathbf{v}_B$.

Exemple 5.1.4. Soit l'automate défini par $\mathbb{X} = \{\mathtt{a},\mathtt{b},\mathtt{c}\}$, $\mathbb{A} = \{0,1\}$, $x_0 = a$ et $\mathbb{F} = \{b\}$ dont le digraphe est donné par



L'état initial est signalé par la flèche « start » tandis que les états finaux d'acceptation sont signalés par un double cercle. L'automate ci-dessus accepte le mot $\boldsymbol{\alpha} = 1101$ présenté en entrée tandis qu'il rejette le mot $\boldsymbol{\beta} = 11010$. En fait, si un mot $\boldsymbol{\gamma} = \gamma_1 \cdots \gamma_{|\gamma|} \in \mathbb{A}^*$ est présenté en entrée de l'automate, alors en partant de x_0, l'automate évolue selon $\Delta_{\boldsymbol{\gamma}}(x_0) = \delta_{\gamma_{|\gamma|}} \circ \cdots \circ \delta_{\gamma_1}(x_0)$. Si $\Delta_{\boldsymbol{\gamma}}(x_0) \in \mathbb{F}$, alors le mot $\boldsymbol{\gamma}$ est accepté, sinon il est rejeté. Dans cet exemple $\Delta_{\boldsymbol{\alpha}}(a) = b \in \mathbb{F}$ tandis que $\Delta_{\boldsymbol{\beta}}(a) = c \notin \mathbb{F}$.

Définition 5.1.5. Un langage L est **régulier** s'il existe un automate déterministe fini M qui le reconnaît, i.e. si $L = \mathcal{L}(M)$.

Si L, L_1 et L_2 désignent des langages sur $\mathbb{A}$, on définit leur
— **réunion** : $L_1 \cup L_2 = \{\alpha \in \mathbb{A}^* : \alpha \in L_1 \text{ ou } \alpha \in L_2\}$,
— **concaténation (ou composition)** : $L_1 L_2 := L_1 \circ L_2 = \{\alpha\beta \in \mathbb{A}^* : \alpha \in L_1 \text{ et } \beta \in L_2\}$,
— **clôture monoïdale** : $L^* = \cup_{k\in\mathbb{N}} L^k$, avec $L^0 = \{\varepsilon\}$.

Théorème 5.1.6. *La classe des langages réguliers est fermée par réunion finie.*

Démonstration. Soient L_1 et L_2 deux langages réguliers. Il existe alors deux automates

$$M_1 = (\mathbb{X}^1, \mathbb{A}, \delta^{(1)}, x_0^1, \mathbb{F}^1) \text{ et } M_2 = (\mathbb{X}^2, \mathbb{A}, \delta^{(2)}, x_0^2, \mathbb{F}^2)$$

de AFD tels que $L_1 = \mathcal{L}(M_1)$ et $L_2 = \mathcal{L}(M_2)$. Nous construirons un automate M qui simule M_1 et M_2 et accepte les mots reconnus indifféremment par M_1 ou M_2 *en les lisant une seule fois*. Soit $M = (\mathbb{X}, \mathbb{A}, \delta, x_0, \mathbb{F})$ l'automate défini par $\mathbb{X} = \mathbb{X}^1 \times \mathbb{X}^2$, $\delta_a((x_1, x_2)) = (\delta_a^{(1)}(x_1), \delta_a^{(2)}(x_2))$ pour tout $a \in \mathbb{A}$ et tout $(x_1, x_2) \in \mathbb{X}$, $x_0 = (x_0^1, x_0^2)$ et $\mathbb{F} = (\mathbb{F}^1 \times \mathbb{X}^2) \cup (\mathbb{X}^1 \times \mathbb{F}^2)$. Il est alors évident que M accepte un mot α si soit M_1 soit M_2 l'accepte. □

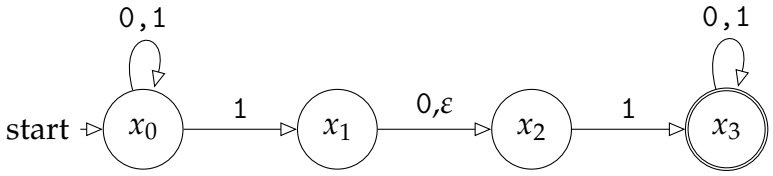
Théorème 5.1.7. *La classe des langages réguliers est fermée par concaténation.*

Nous ne pouvons pas répéter la construction précédente pour montrer la clôture par concaténation (pourquoi ?). Pour montrer ce théorème nous introduirons la notion d'automate non-déterministe.

5.2 Automates finis non-déterministes

Avant de donner la définition d'un automate non-déterministe, donnons un exemple de digraphe d'un tel automate.

Exemple 5.2.1.



Nous constatons les nouveautés suivantes par rapport aux automates déterministes :
— Les transitions δ_a, pour $a \in \mathbb{A}$ ne sont plus de $\mathbb{X}$ dans $\mathbb{X}$ mais de $\mathbb{X}$ dans $\mathcal{P}(\mathbb{X})$.
— Le $\mathrm{dom}(\delta_a)$ peut-être un sous-ensemble strict de $\mathbb{X}$.
— Un nouveau symbole a $\varepsilon \notin \mathbb{A}$ sert à indexer la famille des transitions.
Ainsi, pour le digraphe précédent, les transitions sont définies par

x	$\delta_0(x)$	$\delta_1(x)$	$\delta_\varepsilon(x)$
x_0	$\{x_0\}$	$\{x_0, x_1\}$	$\emptyset$
x_1	$\{x_2\}$	$\emptyset$	$\{x_2\}$
x_2	$\emptyset$	$\{x_3\}$	$\emptyset$
x_3	$\{x_3\}$	$\{x_3\}$	$\emptyset$

ou de manière équivalente par les relations représentées par les matrices

$$D_0 := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} ; \quad D_1 := \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} ; \quad D_\varepsilon := \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Nous constatons immédiatement que $\mathsf{dom}(D_0) = \{x_0, x_1, x_3\}$, $\mathsf{dom}(D_1) = \{x_0, x_2, x_3\}$ et $\mathsf{dom}(D_\varepsilon) = \{x_1\}$. Par ailleurs, la matrice D_1 ne correspond pas à une application mais uniquement à une relation. Dans cet exemple, nous avons aussi que $D_\varepsilon \odot D_\varepsilon = \mathbf{0}$ (D_ε est 2-$\odot$-nilpotente).

Oublions pour l'instant la présence de D_ε. Le fait que D_1 n'est plus une application signifie que partant de x_0 nous aurons plusieurs trajectoires possibles émanant du point x_0. Le fait que $\mathsf{dom}(D_a)$ est strictement contenu dans $\mathbb{X}$ signifie que certaines trajectoires ne pourront pas continuer indéfiniment mais s'arrêteront après un nombre fini d'étapes. Venons-en maintenant au cas de D_ε. Étant donné que ε est l'élément neutre du monoïde libre $\mathbb{A}^*$, il s'ensuit que lorsque nous considérons un mot arbitraire $\alpha = a_1 \cdots a_n$, nous pouvons intercaler enre chaque lettre du mot α un nombre arbitraire de mots vides ε ; en effet $\alpha = a_1\varepsilon a_2 \cdots a_n = a_1\varepsilon\varepsilon a_2 \cdots a_n = \ldots = a_1\varepsilon^{m_1} \ldots a_2\varepsilon^{m_2} \cdots a_n\varepsilon^{m_n}$, où $(m_k)_{k=1,\ldots,n}$ sont des entiers positifs arbitraires. Cependant, dans cet exemple, la matrice D_ε est $\odot$-nilpotente donc les entiers m_k dans l'expression ci-dessus ne peuvent prendre que les valeurs 0 ou 1. En outre, $\mathsf{dom}(D_\varepsilon) = \{x_1\}$, donc $m_k \neq 0$ si et seulement si le dernier état visité est l'état x_1. Cela signifie que si lors de la lecture du mot α l'automate visite l'état x_1, il peut *instantanement* sauter à l'état x_2 car il y a une arête étiquetée ε reliant x_1 à x_2.

Exercice 5.2.2. Pour l'automate ci-dessus, dessiner l'arbre des toutes les trajectoires calculatoires possibles lorsque le mot en entrée est $\alpha = 010110$.

Exercice 5.2.3. Soit $(D_a)_{a\in\mathbb{A}\cup\{\varepsilon\}}$ une famille de matrices représentant des relations sur $\mathbb{X}$. Dans la suite on utilise le même symbole pour noter les relations et les matrices qui les représentent. On note E la relation D_ε^*.

1. Montrer que $E = \vee_{n\in\mathbb{N}} D_\varepsilon^{\odot n}$, où $D_\varepsilon^{\odot 0} = \mathbb{I}_\mathbb{X}$.
2. Calculer E pour l'automate de l'exemple 5.2.1.
3. Montrer qu'en général, si $\alpha = a_1 \cdots a_n \in \mathbb{A}^*$ est un mot en entrée d'un automate décrit par une famille des relations $(D_a)_{a\in\mathbb{A}\cup\{\varepsilon\}}$, il y aura une trajectoire calculatoire de x à y si et seulement si l'élément $D[\alpha](x,y)$ de la matrice $D[\alpha] = E \odot D_{a_1} \odot E \odot \cdots \odot D_{a_n} \odot E$ est égal à 1.

Définition 5.2.4. Un **automate fini non-déterministe** est la donnée $M = (\mathbb{X}, \mathbb{A}, \delta, x_0, \mathbb{F})$, où $\mathbb{X}$ est un ensemble fini d'états, $\mathbb{F} \subseteq \mathbb{X}$ est l'ensemble des états finaux d'acceptation, $\mathbb{A}$ est l'alphabet fini des messages d'entrée, $x_0 \in \mathbb{X}$ est l'état initial, $\delta = (\delta_a)_{a\in\mathbb{A}\cup\{\varepsilon\}}$, où $\delta_a : \mathbb{X} \to \mathcal{P}(\mathbb{X})$ et $\mathsf{dom}(\delta_a) \subseteq \mathbb{X}$, est la famille des fonctions de transition. L'ensemble d'automates finis déterministes est noté AFN. Le digraphe de l'automate $\mathbb{G}(M)$ sera le digraphe avec des arêtes étiquetées provenant des relations $(D_a)_{a\in\mathbb{A}\cup\{\varepsilon\}}$ définies par la famille δ. Les arêtes étiquetées ε représentent des transitions instantanées vers l'état terminal de l'arête aussitôt que l'état source est visité. Le **langage** reconnu par M est l'ensemble des mots qui engendrent des trajectoires calculatoires de $\mathbb{G}^*(M)$ qui rendent accessibles depuis x_0 les états de $\mathbb{F}$, i.e.

$$\mathcal{L}(M) = \{\boldsymbol{\alpha} \in \mathbb{A}^* : \sum_{y\in\mathbb{F}} D[\boldsymbol{\alpha}](x_0, y) \geq 1\}.$$

5.3 Équivalence entre automates déterministes et non-déterministes

Tout automate déterministe est un automate non-déterministe mais la définition d'un automate fini non-déterministe semble plus générale que celle d'un automate fini déterministe. On s'attend donc à ce que $\mathcal{L}(\text{AFD}) \subset \mathcal{L}(\text{AFN})$. Il apparaît comme une surprise qu'en fait $\mathcal{L}(\text{AFD}) = \mathcal{L}(\text{AFN})$ comme le montre le théorème suivant.

Théorème 5.3.1. *Pour tout $M \in$ AFN il existe un $M' \in$ AFD qui est équivalent à M, i.e. tel que $\mathcal{L}(M') = \mathcal{L}(M)$.*

Démonstration. Soit $M = (\mathbb{X}, \mathbb{A}, \delta, x_0, \mathbb{F}) \in$ AFN un automate non-déterministe qui reconnaît le langage L. Nous construisons un $M' = (\mathbb{Y}, \mathbb{A}, d, Y_0, \mathbb{H}) \in$ AFD tel que $\mathcal{L}(M') = L$, où $\mathbb{Y} = \mathcal{P}(\mathbb{X})$. Nous distinguons deux cas.

[M n'a pas d'arêtes ε] : Dans ce cas, nous définissons $Y_0 = \{x_0\}$ et la famille d des fonctions de transition — pour $a \in \mathbb{A}$ et $Y \in \mathbb{Y}$ — par

$$d_a(Y) = \{x \in \mathbb{X} : x \in \delta_a(y), \text{ pour un } y \in Y\} = \cup_{y\in Y}\delta_a(y).$$

L'ensemble des états finaux d'acceptation $\mathbb{H} = \{Y \in \mathbb{Y} : Y \cap \mathbb{F} \neq \emptyset\}$.

[M a des arêtes ε] : Nous introduisons la notation, pour tout $Y \in \mathbb{Y}$,

$$\mathcal{E}(Y) = \cup_{y\in Y} \cup_{\xi\in\mathbb{G}^*_y(D_\varepsilon):s(\xi)=y} \{t(\xi)\}.$$

$\mathcal{E}(Y)$ sont les états accessibles depuis les points de Y en suivant uniquement des arêtes ε. Il suffit alors de définir $Y_0 = \mathcal{E}(\{x_0\})$, $d_a(Y) = \cup_{y\in Y}\mathcal{E}(\delta_a(y))$ et les états d'acceptation sont de nouveau $\mathbb{H} = \{Y \in \mathbb{Y} : Y \cap \mathbb{F} \neq \emptyset\}$.

À chaque instant l'automate déterministe M' entre dans un état (sous-ensemble de $\mathbb{X}$) égal au sous-ensemble des états qui seraient accessibles par l'automate non-déterministe M au même instant. □

Corollaire 5.3.2. *Un langage L est régulier si, et seulement si, il existe un automate non-déterministe $M \in \text{AFN}$ tel que $\mathcal{L}(M) = L$.*

Démonstration. Si L est régulier, il existe un automate déterministe $M' \in \text{AFD}$ qui le reconnaît. Mais $\text{AFD} \subseteq \text{AFN}$, donc L est reconnu par l'automate non-déterministe M'. Réciproquement, si L est reconnu par un automate $M' \in \text{AFN}$, par le théorème précédent, il existe un automate déterministe $M \in \text{AFD}$ qui est équivalent à M', qui reconnaît par conséquent L. Le langage est alors régulier. □

Nous illustrons la puissance de la notion d'automate non-déterministe en donnant une nouvelle démonstration du théorème 5.1.6 de clôture des langages réguliers par réunions finies.
Nouvelle démonstration du théorème 5.1.6 : Soient L_1 et L_2 deux langages réguliers et $N_1 = (\mathbb{X}^1, \mathbb{A}, \delta^1, x_0^1, \mathbb{F}^1) \in \text{AFN}$ et $N_2 = (\mathbb{X}^2, \mathbb{A}, \delta^2, x_0^2, \mathbb{F}^2) \in \text{AFN}$ deux automates non-déterministes qui les reconnaissent. On construit un automate $N = (\mathbb{X}, \mathbb{A}, \delta, x_0, \mathbb{F}) \in \text{AFN}$ avec

— $\mathbb{X} = \{x_0\} \sqcup \mathbb{X}^1 \sqcup \mathbb{X}^2$ obtenu par réunion *disjointe* des espaces des états et adjonction d'un état supplémentaire qui jouera le rôle d'état initial pour N,
— $\mathbb{F} = \mathbb{F}^1 \cup \mathbb{F}^2$,
—

$$\delta(x,a) = \begin{cases} \delta^1(x,a) & \text{si} \quad x \in \mathbb{X}^1 \\ \delta^2(x,a) & \text{si} \quad x \in \mathbb{X}^2 \\ \{x_0^1, x_0^2\} & \text{si} \quad x = x_0, a = \varepsilon \\ \emptyset & \text{si} \quad x = x_0, a \neq \varepsilon. \end{cases}$$

Il est alors évident que l'automate accepte tous les mots qui seraient acceptés individuellement par N_1 ou N_2. □

Exercice 5.3.3. Donner une interprétation graphique schématique de la démonstration précédente.

Théorème 5.3.4. *La classe des langages réguliers est fermée par*

1. *concaténation,*
2. *clôture monoïdale.*

Démonstration. 1. Soient $M_i = (\mathbb{X}^1, \mathbb{A}, \delta^i, x_0^i, \mathbb{F}^1), i = 1,2$ deux automates finis non-déterministes acceptant respectivement les langages L_i. On construit un automate $M = (\mathbb{X}^1 \sqcup \mathbb{X}^2, \mathbb{A}, \delta, x_0^1, \mathbb{F}^2)$ ayant comme espace des états $\mathbb{X}$ la réunion *disjointe* des $\mathbb{X}^1$ et $\mathbb{X}^2$ et comme fonction de transition

$$\delta(x,a) = \begin{cases} \delta^1(x,a) & \text{si} \quad x \in \mathbb{X}^1 \setminus \mathbb{F}^1 \\ \delta^1(x,a) & \text{si} \quad x \in \mathbb{F}^1, a \neq \varepsilon \\ \delta^1(x,a) \sqcup \{x_0^2\} & \text{si} \quad x \in \mathbb{F}^1, a = \varepsilon \\ \delta^2(x,a) & \text{si} \quad x \in \mathbb{X}^2. \end{cases}$$

Alors, il est évident que l'automate M accepte le langage $L_1 L_2$.

2. Soit $M = (\mathbb{X}, \mathbb{A}, \delta, x_0, \mathbb{F})$ un automate fini non-déterministe acceptant le langage L. On construit l'automate $N = (\mathbb{X} \sqcup \{\hat{x}\}, \mathbb{A}, \delta, \hat{x}, \mathbb{F} \sqcup \{\hat{x}\})$ ayant comme fonction de transition

$$\delta(x,a) = \begin{cases} \delta^1(x,a) & \text{si} \quad x \in \mathbb{X} \setminus \mathbb{F} \\ \delta^1(x,a) & \text{si} \quad x \in \mathbb{F}, a \neq \varepsilon \\ \delta^1(x,a) \sqcup \{\hat{x}\} & \text{si} \quad x \in \mathbb{F}, a = \varepsilon \\ \emptyset & \text{si} \quad x = \hat{x}, a \neq \varepsilon. \end{cases}$$

L'automate N admet alors le langage L^*.

□

Exercice 5.3.5. Donner une « démonstration » graphique du théorème précédent en représentant les automates comme de boîtes noires ayant comme entrée leur état initial et comme sortie leurs ensembles d'états finaux.

5.4 Expressions régulières

Supposons que $\mathbb{A} = \{0,1\}$. Des expressions de la forme $(0 \cup 1)0^*$ ou $(0 \cup 1)^*$ sont connues sous le nom d'expressions régulières et représentent des sous-ensembles spécifiques de $\mathbb{A}^*$ (i.e. des langages). La première désigne le langage des mots de longueur supérieure à 1 qui commencent par 0 ou par 1 et se terminent par un nombre arbitraire de zéros ; la seconde désigne l'ensmble $\mathbb{A}^*$ lui-même. Plus généralement

Définition 5.4.1. On dit que R est une **expression régulière** sur l'alphabet $\mathbb{A}$ si elle admet une des formes suivantes :
— a pour un $a \in \mathbb{A}$,
— ε,
— $\emptyset$,
— $R_1 \cup R_2$, où R_1, R_2 sont d'expressions régulières,
— $R_1 \circ R_2 = R_1 R_2$, où R_1, R_2 sont d'expressions régulières,
— R_1^*, où R_1 est une expression régulière.

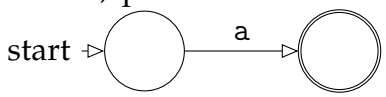
Dans la définition précédente, si $R_1 = a, R_2 = \varepsilon$, les langage décrit par R_1 sera $\{a\}$ et par R_2 sera $\{\varepsilon\}$; ces langages sont composés des uniques mots a ou ε. La relation $R = \emptyset$ décrira le langage vide. On peut craindre que la définition précédente soit circulaire. Cependant, lorsque nous disons que R s'écrit comme $R_1 \circ R_2$, par exemple, cela signifie que R_1 et R_2 ont une longueur de définition plus courte que R ; par conséquent, il s'agit d'une définition récursive, pas circulaire. Signalons enfin qu'en l'absence de parenthèses, la préséance des opérations est *, $\circ$, $\cup$. On distingue enfin entre l'expression régulière R et le langage $\mathcal{L}(R)$ décrit par l'expression régulière.

Théorème 5.4.2. *Un langage est régulier si, et seulement si, il est décrit par une expression régulière.*

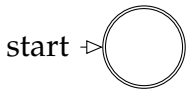
On montre ce théorème en deux étapes, en établissant les deux lemmes suivants.

Lemme 5.4.3. *Si un langage est décrit par une expression régulière R, alors il est régulier.*

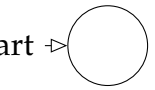
Démonstration. Nous convertissons R en automate fini non-déterministe en considérant les 6 cas énumérés ci-dessus pour la forme d'une expression régulière.

— $R = a$, pour un $a \in \mathbb{A}$. Alors $\mathcal{L}(R) = \{a\}$ et l'automate $M =$ 

accepte ce langage.

— $R = \varepsilon$. Alors $\mathcal{L}(R) = \{\varepsilon\}$ et l'automate $M =$ 

accepte ce langage.

— $R = \emptyset$. Alors $\mathcal{L}(R) = \emptyset$ et l'automate $M =$ 

« accepte » ce langage.

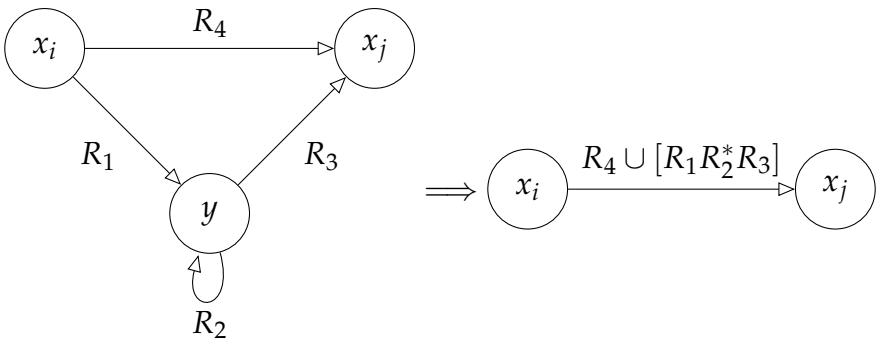
— Les cas $R = R_1 \cup R_2$, $R = R_1 R_2$ ou $R = R_1^*$ sont alors construits par récurrence à partir des 3 cas précédents car les R_1, R_2 apparaissant dans ces expressions sont nécessairement des expressions régulières plus simples (plus courtes) que R.

□

Lemme 5.4.4. *Si L est un langage régulier, alors il existe une expression régulière R qui le décrit, i.e. $\mathcal{L}(R) = L$.*

La démonstration de ce lemme — qui sera donnée un peu plus loin — repose sur une suite de transformations qui modifient localement un automate fini déterministe en automate fini non-déterministe généralisé. Ce dernier est un automate non-déterministe dont les arêtes peuvent être étiquetées par des expressions régulières au lieu de lettres de $\mathbb{A} \cup \{\varepsilon\}$; l'automate lit alors des blocs de symboles en entrée (comme le ferait un automate non-déterministe ordinaire) et accepte le bloc si son état interne devient un état d'acceptation.

Nous décrivons une transformation qui est répétée récursivement sur les arêtes de l'automate :



Cette opération d'excision d'un état et de réparation locale des arêtes, diminue le cardinal de $\mathbb{X}$ d'une unité. Étant donné que l'automate est fini, cette procédure s'arrêtera au bout d'un nombre fini d'opérations. Afin de

complètement décrire la transformation d'un automate finie déterministe en automate fini non-déterministe généralisé (AFNG), nous adjoignons deux états supplémentaires $\{x_d, x_f\}$ à l'ensemble $\mathbb{X}$ pour obtenir l'ensemble des états $\mathbb{Y} = \mathbb{X} \sqcup \{x_d, x_f\}$. Ces états supplémentaires ont les particularités suivantes :

— l'état x_d a uniquement des arêtes sortantes les reliant à tous les autres états de $\mathbb{Y} \setminus \{x_d\}$ mais pas d'arête entrante (sauf le symbole `start`) ;
— l'état x_f a uniquement des arêtes entrantes émanant de tous les autres états de $\mathbb{Y} \setminus \{x_f\}$ mais pas d'arête sortante ;
— de tout état $x \in \mathbb{Y} \setminus \{x_f\}$ émane une unique arête vers chaque état de $\mathbb{Y} \setminus \{x_d\}$;
— les arêtes sont étiquetées par des expressions régulières,
— lorsque $\mathbb{Y}$ est réduit à l'ensemble $\{x_d, x_f\}$, la procédure s'arrête ; l'étiquette de l'unique arête qui relie x_d à x_f est l'expression régulière définie par l'automate.

Définition 5.4.5. Soit $\mathcal{R} := \mathcal{R}_{\mathbb{A}}$, la classe des expressions régulières sur $\mathbb{A}$. Un **automate fini non-déterministe généralisé** M est l'automate fini non-déterministe défini par le quintuplet $(\mathbb{Y}, \mathcal{R}, \delta, x_d, \{x_f\})$, où, pour $y, y' \in \mathbb{Y}$ et $R \in \mathcal{R}$, $\delta(y, R) = y'$ si, et seulement si, $\mathsf{etiq}(y, y') = R$, où

$$\mathsf{etiq} : (\mathbb{Y} \setminus \{x_f\}) \times (\mathbb{Y} \setminus \{x_d\}) \to \mathcal{R}$$

est la fonction d'étiquetage des arêtes.

Notation 5.4.6. Étant donné que la connaissance de l'une des fonctions δ et etiq détermine l'autre, nous utiliserons indifféremment les quintiplets $(\mathbb{Y}, \mathcal{R}, \delta, x_d, \{x_f\})$ ou $(\mathbb{Y}, \mathcal{R}, \mathsf{etiq}, x_d, \{x_f\})$ pour désigner un automate M fini non-déterministe généralisé. On note AFNG la classe de ces automates.

Nous sommes maintenant en mesure de montrer le lemme 5.4.4, à l'aide de l'algorithme récursif 5.4.7 ci-dessous.

Algorithme 5.4.7. afn2exreg

Require: $M = (\mathbb{Y}, \mathcal{R}, \mathsf{etiq}, x_d, \{x_f\}) \in$ AFNG.
Ensure: Expression régulière $R \in \mathcal{R}$.

```
k ← card𝕐
if k=2 then
  R ← etiq(x_d, x_f)
  retourner R
else
  choisir y ∈ 𝕐 \ {x_d, x_f}
  𝕐 ← 𝕐 \ {y}
  repeat
    choisir (x, z) ∈ (𝕐 \ {x_f, y}) × (𝕐 \ {x_d, y})
    R_1 ← etiq(x, y)
    R_2 ← etiq(y, y)
    R_3 ← etiq(y, z)
    R_4 ← etiq(x, z)
    etiq(x, z) ← R_1 R_2^* R_3 ∪ R_4
  until que les couples (x, z) aient balayé la totalité de (𝕐 \ {x_f, y}) ×
  (𝕐 \ {x_d, y})
  M ← (𝕐, ℛ, etiq, x_d, {x_f})
  afn2exreg(M)
end if
```

Démonstration du lemme 5.4.4 : Si L est un langage régulier, alors il est accepté par un automate $M \in$ AFD $\subseteq$ AFNG. En appliquant l'algorithme 5.4.7, nous obtenons une expression régulière R qui décrit le langage $L = \mathcal{L}(R)$. □

5.5 Automates finis probabilistes et langages stochastiques

Définition 5.5.1. Un **automate fini probabiliste** (AFP) M est le quintuplet

$$M = (\mathbb{X}, \mathbb{A}, (P_a)_{a\in\mathbb{A}}, \boldsymbol{\pi}, \mathbb{F}),$$

où $\mathbb{X}$ est l'ensemble fini des états de l'automate, $\mathbb{A}$ l'alphabet fini de l'entrée, $(P_a)_{a\in\mathbb{A}}$ une collection (indexée par l'ensemble $\mathbb{A}$) de matrices stochastiques $|\mathbb{X}| \times |\mathbb{X}|$, $\boldsymbol{\pi} = (\pi_0(x))_{x\in\mathbb{X}}$ un vecteur de probabilité (ligne) sur $\mathbb{X}$ et $\mathbb{F} \subseteq \mathbb{X}$, l'ensemble des états finaux. On étend la famille $(P_a)_{a\in\mathbb{A}}$ en une famille $(P_\alpha)_{\alpha\in\mathbb{A}^*}$ en définissant, pour $\alpha = a_1 \cdots a_{|\alpha|}$,

$$P_\alpha = P_{a_1} \cdots P_{a_{|\alpha|}} = \prod_{i=1}^{|\alpha|} P_{a_i}.$$

Cette expression induit une chaîne de Markov $(X_n)_{n=0,\ldots,|\alpha|}$ sur $\mathbb{X}$ (et une

mesure de probabilité[3], notée $\mathbb{P}_{\boldsymbol{\pi},\alpha}$, sur les trajectoires calculatoires de l'automate) par la formule

$$\mathbb{P}_{\boldsymbol{\pi},\alpha}(X_0 = x_0, \ldots, X_k = x_k) = \pi(x_0) P_{a_1}(x_0, x_1) \cdots P_{a_k}(x_{k-1}, x_k),$$

pour tout $\alpha \in \mathbb{A}^*$ et $0 \leq k \leq |\alpha|$. On note $\mathbb{1}_{\mathbb{F}}$ le vecteur (colonne) qui est l'indicatrice de $\mathbb{F}$. Pour un $r \in [0,1[$, le **langage accepté à seuil** r par l'automate est l'ensemble de mots qui mettent l'automate dans un état final avec probabilité supérieure à r, i.e.

$$\mathcal{L}_r(M) = \{\alpha \in \mathbb{A}^* : \boldsymbol{\pi} P_\alpha \mathbb{1}_{\mathbb{F}} > r\} = \{\alpha \in \mathbb{A}^* : \mathbb{P}_{\boldsymbol{\pi},\alpha}(X_{|\alpha|} \in \mathbb{F}) > r\}.$$

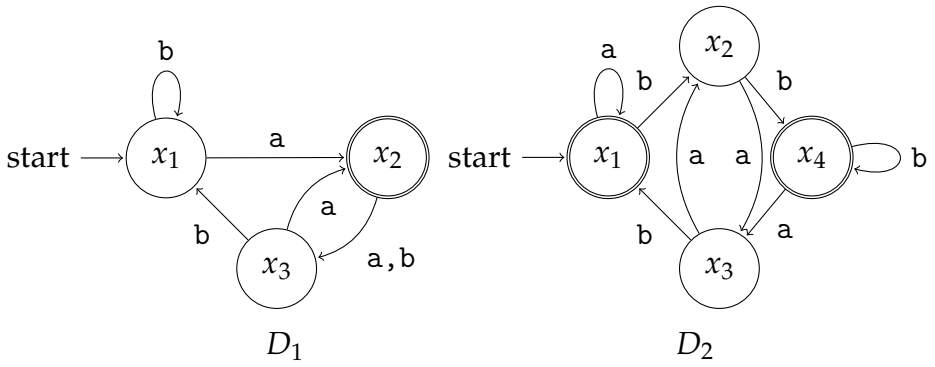
Définition 5.5.2. Un langage $L \subseteq \mathbb{A}^*$ est **r-stochastique**, pour un $r \in [0,1[$, s'il existe un $M \in \text{AFP}$ tel que $L = \mathcal{L}_r(M)$; il est **stochastique** s'il existe un $r \in [0,1[$ pour lequel il est r-stochastique.

Contrairement aux automates finis déterministes ou non-déterministes qui sont équivalents à des langages réguliers, les langages stochastiques ne sont pas nécessairement réguliers. Par exemple, on peut montrer que le langage r-stochastique introduit à l'exercice **??** n'est régulier que si, et seulement si, $r \in \mathbb{Q} \cap [0,1[$.

5.6 Exercices

Automates finis

30. On note D_1 et D_2 les automates finis déterministes suivants :



(a) Quelle est la suite des états à travers lesquels passent les automates lorsqu'on leur présente l'entrée `aabb` ?
(b) Les automates acceptent-ils l'entrée `aabb` ?
(c) Les automates acceptent-ils l'entrée ε ?
(d) Donner les descriptions formelles de ces automates.

31. Un automate fini est donné par le quintuplet $(\{a,b,c,d,e\}, \{0,1\}, \delta, c, \{c\})$, où δ est définie par le tableau :

3. Cf. cours de *Probabilités et statistique pour la théorie de l'information*.

x	$\delta_0(x)$	$\delta_1(x)$
a	a	b
b	a	c
c	b	d
d	c	e
e	d	e

Dessiner le diagramme de l'automate.

32. Chacun des langages ci-dessous peut-être défini comme l'intersection de deux langages plus simples. Construire les automates finis déterministes qui décrivent ces langages plus simples et les combiner pour construire l'automate qui reconnaît les langages L_1 et L_2. Dans tous les cas $\mathbb{A} = \{\mathtt{a},\mathtt{b}\}$.

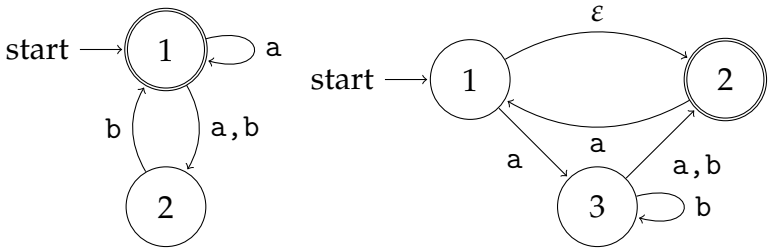
    (a) $L_1 = \{\alpha : \alpha$ contient exactement deux a et au moins deux b$\}$.
    (b) $L_2 = \{\alpha : \alpha$ contient un nombre pair de a et chaque a est suivi par au moins un b$\}$.

33. Donner les diagrammes des automates finis non-déterministes avec le nombre prescrit d'états qui reconnaissent les langages suivants. Dans chaque cas, $\mathbb{A} = \{\mathtt{0},\mathtt{1}\}$.

    (a) $\{\alpha : \alpha$ se termine par 00$\}$ (avec trois états).
    (b) $\mathtt{1}^*(\mathtt{001}^+)^*$. (Commencer par construire un automate sans contrainte sur le nombre des états pour simplifier ensuite en un automate à 4 états. Est-il possible de le simplifier encore pour ne laisser que 3 états ?)

34. Montrer que tout automate fini non-déterministe avec plusieurs états d'acceptation peut être converti en un automate fini non-déterministe équivalent avec un seul état d'acceptation.

35. Utiliser la procédure canonique introduite en cours pour transformer les automates non-déterministes N_1 et N_2, dont les diagrammes sont donnés ci-dessous, en automates déterministes équivalents.



N_1 N_2

36. *Un automate déterministe et un non-déterministe qui reconnaissent le même langage.* [Extrait du contrôle du 21 avril 2021].

    Soit $L = L_1 \cup L_2$ le langage régulier sur l'alphabet $\mathbb{A} = \{a, b\}$ exprimable comme réunion de deux langages réguliers plus simples :

    $L_1 = \{\alpha \in \mathbb{A}^* : \alpha$ contient un nombre pair de $a\}$ et $L_2 = \{\alpha \in \mathbb{A}^* : \alpha$ commence par un

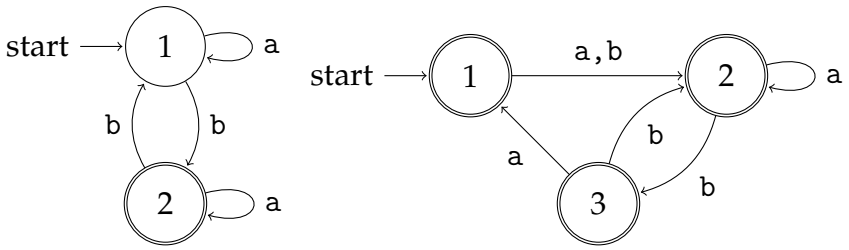
(a) Déterminer les automates finis déterministes $M_1 = (\mathbb{X}, \mathbb{A}, x_0, \gamma, \mathbb{F})$ et $M_2 = (\mathbb{Y}, \mathbb{A}, y_0, \delta, \mathbb{G})$ qui reconnaissent respectivement L_1 et L_2. (Il suffit de dessiner leurs graphes).

(b) Utiliser la méthode standard vue en cours (de produit cartésien des automates M_1 et M_2) pour donner la description de l'automate déterministe $M = (\mathbb{W}, \mathbb{A}, w_0, d, \mathbb{H})$ qui reconnaît $L = L_1 \cup L_2$, i.e. donner la définition de $\mathbb{W}$, le tableau de valeurs de la fonction d, la définition de w_0 et de $\mathbb{H}$. Compléter votre réponse en donnant le graphe de M.

(c) Dessiner le graphe (simplifié) de l'automate fini non-déterministe N — obtenu par la méthode canonique vue en cours — qui reconnaît $L = L_1 \cup L_2$.

Expressions régulières

37. Utiliser la procédure canonique introduite en cours pour transformer les automates non-déterministes N_1 et N_2, dont les diagrammes sont donnés ci-dessous, en expressions régulières.



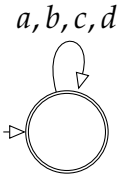
N_1 N_2

38. Soit L un langage sur l'alphabet $\mathbb{A}$. Montrer que $L = L^+ \Leftrightarrow LL \subseteq L$.

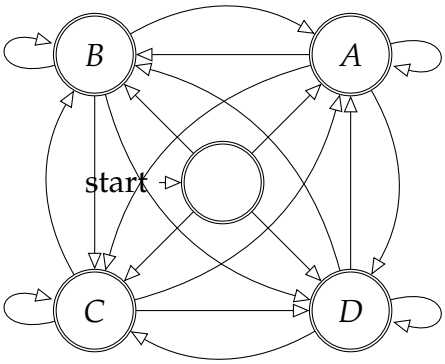
39. Convertir les expressions régulières suivantes en automates finis non-déterministes selon la procédure canonique du cours.

(a) $\mathtt{a(abb)}^* \cup \mathtt{b}$.
(b) $\mathtt{a}^+ \cup \mathtt{ab}^+$.
(c) $(\mathtt{a} \cup \mathtt{b}^+)\mathtt{a}^+\mathtt{b}^+$.

40. Soit $\mathbb{A} = \{a, b, c, d\}$.

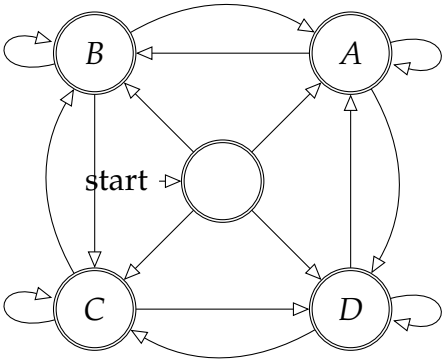
(a) Soient $M_1 =$ 

et

$M_2 =$ 

(Pour des raisons de clarté, nous avons omis les étiquettes des arêtes de M_2. Chaque arête pointant sur un nœud indexé par une lettre majuscule sera étiquetée par la même lettre minuscule. Ainsi, au nœud A arrivent 5 arêtes étiquetées a, etc.) Déterminer $\mathcal{L}(M_1)$ et $\mathcal{L}(M_2)$.

(b) En utilisant la même convention pour les étiquettes, on note

$M_3 =$ 

Déterminer $\mathcal{L}(M_3)$.

(c) On suppose maintenant que $\mathbb{A}$ au lieu d'être un simple ensemble est un sous-ensemble d'un groupe, noté $\mathbb{F}_2$. On suppose que $a = c^{-1}$ et $b = d^{-1}$ et qu'aucune autre relation ne relie les éléments de $\mathbb{A}$. Le groupe $\mathbb{F}_2$, appelé groupe libre à deux générateurs, est dénombrable et ses éléments peuvent être obtenus comme produits arbitraires d'éléments de $\mathbb{A}$. Monter qu'il existe une bijection [4] entre $\mathcal{L}(M_3)$ et $\mathbb{F}_2$? Quelle relation a $\mathcal{L}(M_2)$ avec $\mathbb{F}_2$?

(d) Maintenant, les éléments de $\mathbb{A}$, en sus des relations $a = c^{-1}$ et $b = d^{-1}$, vérifient aussi la relation $ab = ba$. Quel est le groupe engendré par $\mathcal{L}(M_1)$ ou $\mathcal{L}(M_2)$?

4. Un groupe engendré à partir d'un ensemble générateur fini est *hyperbolique* si (en tant qu'ensemble) il est en bijection avec un langage régulier. Par conséquent $\mathbb{F}_2$ est hyperbolique ; il constitue même l'archétype d'un groupe hyperbolique.

41. *Langages sur des mots infinis.* [Extrait du devoir maison du 22 mars 2020].

    Définition : Soient $\mathbb{A}$ un alphabet fini et $M = (\mathbb{X}, \mathbb{A}, x_0, \delta, \mathbb{F}) \in \text{AFD}$ un automate fini déterministe. On considère l'ensemble de mots infinis sur $\mathbb{A}$:

    $$\mathbb{A}^{\mathbb{N}_>} := \{\boldsymbol{\alpha} = \alpha_1\alpha_2\alpha_3\ldots, \alpha_i \in \mathbb{A}\}$$

    et, pour tout $\boldsymbol{\alpha} \in \mathbb{A}^{\mathbb{N}_>}$, on note $\mathbf{x}_M(\boldsymbol{\alpha}) = x_0x_1x_2x_3\ldots \in \mathbb{X}^{\mathbb{N}}$ la suite infinie d'états internes de l'automate dont les termes sont définis par la récurrence

    $$x_{k+1} = \delta(x_k, a_{k+1}), k \in \mathbb{N}.$$

    (a) On note

    $$\mathcal{L}_\infty^\exists(M) = \{\boldsymbol{\alpha} \in \mathbb{A}^{\mathbb{N}_>} = \mathbf{x}_M(\boldsymbol{\alpha}) \text{ contient au moins un état de } \mathbb{F}\}$$

    et on dit qu'un langage de mots infinis $\Lambda \subset \mathbb{A}^{\mathbb{N}_>}$ est **existentiellement régulier**, s'il existe un $M \in \text{AFD}$ tel que $\Lambda = \mathcal{L}_\infty^\exists(M)$.

    (b) On note

    $$\mathcal{L}_\infty^\omega(M) = \{\boldsymbol{\alpha} \in \mathbb{A}^{\mathbb{N}_>} = \mathbf{x}_M(\boldsymbol{\alpha}) \text{ contient une infinité d'états de } \mathbb{F}\}$$

    et on dit qu'un langage de mots infinis $\Lambda \subset \mathbb{A}^{\mathbb{N}_>}$ est **ω-régulier**, s'il existe un $M \in \text{AFD}$ tel que $\Lambda = \mathcal{L}_\infty^\omega(M)$.

    Montrer que le langage des mots infinis sur $\mathbb{A} = \{a, b\}$ contenant au moins deux b est à la fois existentiellement et ω-régulier.

42. *Les codes comme langages.* [Extrait du contrôle du 6 mars 2019]. Soit un alphabet fini $\mathbb{A} = \{0, 1\}$. On rappelle qu'un **langage** (sur $\mathbb{A}$) est une partie distinguée $L \subseteq \mathbb{A}^*$. Si L et M sont deux langages sur $\mathbb{A}^*$, leur produit (par concaténation) est $LM := \{\alpha\beta, \alpha \in L, \beta \in M\}$ et la clôture monoïdale du langage L est $L^* := \cup_{n\in\mathbb{N}} L^n$, où L^n est la produit par concaténation n fois.

    Un **code** (sur $\mathbb{A}$) est un langage $K \in \mathbb{A}^*$ tel que tout mot de K^* se décompose **de manière unique** en produit par concaténation de mots de K.

    Dans la suite, on considère une fonction $\pi : \mathbb{A} \to \mathbb{R}_>$ telle que $\sum_{a\in\mathbb{A}} \pi(a) = 1$. On étend π sur $\mathbb{A}^*$ en définissant, pour un mot $\alpha = a_0 \cdots a_{m-1} \in \mathbb{A}^m$, $\pi(\alpha) = \prod_{0\le i<m} \pi(a_i)$ et on étend π sur tout langage L par $\pi(L) = \sum_{\alpha\in L} \pi(\alpha)$.

    (a) Pour deux langages L et M sur $\mathbb{A}$, montrer que

    $$\pi(LM) \le \pi(L)\pi(M).$$

    (b) Montrer que $\pi(L^*) = \infty \Rightarrow \pi(L) \ge 1$.

    (c) Si K est un code sur $\mathbb{A}$, montrer que pour tout entier $n \ge 1$, on a $\pi(K^n) = \pi(K)^n$.

(d) Réciproquement, montrer que si $K \subseteq \mathbb{A}^*$ est un langage qui vérifie, pour tout entier $n \geq 1$, l'égalite $\pi(K^n) = \pi(K)^n$, alors K est un code. *Suggestion :* Faire une démonstration par l'absurde.

(e) Soit K un code. Considérer la famille $(K_m)_{m\in\mathbb{N}_>}$ avec $K_m = \{\alpha \in K : |\alpha| \leq m\}$ qui est croissante et exhaustive de K, i.e. $K_m \uparrow K$. Montrer que pout tout $n \geq 1$, on a $\pi(K_m^n) \leq mn$. *Suggestion :* remarquer que $\mathbb{A}$ est sûrement un code !

(f) En conclure que tout code K vérifie $\pi(K) \leq 1$. *Suggestion :* remarquer que K_m, comme sous-langage d'un code, est sûrement un code !

6

Langages non-réguliers et machines de Turing

Le chapitre précédent a mis en évidence l'équivalence entre langages réguliers, automates finis et expressions régulières. Si nous considérons le langage $L = \{a^n b^n, n \geq 0\} \subset \{a, b\}^*$, nous constatons que si nous cherchons un automate (fini) qui le reconnaît nous n'y arrivons pas ; toute tentative de construction d'un tel automate échoue en nous laissant l'impression que plus n augmente, plus le nombre d'états de l'automate doit augmenter. En réalité, ce langage n'est pas régulier. Il est donc important d'avoir un critère qui nous permet de reconnaître si un langage est régulier ; si ce n'est pas le cas, il faut chercher un procédé plus général qu'un automate fini.

6.1 Tests de non-régularité

6.1.1 Lemme de l'itération

Le lemme de l'itération [1] nous apprend que tout langage régulier vérifie une certaine propriété itérative. Si un langage n'a pas cette propriété, il est nécessairement non-régulier.

1. Le lemme de l'itération est appelé *pumping lemma* en anglais, ce terme est parfois traduit en français par « lemme de pompage », traduction pour le moins absurde car il n'y a rien à pomper .... Il s'agit d'une traduction littérale de *to pump* en pomper. Or, tant en français qu'en anglais, la signification première de pomper (V.T.) est : puiser, extraire un liquide à l'aide d'une pompe. La nomenclature anglaise provient de la locution idiomatique *to pump something out*, signifiant « produire ou émettre quelque chose en très grande quantité », signification totalement absente du mot « pompage ». Ex. *Computer industry pumps out polluting waste.*

Théorème 6.1.1 (Lemme de l'itération ou « *pumping lemma* »). *Si L est un langage régulier sur $\mathbb{A}$, alors il existe un entier $p \geq 1$ tel que pour tout $\zeta \in L$ avec $|\zeta| \geq p$, il existe une décomposition $\zeta = \alpha\beta\gamma$ en des mots $\alpha, \beta, \gamma \in \mathbb{A}^*$ vérifiant*
— $|\beta| > 0$,
— $|\alpha\beta| \leq p$,
— $\forall r \in \mathbb{N}, \alpha\beta^r\gamma \in L$.

Démonstration. La régularité du langage L implique l'existence d'un automate fini $M = (\mathbb{X}, \mathbb{A}, \delta, x_0, \mathbb{F}) \in \text{AFD}$, tel que $\mathcal{L}(M) = L$. Soient $p = \text{card}\mathbb{X}$ (on a nécessairement $p < \infty$ car l'automate est fini) et $\zeta = a_1 \cdots a_n \in \mathbb{A}^*$ avec $n \geq p$. L'automate commence en x_0 et ensuite suit une trajectoire calculatoire $\mathbf{x} = x_0 \cdots x_n$, où $x_i = \delta(x_{i-1}, a_i), i = 1, \ldots, n$ et $x_n \in \mathbb{F}$ (car ζ est accepté). Alors $|\mathbf{x}| = n + 1 > p$. Or il existe seulement p états distincts ; nécessairement donc il existe au moins deux indices i, j avec $0 \leq i < j \leq n$ tels que $x_i = x_j =: y$. Considérons alors les décompositions

$$\zeta = \underbrace{a_1 \cdots a_{i-1}}_{\alpha}\underbrace{a_i a_{i+1} \cdots a_{j-1}}_{\beta}\underbrace{a_j a_{j+1} \cdots a_n}_{\gamma}$$
$$\mathbf{x} = \underbrace{x_0 \cdots x_{i-1}}_{\mathbf{u}}\underbrace{y x_{i+1} \cdots x_{j-1}}_{\mathbf{v}}\underbrace{y x_{i+1} \cdots x_n}_{\mathbf{w}}.$$

En d'autres termes, si le mot ζ est présenté en entrée et est accepté par l'automate, alors la trajectoire calculatoire $\mathbf{x}$ suivie se termine dans $\mathbb{F}$, i.e. $x_n \in \mathbb{F}$. Si nous présentons le mot $\zeta' = \alpha\beta\beta\gamma$ à l'automate, il est évident que ζ' sera accepté car l'automate suivra la trajectoire calculatoire $\mathbf{x}' = \mathbf{uvvw}$ qui est aussi une trajectoire admissible par δ se terminant dans $\mathbb{F}$. Il en sera de même pour tous les mots $\alpha\beta^r\gamma$, avec $r \geq 0$. Ainsi, à partir d'un seul mot $\alpha\beta\gamma \in L$, le lemme produit *(pumps out)* toute une famille infinie $\alpha\beta^r\gamma \in L, r \in \mathbb{N}$. □

— Les mots α ou γ peuvent être vides.
— Si $|\beta| = 0$ (i.e. $\beta = \varepsilon$ ou $\zeta = \alpha\gamma$), le lemme ne nous apprend rien car la famille infinie est constante i.e. $\alpha\beta^r\gamma = \zeta \in L$, pour tout $r \in \mathbb{N}$.
— Si $|\alpha\beta| > p$, alors le mot $\alpha\beta$ contiendrait déjà une répétition. Sans perte de généralité, nous pouvons donc supposer que $|\alpha\beta| \leq p$.

Le lemme de l'itération est utile dans sa négation (cf. exercice 43). Si les 3 propriétés du lemme sont vérifiées par une décomposition des mots de L, cela ne veut pas dire que le langage est nécessairement régulier car on peut montrer qu'il existe de langages non-réguliers qui vérifient les 3 conditions (cf. exercice 46). Une condition nécessaire et suffisante de régularité est fournie par le résultat plus fort du théorème de Myhill et Nerode (cf. théorème 6.1.5 ci-dessous).

6.1.2 Théorème de Myhill-Nerode

Définition 6.1.2. Soit $L \subseteq \mathbb{A}^*$ un langage.

— Soient deux mots arbitraires $\alpha, \beta \in \mathbb{A}^*$. On dit qu'un mot $\gamma \in \mathbb{A}^*$ permet de les discerner si, et seulement si, un et un seul des mots $\alpha\gamma$ et $\beta\gamma$ est dans le langage.
— Deux mots α et β sont **indiscernables par le langage** (on note $\alpha \equiv_L \beta$) s'il n'existe pas de mot γ qui permet de les discerner. En d'autres termes, $\forall \gamma \in \mathbb{A}^*$

$$[\alpha\gamma \in L] \Leftrightarrow [\beta\gamma \in L].$$

Lemme 6.1.3. *L'indiscernabilité par un langage $L \subseteq \mathbb{A}^*$ est une relation d'équivalence sur $\mathbb{A}^*$.*

Démonstration. Cf. l'exercice 45. □

Définition 6.1.4. Soit R une relation d'équivalence sur un ensemble dénombrablement infini $\mathbb{W}$. On appelle **indice** de R le nombre de classes d'équivalence de R, i.e. $\mathsf{ind}(R) = \mathsf{card}(\mathbb{W}/R)$.

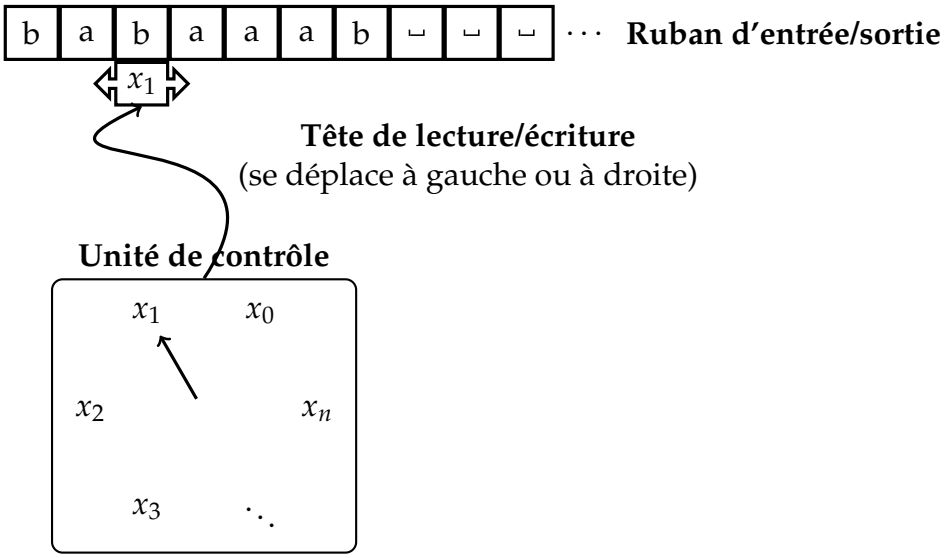
Théorème 6.1.5 (Théorème de Myhill et Nerode)**.** *Un langage L est régulier si, et seulement si,* $\mathsf{ind}(\equiv_L) < \infty$.

6.2 Machines de Turing

Dans ce cours nous discutons uniquement de machines de Turing *classiques* et d'ordinateurs *classiques* — par opposition aux machines de Turing et ordinateurs *quantiques*. Nous omettrons donc dorénavant le qualificatif *classique*.

Une machine de Turing peut faire tout ce qu'un ordinateur peut faire et nous utiliserons le terme de machine de Turing comme modèle théorique d'un algorithme. Cependant, certains problèmes peuvent ne pas admettre de solution algorithmique, i.e. restent au delà des limites théoriques de tout calcul effectif.

Nous présentons dans la suite les éléments constitutifs et le fonctionnement d'une machine de Turing. La machine a une unité de contrôle qui permet de changer ses états internes, déplacer une tête de lecture/écriture au dessus d'un ruban (semi-)infini. Initialement, les cellules du ruban contiennent les lettres composant le mot $\alpha = a_1 \dots a_n \in \mathbb{A}^*$ (entrée de la machine) ; le mot est complété en un mot infini par des $\sqcup$ et la tête se trouve au dessus de la première lettre du mot α. La machine, en lisant a_1 et son état initial x_0 écrit un symbole dans la première cellule (éventuellement le même que celui qu'elle vient de lire), se déplace sur une cellule adjacente (uniquement vers la droite s'il la position courante est la première du mot) et change son état interne selon une fonction de transition.



Si la machine entre dans un état d'acceptation, elle s'arête et accepte le mot α ; si elle entre dans un état de rejet, elle s'arrête aussi et rejette le mot α.

Définition 6.2.1. Une **machine de Turing** M est le sextuplet $(\mathbb{X}, \mathbb{A}, \mathbb{B}, \delta, x_0, \mathbb{F})$ où
- $\mathbb{X}$ est l'ensemble des états internes,
- $\mathbb{A}$ est l'alphabet de codage du langage de l'entrée,
- $\mathbb{B} \supset \mathbb{A}$ est l'alphabet (étendu) du ruban (en particulier $\text{␣} \in \mathbb{B}$ mais $\text{␣} \notin \mathbb{A}$),
- $\delta : \mathbb{X} \times \mathbb{B} \to \mathbb{X} \times \mathbb{B} \times \{g, d\}$ est la fonction de transition ($g := -1$ signifiant déplacement de la tête une cellule à gauche, $d := +1$ une cellule à droite),
- x_0 est l'état initial,
- $\mathbb{F} = \mathbb{F}_{\text{acc}} \sqcup \mathbb{F}_{\text{rej}}$ est l'ensemble des états d'arrêt, scindé en $\mathbb{F}_{\text{acc}} = \{x_{\text{acc}}\}$ et $\mathbb{F}_{\text{rej}} = \{x_{\text{rej}}\}$ (on suppose toujours que $x_0 \neq \mathbb{F}$).

La classe des machines de Turing est notée MT.

La « machine » de Turing est une construction abstraite (théorique) qui sert de prototype à tout calcul effectué sur ordinateur. À l'occasion du centenaire (2012) depuis la naissance d'Alan Tuirng, les élèves de l'ÉNS de Lyon ont construit une réalisation mécanique d'une machine de Turing.

L'espace des configurations du complexe composé de la machine, du ruban et de la tête de lecture est l'espace $\mathbb{S} = \mathbb{X} \times \mathbb{B}^{\mathbb{N}} \times \mathbb{N}_{>}$. À l'instant initial $t = 0$,
- la machine reçoit en entrée le mot $\alpha = a_1 \dots a_{|\alpha|} \in \mathbb{A}^*$ (qui est représenté sur le ruban comme le mot $B_0 = \beta^0 := \alpha \text{␣␣} \cdots \in \mathbb{B}^{\mathbb{N}}$),
- l'état interne de la machine est $X_0 = x_0$
- et la position de la tête est $P_0 = 1$ au dessus de la première cellule du ruban,

ce qui fait que la configuration initiale $S_0 := (X_0, B_0, P_0) = (x_0, \beta^0, 1)$. Supposons qu'à l'instant t la configuration instantanée est $S_t := (X_t, B_t, P_t) = (x, \beta, p)$. Écrivons $\beta = b_1 b_2 \cdots \in \mathbb{B}^{\mathbb{N}}$ et notons $c = b_p$ (la lettre de β dans la

cellule numéro p). Si $\delta(x,c) = (x',c',l) \in \mathbb{X} \times \mathbb{B} \times \{-1,1\}$, alors, à l'instant $t+1$, la configuration sera $S_{t+1} = (x', \beta', p+l)$ où $\beta' = b_1 \dots b_{p-1} c' b_{p+1} \cdots$. Si la tête se trouve en première position ($P_t = 1$) et elle essaie de se déplacer à gauche, la machine s'arrête. Par ailleurs, comme $\sqcup \notin \mathbb{A}$, lorsque la tête rencontre le premier caractère $\sqcup$ à droite, cela signifie la fin de l'entrée. La fonction de transition δ induit donc un système dynamique sur $\mathbb{S}$.

Définition 6.2.2. Soit $\tau := \tau_M(\alpha) = \inf\{t \geq 0 : X_t \in \mathbb{F}\} \in \mathbb{N} \cup \{+\infty\}$ le **temps d'arrêt** de la machine [2]. Si $\tau < \infty$ et

— si $X_\tau = x_{\text{acc}}$ alors l'entrée α est acceptée,

— si $X_\tau = x_{\text{rej}}$ alors l'entrée α est rejetée.

Si α est accepté, et on enlève du mot B_τ qui se trouve sur le ruban la plus grande suite infinie de $\sqcup$ qui le complète à droite, nous obtenons un mot fini $\gamma \in \mathbb{B}^*$ qui contient le résultat final du calcul correspondant à l'entrée α.

Remarque 6.2.3. D'après cette définition donc, une machine de Turing, pas seulement elle accepte des mots d'un langage qui lui est spécifique mais implémente aussi une application [3] *partielle* $\mathbb{A}^* \ni \alpha \mapsto \mathsf{Tur}(\alpha) = \gamma \in \mathbb{B}^*$ dont le domaine de définition est

$$\mathsf{dom}(\mathsf{Tur}) = \{\alpha \in \mathbb{A}^* : \tau = \tau(\alpha) < \infty, X_\tau = x_{\text{acc}}\}.$$

Il est évident, que si $\tau(\alpha) = \infty$, alors la machine est entrée dans un calcul sans fin et elle ne peut pas décider si α appartient au langage.

Définition 6.2.4. Soit M une machine de Turing.

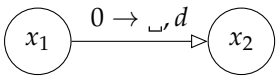
1. On appelle **langage reconnu** par M l'ensemble de mots
$$\mathcal{L}(M) = \{\alpha \in \mathbb{A}^* : \tau(\alpha) < \infty \text{ et } X_{\tau(\alpha)} = x_{\text{acc}}\}.$$
2. La classe de langages pour lesquels il existe une machine de Turing qui les reconnaît constitue la classe des **langages Turing-reconnaissables**.
3. Si pour tout $\alpha \in \mathbb{A}^*$, nous avons $\tau(\alpha) < \infty$, la machine M est dite **décideuse**. Le langage accepté par une machine de Turing décideuse est dit **décidé**. Un langage est dit **Turing-décidable** s'il existe une machine de Turing décideuse qui l'accepte.

Il est évident que la classe des langages Turing-décidables est contenue dans celle des langages Turing-reconnaissables.

Exemple 6.2.5. Le langage $L = \{0^{2^n}, n \in \mathbb{N}\}$ est Turing décidable. En effet, on peut considérer la machine de Turing présentée dans la figure 6.1 ayant comme alphabet de langage le singleton $\mathbb{A} = \{0\}$, comme alphabet de ruban l'ensemble $\mathbb{B} = \mathbb{A} \sqcup \{\checkmark, \sqcup\}$ et comme fonction de transition l'application définie par les étiquettes des arêtes [4].

2. En réalité, $t \geq 1$ car $x_0 \notin \mathbb{F}$.

3. Cette application n'a pas nécessairement d'utilité ; elle peut juste contenir les calculs intermédiaires qui ont servi pour décider si α appartient au langage.

4. Dans le schéma 6.1 de la machine de Turing, l'étiquetage des arêtes est évident. Par exemple, 

signifie que $\delta(x_1, 0) = (x_2, \sqcup, d)$.

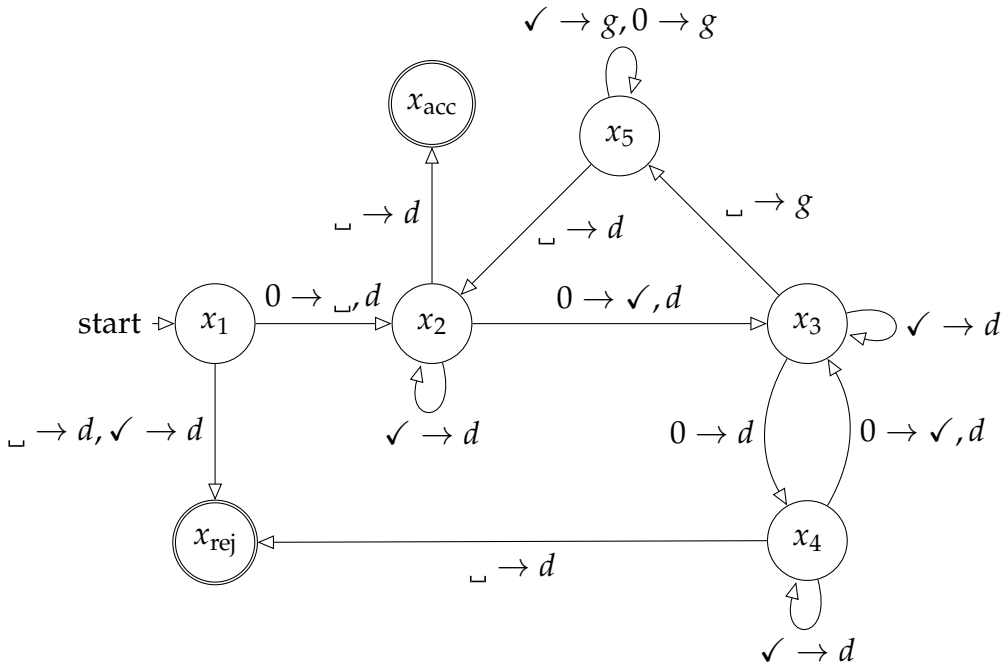


FIGURE 6.1 – Schéma de la machine de Turing reconnaissant le langage $\{0^{2^n}, n \in \mathbb{N}\}$.

Il faut noter qu'il existe plusieurs variantes de la machine de Turing de base :

— l'ensemble des déplacements de la tête $\{g, d\}$ peut être enrichi par le « mouvement consistant à rester sur place », i.e. devenir $\{g, d, s\}$ (alors $\delta : \mathbb{X} \times \mathbb{B} \to \mathbb{X} \times \mathbb{B} \times \{g, d, s\}$) ;

— la machine peut devenir multi-rubans (alors $\delta : \mathbb{X} \times \mathbb{B}^r \to \mathbb{X} \times \mathbb{B}^r \times \{g, d, s\}^r$, où r le nombre de rubans) ;

— la machine peut devenir non-déterministe (alors $\delta : \mathbb{X} \times \mathbb{B} \to \mathcal{P}(\mathbb{X} \times \mathbb{B} \times \{g, d, s\})$). La classe des machines de Turing non-déterministes est notée MTN (par opposition à la classe de machines déterministes, notée MTD).

Toutes ces variantes ont la même puissance calculatoire qu'une machine déterministe mono-ruban et à déplacements g, d. Pour chaque variante généralisante, il est en effet possible de construire une machine de Turing déterministe mono-ruban et à déplacements g, d qui la simule. Ce résultat s'appelle **robustesse calculatoire** de la machine de Turing. En effet, nous avons

Théorème 6.2.6. *Pour toute $M \in$ MTN, il existe une $M' \in$ MTD qui lui est équivalente.*

Idée de la démonstration : Nous devons faire l'exploration des toutes les trajectoires calculatoires engendrées par le digraphe de M (comme nous l'avons fait pour démontrer l'équivalence entre AFN et AFN). Cependant, il y a un piège à éviter dans le cas présent. Pour les automates finis, toutes les trajectoires de calcul sont finies ; par conséquent, il n'y a aucune obstruction d'explorer chacune d'elles en profondeur jusqu'à ce qu'elle aboutisse dans un état de $\mathbb{F}$ ou qu'elle s'évanouisse. Ici, certaines trajectoires peuvent être

infinies. Par conséquent il faut explorer l'arbre des trajectoires niveau par niveau. Dès qu'à un niveau, une trajectoire visite $\mathbb{F}$, la machine s'arrête et décide. □

Corollaire 6.2.7. *Un langage L est Turing-reconnaissable si, et seulement si, une $M \in$ MTN le reconnaît.*

6.3 Définition d'un algorithme

Nous avons tous aujourd'hui une idée intuitive de ce qu'est un algorithme ; on cite spontanément le crible d'Ératosthène pour déterminer les nombres premiers inférieurs à un nombre N ou la méthode d'Euclide pour calculer le pgcd de deux entiers strictement positifs comme des exemples d'algorithmes connus depuis l'antiquité — avant que le mot n'existât. Le mot algorithme, paru vers 1220, n'était cependant pas si précis à ces débuts. Voici les lemmes que l'on trouve pour ce mot dans trois dictionnaires différents (du plus récent au plus ancien) :
ALGORITHME, subst. masc.
• Méthode de calcul qui indique la démarche à suivre pour résoudre une série de problèmes équivalents en appliquant dans un ordre précis une suite finie de règles [5].
• Ensemble des règles opératoires intervenant dans toute espèce de calcul [6].
• Terme didactique. L'art de calculer [7].

On constate que la précision de la définition du terme évolue beaucoup avec le temps ; finalement, la notion n'a été mathématiquement définie qu'au cours du 20e siècle, grâce à l'impulsion initiale donnée par Hilbert avec son fameux 10e problème (cf. page 88), aux travaux de Turing [67] et de Church [15], de logiciens comme Russell [77] et surtout à l'avènement des ordinateurs et l'essor de l'informatique. Nous utiliserons dans la suite le mot « algorithme » comme *synonyme* de machine de Turing.

Dans la description d'une machine de Turing nous pouvons utiliser quatre niveaux.

Bas niveau : où l'on doit donner la construction précise de la machine, comme nous l'avons fait lors de l'étude de l'exemple 6.2.5 ; cette description s'apparente à un programme écrit en langage machine.

Niveau d'implémentation : où, par des phrases, on doit décrire comment la machine se déplace et met à jour les valeurs dans les cellules du ruban (dans la mémoire) ; cette description s'apparente à l'écriture d'un programme en langage de programmation.

Niveau descriptif : où l'on décrit par des phrases les opérations mathématiques que l'on doit effectuer sans se soucier de l'implémen-

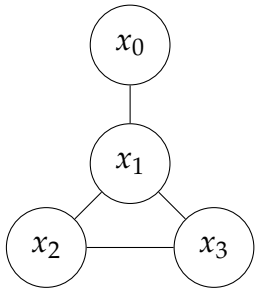
5. Dictionnaire de l'Académie, 9e édition, lemmes de A à MAPPEMONDE publiés, les autres en cours de publication.
6. Trésor de la langue française, CNRS ÉDITIONS, Paris (2004).
7. Dictionnaire de l'Académie, 4e édition, Paris (1762).

tation ; ce niveau s'apparente à un pseudo-code, comme celui donné par exemple lors de l'étude de l'algorithme 5.4.7.

Niveau conceptuel : où la machine est décrite par sa fonctionnalité de principe ; nous admettons ou nous avons déjà démontré l'existence d'une telle machine et nous nous en servons uniquement pour faire des raisonnements logiques.

En entrée, une machine de Turing a *toujours* besoin d'un mot. Si nous voulons appliquer un algorithme sur un objet O plus compliqué, nous devons toujours le coder comme un mot $\langle O \rangle$ d'un alphabet. Nous distinguons donc l'objet O de son codage $\langle O \rangle$.

Exemple 6.3.1. Supposons que $\mathbb{G}$ est un graphe non-dirigé et sans arêtes multiples. Nous souhaitons coder le graphe dans un alphabet. Nous pouvons utiliser l'alphabet quaternaire $\mathbb{A} = \{0, 1, |, \odot\}$. Alors le graphe

$\mathbb{G} :=$ 

pourra être codé dans cet alphabet comme

$$\langle \mathbb{G} \rangle = 0|1|10|11 \odot 0|1 \odot 1|10 \odot 1|11 \odot 10|11.$$

En effet, en éclatant le mot aux séparateurs $\odot$, nous obtenons un premier sous-mot $0|1|10|11$ et 4 autres sous-mots $0|1$, $1|10$, $1|11$ et $10|11$. En éclatant de nouveau ces mots aux séparateurs $|$, on obtient 0 1 10 11 codant les 4 sommets du graphe et 0 1, 1 10, 1 11 et 10 11 codant les paires des sommets qui sont des extrémités des arêtes. À titre d'illustration, voici les instructions (en macros TikZ) qui permettent de générer en LaTeX le graphe utilisé dans cet exemple :

```
\begin{tikzpicture}
\node[state] (0)   {$x_0$};
\node[state] (1) [below=of 0] {$x_1$};
\node[state] (2) [below left=of 1] {$x_2$};
\node[state] (3) [below right=of 1] {$x_3$};
\path[-] (0) edge  (1);
\path[-] (1) edge  (2);
\path[-] (1) edge  (3);
\path[-] (3) edge  (2);
\end{tikzpicture}
```

En faisant abstraction des instructions de placement (comme l'instruction `[below=of 0]`, par exemple) des objets sur le plan, utilisées uniquement

pour imposer une visualisation du graphe, ces instructions ne font que définir les listes des nœuds et des arêtes.

Supposons maintenant que nous veuillons déterminer — par un algorithme — si le graphe G de l'exemple 6.3.1 est connexe. Voici un algorithme présenté au niveau descriptif.

Algorithme 6.3.2. TestConnexité

Require: $\langle\, \mathbb{G} \,\rangle$.
Ensure: propriété de connexité de $\mathbb{G}$.
  $\mathbb{G}^0 \leftarrow$ ensemble des nœuds du graphe
  $\mathbb{G}^1 \leftarrow$ ensemble des arrêtes (non-dirigées) du graphe
  choisir $x \in \mathbb{G}^0$
  $M_0 \leftarrow \{x\}$
  $M_1 \leftarrow \bigcup_{\alpha \in \mathbb{G}^1_x} \alpha$ /* α est une arête non-dirigée, i.e. $\alpha = \{s(\alpha), t(\alpha)\}$ */
  if $M_1 = \mathbb{G}^0$ **then**
    $\mathbb{G}$ est connexe
  else
    repeat
      choisir $x \in M_1 \setminus M_0$
      $M_0 \leftarrow M_0 \cup \{x\}$
      $M_1 \leftarrow \left(\bigcup_{\alpha \in \mathbb{G}^1_x} \alpha\right) \cup M_1$
    until $M_1 = M_0$ or $M_1 = \mathbb{G}^0$
    if $M_1 = \mathbb{G}^0$ **then**
      $\mathbb{G}$ est connexe
    else
      $\mathbb{G}$ est non connexe
    end if
  end if

6.4 Décidabilité, reconnaissance, problème d'arrêt

Nous avons introduit la machine de Turing comme modèle théorique d'un algorithme qui peut être exécuté sur un ordinateur universel. Certains problèmes peuvent être résolus algorithmiquement, d'autres non. Nous allons explorer certains problèmes et utiliser le raisonnement de machine de Turing pour étudier leur résolubilité.

Précisons d'emblée que si M est une machine de Turing, alors on peut toujours la coder en un mot fini $\langle\, M \,\rangle$ sur un alphabet. En effet, une machine de Turing n'est qu'un graphe dirigé étiqueté. Si M utilise comme entrées des mots α d'un certain alphabet $\mathbb{A}$, nous pouvons coder la machine et l'entrée sur le même alphabet et noter $\langle\, M, \alpha \,\rangle$ le code conjoint. Nous rappelons aussi qu'un automate fini est (un cas simple) de machine de Turing.

Voici comment le problème de Hilbert peut être formulé. On note généralement p un polynôme à un nombre arbtraire de variables dont tous les coefficients sont entiers et H le langage

$$H = \{p : p \text{ possède des racines entières }\}.$$

Hilbert demandait (en de termes d'aujourd'hui) si ce langage était décidable (et il pensait qu'oui). Nous pouvons montrer aujourd'hui le

Théorème 6.4.1. *1. H est Turing-reconnaissable.*
2. H n'est pas Turing-décidable.

Idée de la démonstration : Examinons le cas de polynômes d'une seule variable. On cherche à décider le langage H_1 où un polynôme d'une seule variable de degré arbitraire possède des racines entières. On construit une machine de Turing qui reçoit en entrée le mot $\langle\, p \,\rangle$ qui code p. On calcule les valeurs p sur les valeurs successives de la suite $y = (y_n)_{n\in\mathbb{N}}$ ou

$$y_n = \begin{cases} n/2 & \text{si} \quad n \in 2\mathbb{N} \\ -(n+1)/2 & \text{si} \quad n \in 2\mathbb{N}+1. \end{cases}$$

On note $\tau(p) = \inf\{n : p(y_n) = 0\}$. Si p a une racine r entière, alors $\tau(p) < \infty$ et évidemment $r = y_\tau$. Par conséquent, le sous-langage H_1 de H (concernant les polynômes à une variable) est Turing-reconnaissable. Il est par ailleurs facile de démontrer que si $p(x) = b_n x^n + \ldots b_1 x + b_0$ est un polynôme, alors toute racine r de p vérifie l'encadrement $|r| \leq R(p) := (n+1)\frac{c}{|b_n|}$, où $c = \max_{k=0,\ldots,n} |b_k|$. Par conséquent, le problème H_1 devient décidable, car il suffit d'arrêter l'algorithme à l'instant $\tau(p) \wedge R(p)/2$. Le résultat de Matiyasevich démontre précisément que des telles encadrements pour les racines n'existent pas si le polynôme est à plusieurs variables. Par conséquent, le langage H n'est pas décidable dans le cas à plusieurs variables. □

Beaucoup de problèmes algorithmiques peuvent se traduire en des problèmes sur des graphes, tels que connexité du graphe, existence de circuits hamiltoniens, etc. Il est donc utile de commencer par un algorithme de base en théorie des graphes.

L'algorithme 6.3.2 permet d'établir le théorème suivant.

Théorème 6.4.2. *Le langage relatif aux graphes non-dirigés*

$$\Lambda_{\text{CNX}} := \{\langle\, \mathbb{G} \,\rangle : \mathbb{G} \textit{ est connexe}\}$$

est Turing-décidable.

D'autres exemples sont fournis par les langages reconnus par des automates finis. Par exemple, nous avons le

Théorème 6.4.3. *Le langage*

$$\Lambda_{\text{AFD}} = \{\langle\, D, \alpha \,\rangle : D \in \text{AFD } \textit{qui accepte l'entrée } \alpha\}$$

est Turing-décidable.

Démonstration. On construit une machine universelle $M \in \text{MT}$ capable de
- — vérifier que le mot $\langle D, \alpha \rangle$ est une description légitime d'un automate et de son entrée,
- — simuler $D \in \text{AFD}$ et
- — exécuter D sur l'entrée α.

Si la simulation finit dans un état d'acceptation de D, alors la machine s'arrête et accepte le mot $\langle D, \alpha \rangle$; sinon elle s'arrête et le rejette. Comme un automate fini s'arête dans un temps n'excédant pas $|\alpha|$, la machine M est une décideuse du langage Λ_{AFD}. □

Théorème 6.4.4. *Il existe des langages qui ne sont pas Turing-reconnaissables.*

Démonstration. Sans perte de généralité, nous pouvons utiliser le même alphabet $\mathbb{A}$ pour les langages et le codage des machines de Turing. Une machine de Turing M est en définitive un graphe dirigé étiqueté ; elle sera donc codée par un mot fini $\langle M \rangle \in \mathbb{A}^*$. Par conséquent, il existe une bijection entre MT et un sous-ensemble de $\mathbb{A}^*$. Comme $\mathbb{A}^*$ est dénombrable (car il est une réunion dénombrable d'ensemble finis), nécessairement la classe des MT est dénombrable. Or, un langage L est une partie arbitraire de $\mathbb{A}^*$; par conséquent la classe des langages Λ sur l'alphabet $\mathbb{A}$ est en bijection avec $\mathcal{P}(\mathbb{A}^*) \simeq \mathbb{R}$ qui est non-dénombrable. Il n'existe donc pas de surjection possible de l'ensemble des machines de Turing sur l'ensemble des langages car il y a beaucoup plus de langages sur $\mathbb{A}$ que de machines de Turing. □

Théorème 6.4.5. *Le langage*

$$\Lambda_{\text{MT}} = \{\langle M, \alpha \rangle : M \in \text{MT } \textit{qui accepte l'entrée } \alpha\}$$

est Turing-reconnaissable mais il n'est pas Turing-décidable.

Démonstration. Afin d'établir la reconnassabilité, on construit une machine de Turing universelle U capable de
- — vérifier que le mot $\langle M, \alpha \rangle$ est une description légitime d'une machine de Turing et de son entrée,
- — simuler $M \in \text{MT}$ et
- — exécuter M sur l'entrée α.

Si $\tau_M(\alpha) < \infty$ et M s'arrête dans l'état d'acceptation, alors U s'arrête et accepte le mot $\langle M, \alpha \rangle$, si $\tau_M(\alpha) < \infty$ et M s'arrête dans l'état de rejet, alors U s'arrête et rejette le mot $\langle M, \alpha \rangle$. Si $\tau_M(\alpha) = \infty$, alors la machine M ne peut pas décider et par conséquent U non plus.

Pour établir la non-décidabilité, nous supposons que nous pouvons construire une machine de Turing A décideuse du langage Λ_{MT} i.e. qui s'arrête dans un temps fini pour tout mot $\langle M, \alpha \rangle$ et capable de simuler tout $M \in \text{MT}$. Alors A s'arrête et accepte $\langle M, \alpha \rangle$ si M accepte α et s'arrête et rejette $\langle M, \alpha \rangle$ si M n'accepte pas α. Nous construisons ensuite une machine de Turing D universelle et décideuse qui utilise la machine A comme sous-programme. Plus précisément, si $\langle M \rangle$ est une description légitime d'une machine de Turing, lorsqu'on présente à D l'entrée $\langle M \rangle$, elle exécute la routine A sur

le mot $\langle M, \langle M \rangle \rangle$ et elle s'arrête et rejette si A accepte, elle s'arrête et accepte si A rejette. Les ensembles de machines de Turing et des mots qui les décrivent sont dénombrables. On peut donc dresser un tableau à double entrée $(M_k, \langle M_l \rangle)_{k,l\in\mathbb{N}}$ qui contiendra toutes les machines de Turing possibles et toutes les descriptions possibles de machines de Turing. La machine D et sa descrption $\langle D \rangle$ seront donc nécessairement des éléments du tableau.

Maintenant, faisons tourner la machine D sur sa propre description $\langle D \rangle$. Alors

$$D :=\text{« sur entrée } \langle D \rangle, \quad \text{s'exécute sur } \langle D \rangle \text{ et } \begin{cases} \text{accepte } \langle D \rangle & \text{si } D \text{ rejette } \langle D \rangle \\ \text{rejette } \langle D \rangle & \text{si } D \text{ accepte } \langle D \rangle \text{ ».} \end{cases}$$

Nous arrivons à une contradiction car nous avons supposé l'existence de la machine A (d'où découle aussi l'existence de D). Donc A n'existe pas et le problème n'est pas décidable. □

Corollaire 6.4.6. *Le langage $\overline{\Lambda}_{\mathrm{MT}}$ est Turing-non-reconnaissable.*

Démonstration. Nous avons montré dans le théorème 6.4.5 que Λ_{MT} est reconnaissable. Si $\overline{\Lambda}_{\mathrm{MT}}$ était aussi reconnaissable, alors par le résultat montré dans l'exercice 55, le langage Λ_{MT} serait décidable. □

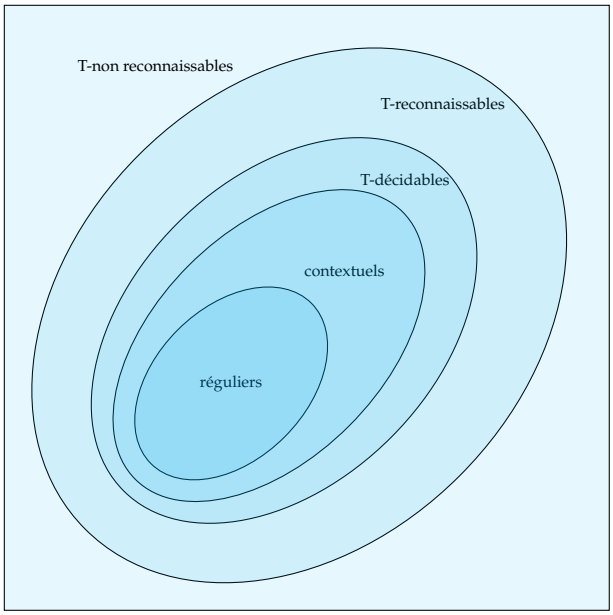


FIGURE 6.2 – Relations entre différentes classes de langages.

6.5 Calculabilité

On rappelle qu'en remarque 6.2.3 nous avons introduit, pour une machine de Turing M, la notion de fonction partielle Tur_M qui associe à chaque

mot $\alpha \in \mathcal{L}(M)$, le mot (privé de la suite infinie de ses blancs finaux) γ que l'on lit sur le ruban à l'instant $\tau_M(\alpha)$ et qui correspond au « résultat du calcul » effectué par la machine.

Définition 6.5.1. Une application partielle $F : \mathbb{A}^* \to \mathbb{A}^*$ est dite **Turing-calculable** s'il existe une machine $M \in \text{MT}$ telle que $F = \mathsf{Tur}_M$ (i.e. $\mathsf{dom}(F) = \mathsf{dom}(\mathsf{Tur}_M)$ et $F(\alpha) = \mathsf{Tur}_M(\alpha)$, pour tout $\alpha \in \mathsf{dom}(F)$).

On peut étendre la notion de calculabilité pour des fonctions définies sur des mots d'un alphabet à des fonctions définies sur $\mathbb{N}$.

Soient $b \geq 2$ et $N \in \mathbb{N}$. On peut toujours représenter N en base b comme $N = \sum_{k=0}^{n} \beta_k b^k$, où

$$n := n(N) = \begin{cases} 0 & \text{si } N = 0, \\ \lfloor \log_b(N) + 1 \rfloor & \text{si } N \geq 1 \end{cases}$$

et $\beta_k \in \mathbb{B}_b := \{0, \ldots, b-1\}$ pour $k = 0, \ldots n$. On note

$$\mathbb{N} \ni N \mapsto \mathsf{rep}_b(N) = \beta = \beta_0 \cdots \beta_{n-1} \in \{0, \ldots, b-1\}^n$$

$$\{0, \ldots, b-1\}^+ \ni \beta \mapsto \mathsf{num}_b(\beta) = \sum_{k=0}^{|\beta|-1} \beta_k b^k \in \mathbb{N}.$$

Les fonctions représentation et valeur numérique, notées respectivement rep_b et num_b ci-dessus, définies pour des bases $b \geq 2$, peuvent-être étendues aussi à la base de numération unaire ($b = 1$) qui consiste à représenter tout entier N comme $N = \underbrace{1 + \ldots + 1}_{N}$. Noter cependant que la représentation unaire d'un entier est exponentiellement plus longue que toute représentation en base b, avec $b \geq 2$.

Définition 6.5.2. À toute fonction $f : \mathbb{N} \to \mathbb{N}$ on associe une fonction $\hat{f}_b : \mathbb{B}_b^+ \to \mathbb{B}_b^+$, définie par

$$\hat{f}_b(\beta) = \mathsf{rep}_b\left(f(\mathsf{num}_b(\beta))\right)$$

qui rend le diagramme suivant commutatif.

$$\begin{array}{ccc} \mathbb{N} & \xrightarrow{\quad f \quad} & \mathbb{N} \\ {\scriptstyle \mathsf{num}_b}\uparrow & & \downarrow{\scriptstyle \mathsf{rep}_b} \\ \mathbb{B}_b^+ & \xrightarrow{\quad \hat{f}_b \quad} & \mathbb{B}_b^+ \end{array}$$

Définition 6.5.3.

1. Soient $X \subseteq \mathbb{N}$ et $b \in \mathbb{N}_>$ arbitraire. L'ensemble X est **Turing-décidable** en base b si le langage $\{\mathsf{rep}_b(x), x \in X\}$ est Turing-décidable.
2. Une fonction $f : \mathbb{N} \to \mathbb{N}$ est **Turing-calculable** si la fonction $\hat{f}_b : \mathbb{B}_b^* \to \mathbb{B}_b^*$ est Turing-calculable en base b, avec $b \geq 2$.

6.6 Exercices

Tests de non-régularité

43. Utiliser le lemme de l'itération (pumping lemma) pour établir que les langages suivants ne sont pas réguliers.
    (a) $L_1 = \{0^n 1^n 2^n : n \in \mathbb{N}\}$.
    (b) $L_2 = \{\alpha\alpha\alpha : \alpha \in \{\mathtt{a},\mathtt{b}\}^*\}$.
    (c) $L_3 = \{\mathtt{a}^{2^n} : n \in \mathbb{N}\}$.
44. Montrer que l'indiscernabilité par rapport à un langage $L \subseteq \mathbb{A}^*$, notée $\equiv_L$, est une relation d'équivalence sur $\mathbb{A}^*$.
45. Soient $L \subseteq \mathbb{A}^*$ un langage et $M = (\mathbb{X}, \mathbb{A}, \delta, x_0, \mathbb{F}) \in \text{AFD}$ un automate fini qui accepte L. On note Δ l'extension de $\delta : \mathbb{X} \times \mathbb{A} \to \mathbb{X}$ en $\Delta : \mathbb{X} \times \mathbb{A}^* \to \mathbb{X}$, où $\Delta_\alpha(x) = \delta_{\alpha_{|\alpha|}} \circ \cdots \circ \delta_{\alpha_1}(x)$.
    (a) On dit que deux mots $\alpha, \beta \in \mathbb{A}^*$ sont indiscernables par rapport à l'automate M si $\Delta_\alpha(x_0) = \Delta_\beta(x_0)$ et on note $\alpha \equiv_M \beta$. Montrer que $\equiv_M$ est une relation d'équivalence sur $\mathbb{A}^*$.
    (b) Montrer que pour $\alpha, \beta \in \mathbb{A}^*$, la relation $\alpha \equiv_M \beta$ implique la relation $\alpha \equiv_L \beta$.
    (c) En conclure le théorème de Myhill-Nerode.
46. Soit $L \subseteq \{a, b, c\}^*$ le langage :
$$L = \{ab^\ell c^\ell, \ell \in \mathbb{N}\} \cup \{a^k b^\ell c^m, k, \ell, m \in \mathbb{N} \wedge k \neq 1\}.$$
    (a) Montrer que $\zeta = ab^k c^k$, avec $k \geq 1$ satisfait les conditions du lemme de l'itération.
    (b) Montrer que $\zeta = aab^k c^\ell$, pour tout $k, \ell \in \mathbb{N}$, vérifie les conditions du lemme de l'itération.
    (c) En examinant quelques autres cas partiels, conlcure que les mots de L, de longueur ≥ 2, satisfont le lemme de l'itération.
    (d) Déterminer $L' = L \cap \mathcal{L}(ab^*c^*)$.
    (e) Montrer que L' ne satisfait pas les conditions du lemme de l'iteration (donc il n'est pas régulier).
    (f) Conclure que L n'est pas régulier. *Suggestion :* Il est rappelé que si R est une expression régulière, alors $\mathcal{L}(R)$ est un langage régulier et que la classe de langages réguliers est fermée par intersection. Faire un raisonnement par l'absurde.
47. *Un langage non-régulier.* [Extrait du contrôle du 21 avril 2021].
    Soit $\mathbb{A} = \mathbb{A}_b \sqcup \mathbb{A}_s$ où $\mathbb{A}_b = \{0, 1\}$ est l'alphabet binaire et $\mathbb{A}_s = \{\oplus, \|\}$ est un alphabet de symboles. À chaque chaîne de bits $X \in \mathbb{A}_b^*$, on associe l'entier dont X est la repésentation binaire. (On suppose que les mots X, Y, Z ne commencent pas par 0 et que le mot vide représente la valeur numérique 0). Le mot sur $\mathbb{A}^*$ de la forme $X \parallel Y \oplus Z$ est interprété comme une égalité numérique entre les valeurs, i.e. $\mathsf{num}(X) = \mathsf{num}(Y) + \mathsf{num}(Z)$. Par exemple $10010 \parallel 1011 \oplus 101$ est interprétée comme l'égalité numérique (fausse !) $18 = 11 + 5$.

Montrer que le langage L sur $\mathbb{A}^*$ défini par

$$L = \{X \parallel Y \oplus Z \in \mathbb{A}^*, X, Y, Z \in \mathcal{L}(\varepsilon \cup 1\mathbb{A}_b^*) \text{ et } \mathsf{num}(X) = \mathsf{num}(Y) + \mathsf{num}(Z) \text{ est vraie}\}$$

n'est pas régulier.

48. *La fonction de croissance d'un langage régulier ne peut pas être une fonction arbitraire.* [Extrait du contrôle du 21 avril 2021].

    (a) Soit $L \subseteq \mathbb{A}^*$ un langage arbitraire. L'application $\kappa_L : \mathbb{N} \to \mathbb{N}$, définie par $\kappa_L(n) = |L \cap \mathbb{A}^n|$, est appelée **fonction de croissance** de L. Pour un n fixé, majorer et minorer $\kappa_L(n)$; ces bornes sont-elles saturées ?

    (b) Soit le langage $L = \bigcup_{r:r \text{ premier}} \mathbb{A}^r$, $\mathbb{A}$ un alphabet fini arbitraire avec $|\mathbb{A}| > 1$. Monter que L n'est pas régulier.

    (c) Conclure qu'il existe des fonctions $f : \mathbb{N} \to \mathbb{N}$ qui ne peuvent être des fonctions de croissance d'aucun langage régulier. *Suggestion :* il suffit d'en exhiber une.

49. *Langages bornés.* [Extrait du devoir maison du 22 msrs 2020].
    Définition A : Soient $\mathbb{A}$ un alphabet fini de cardinal $|\mathbb{A}| \geq 2$ et $L \subset \mathbb{A}^*$ un langage.
    — Pour un $k \in \mathbb{N}_>$, on dit que le langage L est k**-borné**, s'il existe une famille génératrice de k mots $\beta_1, \ldots, \beta_k \in \mathbb{A}^*$ tels que

    $$\forall \alpha \in L, \exists p_1 := p_1(\alpha) \in \mathbb{N}, \ldots \exists p_k := p_k(\alpha) \in \mathbb{N} : \alpha = \beta_1^{p_1} \cdots \beta_k^{p_k}.$$

    On note alors : $L \subseteq \beta_1^* \cdots \beta_k^*$.
    — L est dit **borné**, s'il existe un $k \in \mathbb{N}_>$ pour lequel L est k-borné.

    Définition B : Soit $\alpha = a_1 \cdots a_n$ un mot sur l'alphabet $\mathbb{A}$ avec $|\mathbb{A}| \geq 2$. L'image miroir de α, notée $\overleftarrow{\alpha}$, est définie par

    $$\overleftarrow{\alpha} = a_n \cdots a_1.$$

    L'image miroir du langage est définie par $\overleftarrow{L} = \{\overleftarrow{\alpha} : \alpha \in L\}$.

    Définition C : Soient $L \subseteq \mathbb{A}^*$ et $\alpha \in L \setminus \{\varepsilon\}$. On note

    $$\alpha^\infty = \alpha\alpha \cdots \in \mathbb{A}^{\mathbb{N}}$$

    le mot périodique infini obtenu par concaténations indéfiniment répétées de α.
    Soit $\mathbb{A}$ un alphabet fini avec $|\mathbb{A}| \geq 2$.

    (a) Si $L \subseteq \mathbb{A}^*$ est borné, montrer que $\overleftarrow{L}$ l'est aussi.
    (b) Si les langages $L, M \subseteq \mathbb{A}^*$ sont bornés, montrer que leur concaténation $LM := \{\alpha\beta : \alpha \in L, \beta \in M\}$ et leur réunion $L \cup M$ le sont aussi.

(c) Le but de cette question est de montrer la

Proposition : Les deux affirmations suivantes sont équivalentes :
— L est 1-borné.
— $\forall \alpha, \gamma \in L, \alpha\gamma = \gamma\alpha$.

i. Montrer que si L est 1-borné, alors $\forall \alpha, \gamma \in L, \alpha\gamma = \gamma\alpha$.
ii. Soient $\alpha, \gamma \in L \setminus \{\varepsilon\}$ et $k \in \mathbb{N}$ arbitraires. Montrer qu'il existe $\ell_0 \in \mathbb{N}$, tel que $|\gamma^{\ell_0}| \geq |\alpha^k|$.
iii. En conclure que si $\alpha\gamma = \gamma\alpha$, alors $\alpha^\infty = \gamma^\infty$.
iv. Dans le cas où la concaténation est commutative, il existe donc un mot périodique infini $\xi = \alpha^\infty$. On note β le préfixe (vérifiant nécessairement $1 \leq |\beta| < \infty$) de ξ qui correspond à la plus petite période. Montrer que pour tout $\gamma \in L$, il existe un $p \in \mathbb{N}$ tel que $\gamma = \beta^p$.
v. Conclure que si la concaténation des mots de L est commutative, alors L est 1-borné.

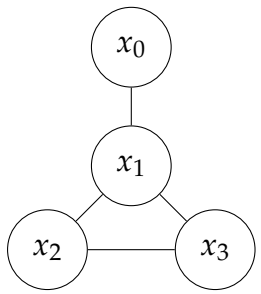
(d) En supposant que $\mathbb{A}$ contient au moins les lettres a, b, définir

$$\gamma_n = \prod_{\ell=0}^{n} (ab^\ell) = aabab^2ab^3 \cdots ab^n, n \in \mathbb{N}.$$

i. Montrer que pour tout $p \geq 1$ et $n \in \mathbb{N}$, le mot γ_n est un préfixe de γ_{n+p} et on désigne par $\sigma_{n,p}$ le suffixe qu'il faut concaténer à γ_n pour obtenir γ_{n+p}. On écrit $\gamma_n \prec \gamma_{n+p}$ et $\gamma_{n+p} = \gamma_n\sigma_{n,p}$.
ii. En supposant que $L = \{\gamma_n, n \in \mathbb{N}\}$ soit borné, il existerait un entier $k \geq 1$ minimal tel que L serait k-borné. Montrer que ceci est impossible.
iii. Conclure qu'il existe des langages non-bornés, en établissant que le langage $L = \{\gamma_n, n \in \mathbb{N}\}$ n'est pas borné.

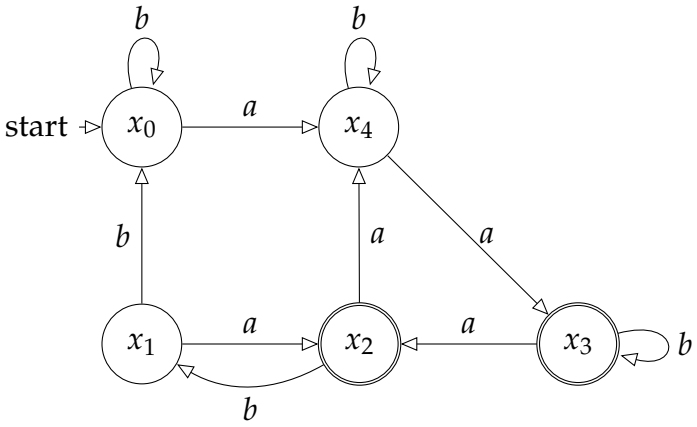
Machines de Turing, décidabilité, faculté de reconnaissance

50. Pour la machine de Turing décideuse du langage $L = \{0^{2^n} : n \in \mathbb{N}\}$ (décrite par le digraphe étiquetté donné dans le texte) et pour l'entrée $\alpha = 0000$, déterminer la suite complète $(S_t)_{t=1,\ldots,\tau(\alpha)}$ des configurations d'où passe la machine avant d'accepter le mot α. *Indication :* $\tau(\alpha) = 21$.
51. Coder, en alphabets ternaire et binaire, le graphe

$\mathbb{G} :=$ 

52. Soit $M = (\mathbb{X}, \mathbb{A}, \delta, x_0, \mathbb{F}) \in$ AFD décrit par le graphe

$M :=$



(a) Représenter l'automate M par un mot $\beta \in \{0,1\}^*$, i.e. $\langle M \rangle = \beta$. *Indication :* $|\mathbb{X}| = 5, |\mathbb{A}| = 2$. Par conséquent, un code sur 3 bits pour *chaque* élément constitutif du graphe est suffisant.
(b) L'automate M accepte-t-il sa propre description $\langle M \rangle$?

53. Modifier l'algorithme **TestConnexité** pour tester la connexité d'un graphe dirigé.

54. Montrer que les langages suivants sont décidables :

(a) $\Lambda_{\text{AFN}} = \{\langle N, \alpha \rangle : N \in \text{AFN}$ qui accepte $\alpha\}$.
(b) $\Lambda_{\text{ExpReg}} = \{\langle R, \alpha \rangle : R \in \text{ExpReg}$ qui accepte $\alpha\}$.
(c) $\Lambda_{\text{VIDE}} = \{\langle V \rangle : V \in \text{AFD}, \mathcal{L}(V) = \emptyset\}$.
(d) $\Lambda_{\text{EQUIV}} = \{\langle A, B \rangle : A, B \in \text{AFD}, \mathcal{L}(A) = \mathcal{L}(B)\}$.

55. Montrer qu'un langage L sur $\mathbb{A}^*$ est Turing-décidable si, et seulement si, les langages L et $\overline{L}$ sont Turing-reconnaissables, où $\overline{L} = \mathbb{A}^* \setminus L$.

7 Complexité

Au chapitre précédent, nous avons examiné la décidabilité de principe d'un langage. Dans ce chapitre nous allons étudier la question : combien de temps faut-il pour décider si une entrée arbitraire appartient au langage. Commençons par un exemple.

Soit le langage $L = \{0^n 1^n, n \in \mathbb{N}\}$. Ce langage est décidable, il existe donc une machine de Turing M qui, pour tout entrée finie α décide si $\alpha \in L$. Pour ce faire M doit une première fois balayer α et examiner si un 1 se trouve avant un 0 (cette opération nécessite au pire $|\alpha|$ déplacements de la tête) ; si c'est le cas, le mot est rejeté, sinon, la tête se déplace de nouveau au début du mot (cela nécessite, au pire, $|\alpha|$ opérations supplémentaires). Ensuite, elle balaie le mot et raie lors de chaque balayage un 0 et un 1. Puisque chaque balayage enlève 2 symboles, au plus $|\alpha|/2$ balayages seront nécessaires, donc au total $|\alpha|^2/2$ sont nécessaires. Finalement, un balayage sera nécessaire pour décider s'il y a des 0 ou des 1 résiduels (non rayés), opération ayant un coût supplémentaire de $|\alpha|$ opérations. Lorsque $|\alpha|$ est grand, le coût total de la décision de ce langage sera donc proportionnel à $|\alpha|^2$. Ici nous avons estimé le coût du cas le plus défavorable. Nous aurions pu estimer le coût moyen pour des entrées aléatoires.

7.1 Classe de complexité P

Définition 7.1.1. Soit $M \in \text{MT}$ une machine décideuse d'un langage L. Le **temps de calcul** ou la **complexité algorithmique temporelle** de M (ou de L) est la fonction $f : \mathbb{N} \to \mathbb{N}$, où $f(n)$ est le nombre maximal d'étapes que doit effectuer M sur une entrée de longueur $|\alpha| = n$.

Notation 7.1.2. Soient $f, g : \mathbb{N} \to \mathbb{N}$. Nous disons

— $f(n) = \mathcal{O}(g(n))$ s'il existe d'entiers strictement positifs n_0 et C tels que $n \geq n_0 \Rightarrow f(n) \leq Cg(n)$.
— $f(n) = o(g(n))$ si $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ ou $\forall \varepsilon > 0, \exists n_0 : n \geq n_0 \Rightarrow f(n) < \varepsilon g(n)$.

Ainsi, $5n^3 + 12n + 8 = \mathcal{O}(n^3)$ tandis que $n = o(n \log n)$.

Définition 7.1.3. Soit $t : \mathbb{N} \to \mathbb{R}_+$. On désigne par $\text{TIME}(t)$ (plus précisément $\text{DTIME}(t)$) la classe de problèmes à complexité algorithmique temporelle t, i.e. les langages L pour lesquels il existe $M \in \text{MTD}$ décideuse qui dans un temps n'excédant pas $t(|\alpha|)$ décide si α appartient au langage.

Exemple 7.1.4. $L = \{0^n 1^n, n \in \mathbb{N}\} \in \text{TIME}(n^2)$. (On note abusivement n^2 la fonction définie par la formule $n \mapsto n^2$).

Définition 7.1.5. La **classe de complexité P** est la classe de langages décidables par une machine de Turing déterministe en temps polynomial, i.e.

$$\text{P} = \cup_{k\in\mathbb{N}} \text{TIME}(n^k).$$

Exemple 7.1.6. Soient $\mathbb{G} = (\mathbb{G}^0, \mathbb{G}^1)$ un graphe dirigé et $x, y \in \mathbb{G}^0$. Alors

$$\Lambda_{\text{PATH}} = \{\langle \mathbb{G}, x, y \rangle : \exists \alpha \in \mathbb{G}^*, s(\alpha) = x, t(\alpha) = y\}$$

est décidable en temps polynomial en $|\mathbb{G}^0|$.

Il est à noter qu'un algorithme de « force brute » pour décider le langage précédent a un temps de calcul sur-exponentiel. En effet, l'algorithme de « force brute » consiste à examiner toutes les trajectoires potentielles pour vérifier si elles connectent x à y. Si $n = |\mathbb{G}^0|$, les trajectoires potentielles ont nécessairement une longueur $\leq n$. Pour explorer les trajectoires, on aura besoin d'au plus n^n étapes. Or $t : n \mapsto n^n$ correspond à une complexité sur-exponentielle. Cependant, on peut montrer que ce problème est dans la classe P car il existe un algorithme de complexité polynomiale (cf. exercice 56) pour le résoudre.

La classe de complexité polynomiale ne différencie pas les algorithmes de complexité linéaire ou quadratique ou cubique ou etc. Pour résoudre un même problème de classe P, nous disposons souvent de plusieurs algorithmes de classe polynomiale avec des puissances différentes. Par exemple, en recourant à l'utilisation des machines multi-rubans (cf. exercice 57), on peut diminuer la puissance comme le montre le

Théorème 7.1.7. *Soit $t : \mathbb{N} \to \mathbb{R}_+$ une application avec $t(n) \geq n$. Toute machine de Turing $M \in \text{MT}$ multi-ruban à complexité $\mathcal{O}(t(n))$ est équivalente à une machine de Turing mono-ruban M' à complexité $\mathcal{O}(t^2(n))$.*

Remarque 7.1.8. Le théorème précédent garantit que le passage d'une machine mono-ruban à une machine multi-ruban est une opération qui ne modifie fondamentalement la complexité du problème (on reste à l'intérieur de la classe P). La classe P regroupe les problèmes qui peuvent être raisonnablement résolus sur ordinateur.

Si le langage que nous étudions est non-régulier, sa complexité ne peut pas être arbitrairement petite comme le montre le

Théorème 7.1.9. *[36, Théorème 3.3, p. 188] Soit $t(n) = o(n \log n)$. Alors la classe $\mathrm{DTIME}_1(t(n))$ ne contient que des langages réguliers (où $\mathrm{DTIME}_k(t)$ désigne la classe des langages décidés par une machine déterministe à k rubans).*

La classe des algorithmes de classe $\mathrm{DTIME}(n \log n)$ contient les problèmes quasilinéaires. Cette classe peut être aussi définie comme $\mathrm{DTIME}(n \log n) = \cap_{k \in \mathbb{N}_>} \mathrm{DTIME}(n^{1+\frac{1}{k}})$. Un exemple d'algorithme quasilinéaire est l'algorithme (voir par exemple [50, p. 45]) MergeSort de rangement récursif d'une liste par séparation en sous-listes élémentaires et fusions des sous-listes par comparaison.

7.2 Machines de Turing non-déterministes et classe de complexité NP

Nous avons vu que la notion d'automate fini non-déterministe est une généralisation très importante qui permet de simplifier certaines démonstrations. Comme une machine de Turing est aussi un automate, une généralisation analogue vers des machines de Turing non-déterministes est possible. Une $M \in \mathrm{MTD}$ induit une chaîne de Markov déterministe $(S_t)_{t \in \mathbb{N}}$ sur l'espace des configurations $\mathbb{S} = \mathbb{X} \times \mathbb{B}^{\mathbb{N}} \times \mathbb{N}_>$, totalement déterminée par la fonction de transition $\delta : \mathbb{X} \times \mathbb{B} \to \mathbb{X} \times \mathbb{B} \times \{-1, 1\}$, i.e. pour $\beta = b_1 \cdots b_{p-1} c b_{p+1} \cdots$ et $\beta' = b_1 \cdots b_{p-1} c' b_{p+1} \cdots$,

$$\mathbb{P}\left(S_{t+1} = (y, \beta', p+d) | S_t = (x, \beta, p)\right) = \begin{cases} 1 & \text{si } \delta(x,c) = (y, c', d), \\ 0 & \text{sinon.} \end{cases}$$

Une machine de Turing (déterministe) est essentiellement équivalente à la donnée de la fonction de transition $\delta : \mathbb{X} \times \mathbb{B} \to \mathbb{X} \times \mathbb{B} \times \{-1, 1\}$. Pour chaque mot $\alpha \in \mathbb{A}^*$ présenté en entrée, complété en un mot de $\mathbb{B}^{\mathbb{N}}$ obtenu de α par une suite infinie finale de $\sqcup$, la fonction δ détermine un système dynamique (S_n), où $S_n = (X_n, A_n, P_n) \in \mathbb{X} \times \mathbb{B}^* \times \mathbb{N}$. La machine est décideuse si elle s'arrête en temps fini pour tout mot d'entrée, i.e. pour tout $\alpha \in \mathbb{A}^*$, on a $\tau(\alpha) < \infty$.

Définition 7.2.1. Une **machine de Turing non-déterministe** M est le sextuplet $(\mathbb{X}, \mathbb{A}, \mathbb{B}, \delta, x_0, \mathbb{F})$ où

— $\mathbb{X}$ est l'ensemble des états internes,

— $\mathbb{A}$ est l'alphabet de codage du langage de l'entrée,
— $\mathbb{B} \supset \mathbb{A}$ est l'alphabet étendu du ruban,
— $\delta : \mathbb{X} \times \mathbb{B} \to \mathcal{P}(\mathbb{X} \times \mathbb{B} \times \{g, d\})$ est la fonction de transition,
— x_0 est l'état initial,
— $\mathbb{F} = \mathbb{F}_{\text{acc}} \sqcup \mathbb{F}_{\text{rej}}$ est l'ensemble des états d'arrêt, scindé en $\mathbb{F}_{\text{acc}} = \{x_{\text{acc}}\}$ et $\mathbb{F}_{\text{rej}} = \{x_{\text{rej}}\}$ (on suppose toujours que $x_0 \neq \mathbb{F}$).

La classe des machines de Turing non-déterministes est notée MTN.

Une machine non-déterministe diffère du fait que sa fonction de transition est à valeurs dans $\mathcal{P}(\mathbb{X} \times \mathbb{B} \times \{g, d\})$. Introduire le **degré maximal de branchement** (dû au non-déterminisme)

$$\mathsf{dmb} = \max_{x \in \mathbb{X}, b \in \mathbb{B}} |\delta(x, b)|.$$

L'évolution, au lieu d'être un système dynamique sur $\mathbb{S}$ devient maintenant un **arbre des trajectoires calculatoires** possibles sur $\mathbb{S}$, dont chaque nœud a au plus dmb descendants immédiats. Cet arbre est entièrement déterminé par le mot α en entrée de la machine et sera noté $\mathsf{ATC}(\alpha)$; il peut avoir des branches finies ou infinies qui se terminent par un ensemble de feuilles, noté $\mathsf{Feuilles}(\alpha)$, isomorphe à une partie de $\mathbb{S}^* \sqcup \mathbb{S}^{\mathbb{N}}$. Cette généralisation permet une classification plus fine de la complexité algorithmique.

Définition 7.2.2. Soit $\sigma = (S_t)_{t=0,\dots,|\sigma|}$. Pour chaque $S \in \mathbb{S}$, on note $X := X(S)$, $B := B(S)$ et $P := P(S)$ les projections de S sur respectivement $\mathbb{X}$, $\mathbb{B}^{\mathbb{N}}$ et $\mathbb{N}_>$. On définit alors

$$\sigma \in \mathsf{Feuilles}(\alpha) \Leftrightarrow \begin{cases} X_k \in \mathbb{F} \text{ ou } \delta(X_k, B_{P_k}) = \emptyset & \text{si } |\sigma| = k \in \mathbb{N} \\ \forall k \in \mathbb{N}, X_k \notin \mathbb{F} & \text{si } |\sigma| = \infty. \end{cases}$$

1. La machine M **accepte** $\alpha \in \mathbb{A}^*$ s'il existe $\sigma \in \mathsf{Feuilles}(\alpha)$ tel que $|\sigma| < \infty$ et $X_{|\sigma|} = x_{\text{acc}}$.
2. La machine M **n'accepte pas** $\alpha \in \mathbb{A}^*$ si pour tout $\sigma \in \mathsf{Feuilles}(\alpha)$, soit $|\sigma| < \infty$ et $X_{|\sigma|} \neq x_{\text{acc}}$, soit $|\sigma| = \infty$.

En introduisant la convention que X_∞ est un état supplémentaire tel que $X_\infty \notin \mathbb{X}$, on peut raccourcir les définitions d'acceptation et de non-acceptation en écrivant

— M **accepte** $\alpha \in \mathbb{A}^*$ s'il existe $\sigma \in \mathsf{Feuilles}(\alpha)$ tel que $X_{|\sigma|} = x_{\text{acc}}$.
— M **n'accepte pas** $\alpha \in \mathbb{A}^*$ si pour tout $\sigma \in \mathsf{Feuilles}(\alpha)$ on a $X_{|\sigma|} \neq x_{\text{acc}}$.

Définition 7.2.3. Soit $N \in$ MTN.

1. N est dite **décideuse** si pour tout $\alpha \in \mathbb{A}^*$ on a $\mathsf{card}(\mathsf{Feuilles}(\alpha)) < \infty$ et pour tout $\sigma \in \mathsf{Feuilles}(\alpha)$, on a $X_{|\sigma|} \in \mathbb{F}$.
2. Si N est décideuse, le **temps d'arrêt** de N sur l'entrée α est défini par

$$\tau_N(\alpha) := \inf\{|\sigma|, \sigma \in \mathsf{Feuilles}(\alpha)\}.$$

3. On définit la **fonction « temps de calcul »** $t_N : \mathbb{N} \to \mathbb{N}$ d'une machine décideuse N par $t_N(n) := \max_{\alpha \in \mathbb{A}^n} \tau_N(\alpha)$.

4. Un langage $L \subseteq \mathbb{A}^*$ est dit **semi-décidé** s'il existe $N \in \text{MTN}$, tel que

$$\alpha \in L \Rightarrow N \text{ accepte } \alpha$$
$$\alpha \notin L \Rightarrow N \text{ n'accepte pas } \alpha.$$

Définition 7.2.4. Soit $t : \mathbb{N} \to \mathbb{N}$. On désigne par $\text{NTIME}(t)$ la classe des langages L pour lesquels il existe une machine de Turing non-déterministe $N \in \text{MTN}$ décideuse qui dans un temps n'excédant pas $t(|\alpha|)$ décide si α appartient au langage. On désigne par NP la classe de langages qui peuvent être décidés en temps polynomial par une machine non-déterministe, i.e.

$$\text{NP} = \cup_{k\in\mathbb{N}}\text{NTIME}(n^k).$$

Remarque 7.2.5. Il est très difficile d'appréhender la nature des langages couverts par la définition 7.2.4 car nous n'avons pas d'intuition de ce que fait précisément une machine de Turing non-déterministe. Le théorème 7.2.6 laisse penser qu'ils sont beaucoup plus compliqués que les langages de la classe P. Nous donnerons dans la définition 7.2.8 une autre définition de la classe NP, plus facile à appréhender, et nous montrerons le théorème 7.2.10 qui établit l'équivalence de deux définitions.

Théorème 7.2.6. *Pour toute machine $N \in \text{MTN}$, ayant une fonction « temps de calcul » t_N, il existe une machine $M \in \text{MTD}$ ayant une fonction « temps de calcul » $t_M = 2^{\mathcal{O}(t_N)}$.*

Idée de la démonstration : Tout nœud dans l'arbre d'exploration de la machine N a au plus dmb descendants (le degré maximal de branchement déterminé par la fonction δ de N). Donc, pour une entrée α, il y a au plus $\text{dmb}^{t_N(|\alpha|)}$ branches à explorer. Par ailleurs, chaque branche a une longueur maximale de $t_N(|\alpha|)$. Le temps de calcul de la machine déterministe simulant N aura donc un temps de calcul $\mathcal{O}(t_N(|\alpha|)\text{dmb}^{t_N(|\alpha|)}) = 2^{\mathcal{O}(t_N(|\alpha|)}$. □

Le langage décrit dans l'exemple 7.1.6 peut-être décidé en temps polynomial si nous faisons l'effort de chercher un algorithme plus subtil que la force brute. Il y a cependant des problèmes pour lesquels *nous ne connaissons pas* d'algorithme permettant de les résoudre en temps polynomial mais si un candidat de solution nous est présenté, nous pouvons vérifier en temps polynomial s'il s'agit effectivement d'une solution.

Exemple 7.2.7. Soient $\mathbb{G} = (\mathbb{G}^0, \mathbb{G}^1)$ un graphe dirigé et $x, y \in \mathbb{G}^0$. Nous rappelons qu'un chemin de $\mathbb{G}^*$ et dit est dit couvrant s'il passe par tous les points de $\mathbb{G}^0$, il est sans recoupement, s'il passe au plus une fois par chaque point de $\mathbb{G}^0$, il est hamiltonien s'il est couvrant et sans recoupement, i.e. il passe exactement une fois par chaque point de $\mathbb{G}^0$. Nous ne connaissons pas si le langage

$$\Lambda_{\text{HAMPATH}} = \{\langle \mathbb{G}, x, y \rangle : \exists \alpha \in \mathbb{G}^*, s(\alpha) = x, t(\alpha) = y, \alpha \text{ hamiltonien } \}$$

est décidable en temps polynomial en $|\mathbb{G}^0|$. Cependant, si α une trajectoire de $\mathbb{G}^*$ reliant x à y nous est donnée, alors nous pouvons vérifier en temps $\mathcal{O}(|\mathbb{G}^0|^2)$ si elle constitue un chemin hamiltonien couvrant.

Définition 7.2.8. — Soit $L \subset \mathbb{A}^*$ un langage. On dit qu'une machine de Turing déterministe V est une **vérificatrice de l'appartenance au langage** si pour tout $\alpha \in L$, il existe un mot γ (le certificat de α) tel que le mot $\langle \alpha, \gamma \rangle$ est accepté par V.
— Un langage appartient à la **classe de complexité NP** si l'appartenance au langage est vérifée dans un temps polynomial.

Exemple 7.2.9. Soit Λ_{HAMPATH} langage défini dans l'exemple 7.2.7. Une vérificatrice est une machine déterministe V qui, si un certificat (chemin) γ est présenté, elle place ce chemin sur le graphe $\mathbb{G}$, codé par le mot $\langle \mathbb{G} \rangle$ et décide dans un temps polynomial si γ est sans recoupement et couvrant.

Revenons encore une fois sur la définition 7.2.8. Un langage L est de complexité NP s'il existe une machine déterministe V telle que pour tout objet décrit par le mot $\alpha \in L$, on peut trouver un mot γ tel que $\langle \alpha, \gamma \rangle$ est accepté par V dans un temps qui est polynomial en $|\alpha|$ et pour tout $\alpha \notin L$ et tout γ, l'objet $\langle \alpha, \gamma \rangle$ est rejeté par V dans un temps polynomial en $|\alpha|$.

Théorème 7.2.10. *Les définitions 7.2.4 et 7.2.8 de la classe NP sont équivalentes.*

Démonstration. Supposons qu'un langage L puisse être décidé par une machine $N \in \text{MTN}$ en temps polynomial. Cela veut dire que pour toute entrée $\alpha \in L$ et à chaque étape du calcul, la machine effectue un choix non-déterministe de la branche (de la fonction multivaluée δ) qui donne naissance à une branche spécifique dans l'arbre des trajectoires calculatoires qui acceptera ce mot. Sans perte de généralité, on peut supposer que la suite des choix effectués est consigné dans le mot γ (qui peut alors constituer le certificat de α). On simule alors une machine vérificatrice V comme suit :

$V =$ « Entrée $\langle \alpha, \gamma \rangle$:
Simuler N sur entrée α.
Se servir de γ pour faire le choix de chaque branchement.
Si cette branche spécifique accepte, alors accepter, sinon rejeter. »

Réciproquement, supposons que V est une vérificatrice en temps $\mathcal{O}(|\alpha|^k)$. On simule alors une machine $N \in \text{MTN}$ comme suit :

$N =$ « Entrée α :
Choisir γ, avec $|\gamma| \leq |\alpha|^k$, de manière non-déterministe.
Exécuter V sur $\langle \alpha, \gamma \rangle$.
Si V accepte, alors accepter, sinon rejeter. »

□

7.3 Complétude NP

Il est évident que $\text{P} \subseteq \text{NP}$. Par ailleurs, le meilleur algorithme que nous disposons pour décider si un langage est dans la classe NP est de complexité exponentielle, signifiant que

$$\text{P} \subseteq \text{NP} \subseteq \text{EXPTIME} = \cup_{k \in \mathbb{N}} \text{TIME}(2^{n^k}).$$

Par contre, il n'est pas du tout clair que P $\neq$ NP. La réponse à cette question est dotée[1] d'un prix de 1 million de dollars US. Une avancée importante vers la solution de ce problème a été accomplie dans les années 70 par Cook [16] et Levine [41] (voir [66] pour une traduction des travaux de Levine). Ces deux chercheurs ont en effet montré qu'il existe de problèmes de classe NP dont la complexité individuelle est reliée à la complexité de toute la classe. Plus précisément,

Définition 7.3.1. Soit $L \in$ NP. Si

$$L \in \text{P} \Leftrightarrow \text{P} = \text{NP},$$

alors L est appelé **NP-complet**.

Autrement dit, il existe de problèmes de classe NP tels que si on arrivait à démontrer qu'ils ne sont en fait que de classe P alors toute la classe NP serait réduite à la classe P. Pour donner un sens précis à cette définition, nous avons besoin de la notion de réductibilité temporelle d'un langage dans un autre.

Définition 7.3.2. Une application $f : \mathbb{A}^* \to \mathbb{A}^*$ est dite **calculable en temps polynomial** s'il existe une machine de Turing déterministe M qui pour tout $\alpha \in \mathbb{A}^*$ s'arrête en temps polynomial en $|\alpha|$ et au moment de l'arrêt de la machine, le mot (amputé de la sous-suite maximale des blancs) qui se trouve sur le ruban de la machine est $f(\alpha) = \mathsf{Tur}(\alpha)$.

Définition 7.3.3. Un langage $L \subset \mathbb{A}^*$ est **réductible en temps polynomial** à un langage $L' \in \mathbb{A}^*$ — on écrit $L \leq_{\mathsf{poly}} L'$ — s'il existe une application $f : \mathbb{A}^* \to \mathbb{A}^*$ calculable en temps polynomial telle que

$$\alpha \in L \Leftrightarrow f(\alpha) \in L'.$$

Nous donnons maintenant une nouvelle définition de la NP-complétude.

Définition 7.3.4. Un langage L est **NP-complet** si

1. $L \in$ NP,
2. $L' \in \text{NP} \Rightarrow L' \leq_{\mathsf{poly}} L$

Remarque 7.3.5. La signification de cette définition est que si L est un langage NP-complet, alors tout langage L' de NP est plolynomialement réductible en L. En d'autres termes, le langage L est représentatif de tous les langages de NP.

Nous arrivons maintenant à la formulation du résultat de Cook et Levine. On appelle **formule booléenne** à k entrées, une application $b : \{0,1\}^k \to \{0,1\}$ exprimable en termes de k variables reliées par des opérations $\wedge, \vee, \neg$. Par exemple l'application $b : \{0,1\}^3 \to \{0,1\}$ définie par $b(x_1, x_2, x_3) = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x}_3)$ est booléenne. Sur le triplet $(0,1,0)$ elle prend la valeur $b(0,1,0) = (0 \vee 1) \wedge (0 \vee \neg 0) = 1$.

1. Voir le site officiel du Clay Mathematical Institute qui a lancé le défi.

Définition 7.3.6. Une formule booléenne est dite ***k*-satisfaisable** si elle est à k variables et s'il existe $(x_1, \dots, x_k) \in \{0,1\}^k$ tel que $b(x_1, \dots, x_k) = 1$. Une formule booléenne est dite **satisfaisable** s'il existe un k pour lequel elle est k-satisfaisable. On note

$$\Lambda_{\text{SAT}} = \{ \langle\, b \,\rangle : b \text{ formule booléenne satisfaisable } \}$$

le langage constitué de formules booléennes satisfaisables.

Théorème 7.3.7 (Théorème de Cook et Levine)**.**

$$\Lambda_{\text{SAT}} \in \text{P} \Leftrightarrow \text{P} = \text{NP},$$

autrement dit le langage Λ_{SAT} *est* NP*-complet.*

7.4 Machines de Turing probabilistes, classe de complexité BPP

Une extension naturelle de la notion de machine de Turing non-déterministe est celle de machine de Turing probabiliste. Pour expliquer la notion, il est plus aisé de commencer par étendre un automate fini non-déterministe en automate probabiliste. Nous avons vu qu'un automate non-déterministe est caractérisé par des fonctions de transition $\delta_a : \mathbb{X} \to \mathcal{P}(\mathbb{X})$ qui peuvent être décrites de manière équivalente comme des matrices D_a à valeurs $\{0,1\}$ qui représentent des relations sur $\mathbb{X}$. Dans l'exemple 5.2.1, nous avions une relation représentée par la matrice $D_1 := \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$. Supposons maintenant que la première ligne de cette matrice (ayant deux termes égaux à 1) soit remplacée par un vecteur de probabilité $p, 1-p, 0, 0$, en gardant les autres lignes identiques. Cette modification transforme la matrice D_1 en matrice sous-stochastique ; l'évolution des trajectoires calculatoires, dont la combinatoire est déterminée par les itérations des matrices D_a, devient maintenant une évolution sous-stochastique, i.e. une chaîne de Markov [2] avec états absorbants. Par conséquent, toute trajectoire calculatoire acquiert maintenant une probabilité.

Cette construction s'étend naturellement à toute machine de Turing non-déterministe. On note $\mathbb{P}$ la probabilité induite sur les trajectoires calculatoires de la machine et on parle alors de **machine de Turing probabiliste** ou **algorithme Monte Carlo**. Un mot α sera alors accepté par M avec une certaine probabilité qui sera la somme sur les probabilités de toutes les trajectoires calculatoires déterminées à partir de α qui finiront dans l'état d'acceptation de la machine. Pour une machine probabiliste, nous n'aurons donc pas un résultat déterministe mais juste une probabilité d'acceptation ou de

2. Cf. cours *Probabilités pour la théorie de l'information*.

rejet de chaque mot. On note MTP la classe des machines de Turing probabilistes.

Définition 7.4.1. — Soient $M \in \text{MTP}$ et $\varepsilon \in [0, 1/2]$. On dit que M reconnaît le langage L avec probabilité d'erreur majorée par ε si
— $\alpha \in L \Rightarrow \mathbb{P}(M \text{ accepte } \alpha) \geq 1 - \varepsilon$,
— $\alpha \notin L \Rightarrow \mathbb{P}(M \text{ rejette } \alpha) \geq 1 - \varepsilon$
— La classe de langages reconnaissables par une machine de Turing probabiliste en temps polynomial avec une probabilité d'erreur $\varepsilon \leq 1/3$ est notée BPP.

Remarque 7.4.2. La valeur $1/3$ majorant la probabilité d'erreur dans la définition ci-dessus n'a rien de particulier. En vertu de la borne de Chernoff 7.4.3, cette valeur peut-être amplifiée (cf. exercice 61 ci-dessous) vers une erreur $\varepsilon \in]0, 1/2[$, arbitrairement proche de 0, tout en gardant la complexité temporelle polynomiale.

Lemme 7.4.3 (Borne de Chernoff). *Soit $(X_i)_{i\in\mathbb{N}}$ une suite de variables aléatoires indépendantes, à valeurs dans $\{0, 1\}$ et de même loi $\mathbb{P}(X_1 = 1) = p \in]0, 1[$. Alors pour tout $a \in]p, 1[$ et tout $n \geq 1$, on a*

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n} X_i \geq a\right) \leq \exp(nh(a, p)),$$

où $h(a, p) := -a \log \frac{a}{p} - (1 - a) \log \frac{1-a}{1-p}$.

Démonstration. Ce lemme est posé comme exercice dans le module *Probabilités pour la théorie de l'information*. □

Nous verrons l'année prochaine (Cours de cryptographie quantique) que la classe de machines de Turing probabilistes peut encore être élargie vers des machines de Turing quantiques qui sont capables de reconnaître en temps polynomial une classe plus vaste de langages, appelée BQP. Nous avons les hiérarchies

$$\text{P} \subseteq \text{NP} \quad \text{et} \quad \text{P} \subseteq \text{BPP} \subseteq \text{BQP},$$

mais nous ne connaissons ni si NP et BPP se comparent ni s'il existe des langages BPP-complets. Cependant, nous pouvons montrer que $\text{P} = \text{NP} \Rightarrow \text{P} = \text{BPP}$.

On peut aussi remarquer qu'il existe d'autres classes de complexité relatives à des machines probabilistes dont nous résumons les caractéristiques ci-dessous :

	BPP=CO-BPP	RP	CO-RP	ZPP
$\alpha \in L, \mathbb{P}(M \text{ acc } \alpha)$	$\geq 2/3$	$\geq 1/2$	$= 1$	$= 1$
$\alpha \notin L, \mathbb{P}(M \text{ rej } \alpha)$	$\geq 2/3$	$= 1$	$\geq 1/2$	$= 1$

Pour la classe ZPP, ce qui reste aléatoire est le temps d'arrêt. Nous exigeons que pour tout $\alpha \in \mathbb{A}^*$, on a $\mathbb{E}(\tau(\alpha)) < \infty$.

7.5 Exercices

Classes P et NP

56. Proposer un algorithme (sous forme de pseudo-code) de complexité polynomiale pour décider le problème

$$\Lambda_{\text{PATH}} = \{\langle \mathbb{G}, x, y\rangle : \exists \alpha \in \mathbb{G}^*, s(\alpha) = x, t(\alpha) = y\}$$

où $\mathbb{G} = (\mathbb{G}^0, \mathbb{G}^1)$ est un graphe dirigé et $x, y \in \mathbb{G}^0$.

57. Proposer un algorithme pour résoudre le problème de décidabilité du problème Λ_{PATH} sur une machine bi-ruban qui s'exécute en temps $\mathcal{O}(n)$.

58. Donner un pseudo-code qui décrit l'algorithme d'Euclide pour le calcul de pgcd de deux entiers. Utiliser ensuite cet algorithme comme un sous programme pour montrer que

$$\Lambda_{\text{COPRIME}}\{\langle\, x, y\,\rangle : \mathsf{pgcd}(x, y) = 1\}$$

est dans la classe P.

59. Donner le pseudo-code d'un algorithme qui vérifie si une trajectoire sur un graphe dirigée est un chemin sans recoupement couvrant, en temps polynomial en le nombre des sommets du graphe.

60. On a en caisse $n \geq 1$ pièces de monnaie ayant de valeurs nominales (pas nécessairement distinctes) $x_1, \ldots, x_n \in \mathbb{N}_>$ (on note $[x_1, \ldots, x_n]$ la liste, éventuellement avec des répétitions, des valeurs nominales (exprimées en centimes d'€) des pièces en caisse. Soit $r \in \mathbb{N}_>$ la somme (exprimée en centimes d'€) que nous devons rendre au client. On veut savoir s'il est possible de rendre la monnaie au client avec les pièces en caisse. On peut reformuler ce problème comme un problème de décision d'appartenance au langage

$$\Lambda_{\text{SUBSET-SUM}} = \{\langle\, \alpha, \beta\,\rangle : \alpha = [x_1, \ldots, x_n] \wedge \exists J \subseteq \{1, \ldots, n\},\ \text{t. q. } \sum_{j\in J} x_j = \mathsf{num}(\beta)\}.$$

— Montrer qu'il existe une machine vérificatrice V qui, à l'entrée $\langle\,\langle\, \alpha, \beta\,\rangle, \gamma\,\rangle$, où $\langle\, \alpha, \beta\,\rangle$ comme ci-dessus et γ une collection d'entiers dont la somme vaut $\mathsf{num}(\beta)$, décide s'il existe une solution.
— Montrer qu'il existe une machine $N \in \text{MTN}$ qui décide $\Lambda_{\text{SUBSET-SUM}}$.
— Conclure que $\Lambda_{\text{SUBSET-SUM}} \in \text{NP}$

Classe BPP

61. Montrer que si $M \in \text{MTP}$ reconnaît un langage L avec une probabilité d'erreur majorée par $1/3$, alors en répétant l'expérience de reconnaissance un grand nombre n de fois, on peut rendre l'erreur ε aussi petit que l'on veuille. Estimer n pour que $\varepsilon \leq 10^{-2}$.

8

Stochasticité, complexité de Kolmogorov, entropie

> Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin. For, as has been pointed out several times, there is no such thing as a random number — there are only methods to produce random numbers, and a strict arithmetic procedure of course is not such a method.
>
> John von Neumann, *Various techniques used in connection with random digits* (1951) [73].

Au chapitre précédent, nous avons vu que la classe de langages reconnus par des machines de Turing probabilistes est *a priori* plus riche que la classe P. Il est donc important de savoir comment écrire des algorithmes Monte Carlo (comment générer des variables aléatoires de loi donnée) car cela permet de résoudre de manière probabiliste des problèmes de manière algorithmiquement beaucoup plus efficace que des méthodes déterministes ou non-déterministes i.e. des méthodes où le cas le moins favorable peut entraîner une dégradation des performances d'un programme. Par ailleurs, la génération de suites de variables aléatoires joue un rôle important dans la construction efficace de clés de chiffrement [70, 52, 50]. Nous sommes donc amenés à poser la

Question fondamentale : Comment générer une suite indépendante honnête de « pile ou face » ?

Sous son apparence anodine, il s'agit de la question la plus profonde que l'on peut poser en théorie des probabilités. Le théorème de Kolmogorov nous enseigne qu'*il existe* un espace abstrait $(\Omega, \mathcal{F}, \mathbb{P})$ et une suite d'applications $X_n : \Omega \to \{0,1\}$ telle que $\mathbb{P}(X_n = 1) = 1/2$ pour tout $n \in \mathbb{N}$ et les X_n sont indépendantes. Ensuite, une fois que nous disposons de la

suite X_n, nous savons comment générer une variable aléatoire sur $[0,1]$ de loi arbitraire (voir exercice 2.3.12 du cours de *Probabilités pour la théorie de l'information*). Mais cette réponse ne fait que déplacer le problème : comment simuler $(\Omega, \mathcal{F}, \mathbb{P})$? Il n'est pas surprenant donc que la personne qui a le plus chercher à donner une réponse convaincante à ce qu'est l'aléa était celle-la même qui a axiomatisé la théorie moderne des probabilités, à savoir Kolmogorov [37].

Dans la suite, une brève présentation des divers aspects de l'aléa est donnée, basée essentiellement sur l'article de revue [69] et le chapitre 2 du livre [42]. Le chapitre se termine par une conclusion sur l'impossibilité de créer un aléa par des méthodes algorithmiques ou physiques classiques qui — même si elle transparaît de manière sous-jacente dans plusieurs textes publiés dans la littérature — présente la vision personnelle de l'auteur sur la question.

8.1 Stochasticité, chaoticité, typicité : trois aspects de l'aléa

On limite l'étude des trois aspects de l'aléa sur les suites aléatoires à valeurs dans $\mathbb{A} = \{0,1\}$. Nous aurons besoin des ensembles suivants des suites binaires

$$\mathbb{A}^* = \cup_{n\in\mathbb{N}}\mathbb{A}^n, \Omega := \partial\mathbb{A}^* = \mathbb{A}^{\mathbb{N}} \text{ et } \overline{\mathbb{A}^*} = \Omega \cup \mathbb{A}^*.$$

Pour un mot infini $\omega \in \Omega$, on note ω_k sa k^e lettre et pour $0 \leq k \leq l$, $\omega_{[k:l]} = \omega_k \cdots \omega_l$ le sous-mot de longueur $l-k+1$ entre la k^e et la l^e lettres. Si $M = \{m_0, \ldots, m_N\} \subseteq \mathbb{N}$ est une partie (éventuellement infinie, i.e. $N = \infty$) de $\mathbb{N}$, on note ω_M la suite extraite $\omega_{m_0} \cdots \omega_{m_N}$.

Pour qu'une suite $\alpha \in \overline{\mathbb{A}^*}$ corresponde à notre intuition de l'aléatoire, elle doit être

- — stochastique, dans le sens qu'elle vérifie certaines propriétés de stabilité fréquentielle,
- — chaotique, dans le sens qu'elle soit désordonnée avec une entropie de Kolmogorov (mesure de l'information contenue) proportionnelle à sa longueur et
- — typique, dans le sens que les suites non-typiques sont dans un ensemble effectivement négligeable.

8.1.1 Stochasticité

Les premiers débats et travaux mathématiques sur la nature de l'aléa sont dus à von Mises qui propose la notion de stabilité fréquentielle [72] comme caractérisation primitive de l'aléa. Aujourd'hui, cette approche est inversée dans le sens que la stabilité fréquentielle est une conséquence du théorème des grands nombres de la théorie des probabilités, non une propriété primitive définissant la notion de probabilité.

Si $\omega = \omega_0\omega_1\omega_2\cdots \in \Omega$ est une suite binaire indépendante infinie distribuée selon la loi μ sur $\mathbb{A}$, la fréquence empirique d'apparition de chaque bit tend vers $\mu(b)$, i.e. pour tout $b \in \mathbb{A}$,

$$\lim_{n\to\infty} \frac{\nu_b^{(n)}(\omega)}{n} = \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} \mathbb{1}_{\{b\}}(\omega_k) = \mu(b).$$

On peut objecter que, pour $\mathbb{A} = \{0,1\}$ et $\mu = (\frac{1}{2}, \frac{1}{2})$, la suite

$$\beta = 010101010101\cdots$$

vérifie cette propriété mais n'a rien d'aléatoire. On pallie cet inconvénient en exigeant que non seulement la suite mais aussi des sous-suites vérifient la propriété de stabilité fréquentielle. Et dans ce cas, la suite précédente ne vérifie pas la propriété de stabilité fréquentielle pour les sous-suites car la suite extraite de termes pairs est la suite $111\cdots$. Cependant, on ne peut pas exiger la propriété de stabilité fréquentielle pour toutes les sous-suites. Pour s'en convaincre, considérons une suite infinie arbitraire $\xi = (x_0, x_1, x_2, \cdots)$ et construisons la suite d'entiers définie par

$$n_0 = \inf\{n \geq 0 : x_n = 0\}$$

et récursivement, tant que n_{m-1} soit fini,

$$n_m = \inf\{n > n_{m-1} : x_n = 0\}.$$

Alors, dans le cas où $\inf\{m : n_m = +\infty\} = +\infty$, la sous-suite $(x_{n_0}, x_{n_1}, x_{n_2}, \cdots)$ ne vérifie pas — par construction — la propriété de stabilité fréquentielle. Il est donc évident que cette propriété doit être vraie uniquement pour certaines sous-suites qui remplissent des conditions particulières.

Définition 8.1.1. Soient $\gamma \in \mathbb{A}^*$ un mot binaire fini arbitraire de longueur $l = |\gamma|$ et $\omega \in \Omega$ une suite binaire infinie. On note

$$i_0 = i_0(\omega, \gamma) := \inf\{m \geq l : \omega_{[m-l:m]} = \gamma\} + 1$$
$$i_k = i_k(\omega, \gamma) := \inf\{m \geq i_{k-1} : \omega_{[m-l:m]} = \gamma\} + 1 \text{ pour } k > 0 \text{ si } i_{k-1} < \infty.$$

La sous-suite ω_M avec $M := M(\omega, \gamma) = \cup_{k\in\mathbb{N}}\{i_k\}$ sera la sous-suite γ-**légale** de α. Les suites **légales** seront les sous-suites obtenues de cette façon pour $\gamma \in \mathbb{A}^*$.

Définition 8.1.2. Une suite est **stochastique** si toutes ses sous-suites légales vérifient la propriété de stabilité fréquentielle.

Proposition 8.1.3. *Soit $\gamma = \gamma_1\cdots\gamma_k$ un mot fini arbitraire tel que $|\gamma| = k$ et $\xi \in \Omega$ une suite stochastique générée avec la loi de Bernoulli de paramètre 1/2. Alors*

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n} \mathbb{1}_{\{\gamma\}}(\xi_{[i:i+k-1]}) = \frac{1}{2^k}.$$

Démonstration. Si $|\gamma| = 1$, alors le résultat est une conséquence du résultat de stabilité fréquentielle pour les suites infinies. Pour $|\gamma| > 1$, on procède par récurrence. Supposons que la formule soit vraie pour un $\gamma \in \mathbb{A}^k$, $k \geq 1$, i.e. $\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n} \mathbb{1}_{\{\gamma\}}(\xi_{[i:i+k-1]}) = \frac{1}{2^k}$. Chaque mot γ dans ξ est suivi soit d'un 0 soit d'un 1 et la suite des successeurs de γ est une sous-suite légale de ξ. Elle vérifie donc la propriété de stabilité fréquentielle. Par conséquent,

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n} \mathbb{1}_{\{\gamma 0\}}(\xi_{[i:i+k]}) = \frac{1}{2^{k+1}} = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n} \mathbb{1}_{\{\gamma 1\}}(\xi_{[i:i+k]}).$$

□

La proposition 8.1.3 signifie
— que tout mot fini doit apparaître une infinité de fois dans une suite stochastique infinie
— et, en particulier, qu'une suite stochastique est une suite normale au sens de Borel [6].

Noter aussi que cette limite peut être obtenue (presque sûrement) comme une conséquence de la loi forte des grands nombres. Cependant, ici elle est obtenue pour *toutes* les suite dites stochastiques. C'est comme si les suites dites stochastiques constituaient l'ensemble Ω privé de l'ensemble des suites exceptionnelles (de mesure nulle) ignorées par le « presque sûrement » de loi forte des grands nombres.

8.1.2 Chaoticité et complexité de Kolmogorov

Intuitivement, une suite de 1000000 bits 0 est plus simple à décrire qu'une suite de 1000000 occurrence d'un jeu de pile ou face honnête car la première est totalement décrite par la phrase très courte « 1000000 bits 0 » tandis que la seconde n'est décrite que par la donnée explicite des un million de bits qui la composent. Kolmogorov [37] définit le caractère aléatoire d'une suite comme la difficulté intrinsèque de la décrire par une méthode plus courte. On a vu qu'une fonction $f : \mathbb{A}^* \to \mathbb{A}^*$ est calculable s'il existe $M \in \text{MTD}$ telle que $f = \text{Tur}_M$.

Définition 8.1.4. Supposons que $\hat{f} : \mathbb{A}^* \to \mathbb{A}^*$ est une fonction calculable totale (i.e. $\text{dom}(\hat{f}) = \mathbb{A}^*$). Elle peut être étendue en une fonction totale $f : \overline{\mathbb{A}^*} \to \overline{\mathbb{A}^*}$. L'application (éventuellement étendue) f est **calculable** lorsque pour des mots $\alpha, \beta \in \overline{\mathbb{A}^*}$, on a :

1. $\alpha \preceq \beta \Rightarrow f(\alpha) \preceq f(\beta)$ (si α est un préfixe de β alors $f(\alpha)$ est un préfixe de $f(\beta)$),
2. si $|\beta| = \infty$, alors $f(\beta)$ est le mot minimal continuation de *toutes* les suites $f(\beta|_n)$ pour $n = 1, 2, \cdots$,
3. l'ensemble de paires $F = F_f := \{(\alpha, \beta) \in \mathbb{A}^* \times \mathbb{A}^* : \beta \preceq f(\alpha)\}$ est énumérable (ou Turing-decidable) [1].

1. Un ensemble est dit énumérable s'il existe une machine de Turing capable d'énumérer tous ses éléments, i.e. un algorithme capable d'imprimer les éléments de l'ensemble.

Cette définition implique que toute application calculable peut être décrite à l'aide d'un algorithme *fini*. Intuitivement, f est un algorithme. Lorsqu'on tape la suite α_0, α_1, ... au clavier d'un ordinateur qui exécute le programme f, on obtient séquentiellement

$$\begin{aligned} \alpha_0 &\mapsto f(\alpha_0) \\ \alpha_0\alpha_1 &\mapsto f(\alpha_0\alpha_1) \\ \alpha_0\alpha_1\alpha_2 &\mapsto f(\alpha_0\alpha_1\alpha_2) \\ &\vdots \end{aligned}$$

En d'autres termes, si $\beta = f(\alpha)$, alors il existe une machine de Turing $M \in \text{MT}$ telle que $\beta = \text{Tur}_M(\alpha)$.

Remarque 8.1.5. L'application totale f et l'ensemble énumérable $F = F_f$ dans la définition 8.1.4 sont conjugués. Il existe en effet une *bijection* entre les applications totales $f : \overline{\mathbb{A}^*} \to \overline{\mathbb{A}^*}$ et les parties énumérables $F \subseteq \mathbb{A}^* \times \mathbb{A}^*$, car pour tout F énumérable, nous pouvons définir une application totale $f := f_F$ par

$$f(\alpha) = \sup\{\gamma \in \mathbb{A}^* : \exists\beta[\beta \preceq \alpha \wedge (\beta, \gamma) \in F]\},$$

où le sup est pour l'ordre défini par la relation $\preceq$.

Définition 8.1.6. Pour $\alpha, \beta \in \mathbb{A}^*$ et f une fonction calculable,

— on dit que la suite β est une **description** de α (relativement à f) si $\alpha \preceq f(\beta)$,

— la **complexité (de Kolmogorov)** du mot α (relativement à f) est la quantité

$$K_f(\alpha) = \inf\{|\beta| : \beta \text{ est une descritpion de } \alpha \text{ relativement à } f\}.$$

— Une application calculable f est dite **optimale** si pour toute application calculable g, il existe une constante C telle que

$$K_g(\alpha) \leq K_f(\alpha) + C,$$

uniformément en α.

En d'autres termes, décrire la suite finie α consiste à trouver une suite finie β, telle que la suite (finie ou infinie) $f(\beta)$ commence par α, c'est-à-dire $f(\beta) = \alpha\gamma$, pour un certain mot γ.

Théorème 8.1.7 (Kolmogorov [37])**.** *L'ensemble des applications optimales est non vide.*

Définition 8.1.8. On appelle **entropie (de Kolmogorov)** d'une suite sa complexité relativement à une application optimale.

Remarque 8.1.9. On voit que la notion d'entropie de Kolmogorov dépend de l'application optimale choisie. Il existe cependant une constante C_2 telle que si $K_1(\cdot)$ et $K_2(\cdot)$ sont des entropies relatives à deux applications optimales différentes, la différence

$$|K_1(\alpha) - K_2(\alpha)| \leq C_2$$

est uniformément bornée. On note alors $K(\alpha)$ l'entropie à une constante près.

Si on choisit comme fonction calculable la fonction identité, $f = \mathrm{id}$, on observe que $K_f(\alpha) = |\alpha|$. On a donc la borne évidente $K(\alpha) \leq |\alpha| + \mathcal{O}(1)$ ce qui montre que l'entropie d'une suite ne peut pas excéder sa longueur que d'une constante. Les suites où l'inégalité précédente dévient une égalité sont les suites chaotiques. Le théorème 8.1.7 permet en outre de définir la complexité (à une constante près) d'une suite de manière universelle :

Définition 8.1.10. Soit β une suite de bits. On définit la **complexité algorithmique** de β par

$$K(\beta) = \inf\{|\langle M, \alpha \rangle| : \alpha \in \{0,1\}^*, M \in \mathrm{MT} \text{ et } \mathsf{Tur}(\alpha) = \beta\}.$$

Définition 8.1.11. Une suite infinie $\omega \in \Omega$ est dite **chaotique** s'il existe une constante C telle que

$$|K(\omega|_n) - n| \leq C$$

pour tout $n \in \mathbf{N}$.

Ainsi, les suites chaotiques sont les suites dont la description nécessite un algorithme aussi long que la suite elle même, i.e. $K(\beta) = \mathcal{O}(|\beta|)$ Les suites aléatoires doivent être chaotiques.

Nous rappelons (cf. cours *Probabilités pour la théorie de l'information*) qu'une suite aléatoire $\mathbf{X} = (X_n)_{n=1,\dots,N}$ i.i.d. à valeurs dans $\{0,1\}$ vérifiant $\mathbb{P}(X_1 = 1) = p \in]0,1[$ a une entropie[2] $H(\mathbf{X}) = NH(p)$, où $H(p) = -p\log_2 p - (1-p)\log_2(1-p) > 0$. Si $p = 1/2$ alors l'entropie prend sa valeur maximale $H(1/2) = 1$. On constate que dans ce cas,

$$|K(\alpha|_n) - n| = |K(\alpha|_n) - H(\mathbf{X}|_n)| \leq C.$$

Une suite infinie uniformément distribuée sur $\{0,1\}$ est chaotique si son **entropie spécifique**[3] est maximale (i.e. 1).

On peut consulter [62] pour une étude plus complète de cette affirmation.

2. Mesurée en bits.

3. L'**entropie spécifique** d'une suite infinie $\mathbf{X} = (X_n)_{n\in\mathbb{N}}$ de variables aléatoires indépendantes et identiquement distribuées est la limite $\lim_{N\to\infty} \frac{H(\mathbf{X}\restriction_N)}{N}$. Cette limite existe en vertu de la propriété de subadditivité de la fonction H (cf. cours *Théorie des probabilités pour la théorie de l'information*).

Il n'existe pas de générateur classique de vrais nombres aléatoires

La notion de chaoticité (cf. définition 8.1.11) exigée pour toute suite aléatoire a une implication majeure : il n'existe pas de suite aléatoire générée par un algorithme classique.

Exemple 8.1.12. Le générateur algorithmique de nombres (pseudo)-aléatoires (sur 32 ou 64 bits) utilisé dans la plupart de suites statistiques de langages de programmation et de logiciels [4] est l'agorithme de torsion de Mersenne, introduit dans [47]. Son nom est dû à sa période (très longue) qui est égale au premier de Mersenne $2^{19937} - 1$. Son pseudo-code tient sur une vingtaine de lignes et son initialisation se fait par quelques centaines de mots de 32 ou 64 bits. Toute la suite des $2^{19937} - 1$ bits générés par l'algorithme est donc décrite (au sens de Kolmogorov) par quelques milliers de bits.

8.1.3 Typicité et ensembles effectivement négligeables

Une autre caractéristique qu'une suite aléatoire indépendante doit posséder est qu'elle doive satisfaire le théorème fort de grands nombres, i.e. que la fréquence empirique d'apparition du bit $b \in \mathbb{A}$ dans une suite indépendante et identiquement distribuée selon une loi μ tende presque sûrement vers $\mu(b)$:

$$\mathbb{P}(\{\alpha \in \Omega : \lim_{n\to\infty} \frac{\nu_b^{(n)}(\alpha)}{n} \neq \mu(b)\}) = 0.$$

La notion de suite typique vise à rendre la notion de convergence presque sûre effectivement vraie, i.e. *construire* un ensemble de suites effectivement négligeable à l'extérieur duquel tous les théorèmes de probabilité sont valables.

On rappelle que nous avons noté $\Omega = \partial\mathbb{A}^* = \mathbb{A}^{\mathbb{N}}$. Pour tout $\gamma \in \mathbb{A}^*$, on note

$$\Omega_\gamma := \{\omega \in \Omega : \gamma \preceq \omega\} = \{\gamma\omega : \omega \in \Omega\},$$

i.e. les suites infinies qui commencent par γ. Comme Ω se surjecte sur $[0,1]$, il est immédiat que Ω_γ correspond à un sous-intervalle $I_\gamma \subseteq [0,1]$ de longueur $|I_\gamma| = \frac{1}{2^{|\gamma|}}$.

Définition 8.1.13. Un ensemble $A \subseteq \Omega$ est **négligeable** si pour tout $\eta > 0$, il existe une suite $(\gamma^{(n)})_{n\in\mathbb{N}}$ de mots dans $\mathbb{A}^*$ telle qe

1. $A \subseteq \cup_{n\in\mathbb{N}} \Omega_{\gamma^{(n)}}$ et
2. $\sum_{n\in\mathbb{N}} 2^{-|\gamma^{(n)}|} \leq \eta$.

4. Microsoft Visual C++, Microsoft Excel, GAUSS, GLib, GNU Multiple Precision Arithmetic Library, GNU Octave, GNU Scientific Library, gretl, IDL, Julia, CMU Common Lisp, Embeddable Common Lisp, Steel Bank Common Lisp, Maple, MATLAB, Free Pascal, PHP, Python, R, Ruby, SageMath, Scilab, Stata.

Cette définition d'ensemble négligeable coïncide donc avec la définition standard d'ensemble Lebesgue-négligeable sur $[0,1]$ car A est négligeable s'il peut être recouvert par une suite d'intervalles de mesure totale inférieure à η. On peut par ailleurs généraliser cette définition à une probabilité arbitraire μ (σ-additive sur les boréliens de $[0,1]$). Une telle probabilité peut être reconstruite à partir de ses valeurs $(\mu(\Omega_\gamma))_{\gamma\in\mathbb{A}^*}$.

Proposition 8.1.14. *Soit $m : \mathbb{A}^* \to [0,1]$ vérifiant les* ***conditions de compatibilité de Kolmogorov***

$$m(\gamma) = \begin{cases} 1 & si\ \gamma = \varepsilon \\ m(\gamma 0) + m(\gamma 1) & \forall \gamma \in \mathbb{A}^*. \end{cases}$$

Alors, il existe une (unique) mesure de probabilité μ sur Ω définie par $\mu(\Omega_\gamma) = m(\gamma)$ pour tout $\gamma \in \mathbb{A}^*$.

La probabilité μ est alors en bijection avec la fonction m.

Définition 8.1.15. Une mesure de probabilité μ sur Ω est **calculable** s'il existe une application calculable $M : \mathbb{A}^* \times \mathbb{Q}_> \to \mathbb{Q} \cap [0,1]$ telle que

$$|M(\gamma,\delta) - \mu(\Omega_\gamma)| \leq \delta,$$

pour tout $\delta > 0$ rationnel et toute suite finie γ.

Définition 8.1.16. Soit μ une probabilité sur Ω en bijection avec une fonction $m : \mathbb{A}^* \to [0,1]$ vérifiant les conditions de compatibilité de Kolmogorov. Un ensemble $N \subset \Omega$ est

— **négligeable** si pour tout $\delta > 0$, il existe une famille $\mathcal{I}$ dénombrable (finie ou infinie) de suites $(\gamma^{(i)})_{i\in\mathcal{I}}$, avec $\gamma^{(i)} \in \mathbb{A}^*$ pour tout $i \in \mathcal{I}$, telle que

$$N \subset \bigcup_{i\in\mathcal{I}} \Omega_{\gamma^{(i)}} \text{ avec } \sum_{i\in\mathcal{I}} m(\gamma^{(i)}) < \delta,$$

— **effectivement négligeable** s'il existe une application calculable $\xi : \mathbb{Q}_> \times \mathbb{N} \to \mathbb{A}^*$ telle que pour tout $\delta > 0$ rationnel et tout entier naturel i, on a

$$N \subset \bigcup_{i\in\mathbb{N}} \Omega_{\xi(\delta,i)} \text{ et } \sum_{i\in\mathbb{N}} \mu(\Omega_{\xi(\delta,i)}) < \delta.$$

Un ensemble A sera alors μ-négligeable s'il existe une suite $(\gamma^{(n)})_{n\in\mathbb{N}}$ de mots de $\mathbb{A}^*$ telle que $(\Omega_{\gamma^{(n)}})_{n\in\mathbb{N}}$ est un recouvrement de A et $\sum_{n\in\mathbb{N}} \mu(\Omega_{\gamma^{(n)}}) = \sum_{n\in\mathbb{N}} m(\gamma^{(n)}) \leq \delta$. Il sera effectivement négligeable si la suite $(\gamma^{(n)})_{n\in\mathbb{N}}$ est calculable et le programme qui la calcule pour tout $\delta > 0$ est effectivement constructible. Ceci impose que le nombre $\delta > 0$ soit rationnel car c'est uniquement sous cette condition que l'on pourra concevoir une machine de Turing à entrée *finie* $\langle\, \mathsf{rep}(\delta), \mathsf{rep}(n) \,\rangle$, (avec $\delta = \mathsf{num}_2(\langle\, \delta \,\rangle)$ et $n = \mathsf{num}_2(\langle\, n \,\rangle)$, qui produit en un nombre fini d'étapes le mot $\xi_{\delta,n} = \mathsf{Tur}(\langle\, \mathsf{rep}(\delta), \mathsf{rep}(n) \,\rangle)$.

Remarque 8.1.17. On n'exige pas que ξ soit totale. Si ξ n'est pas définie pour certain couples, on pose $\Omega_{\xi(\delta,n)} = \emptyset$ et $|\xi(\delta, n)| = \infty$.

Nous considérons dans la suite de ce paragraphe des mesures de probabilités μ sur Ω définies à partir de la mesure m de façon calculable.

Théorème 8.1.18 (Martin-Löf [45])**.** *Si la mesure de probabilité μ sur Ω est calculable, alors il existe un ensemble maximal (pour l'inclusion) effectivement négligeable.*

Remarque 8.1.19 (Formulation équivalente du théorème 8.1.18)**.** Il existe un ensemble universel effectivement négligeable qui inclut tout ensemble effectivement négligeable.

Définition 8.1.20. Une suite est **typique** par rapport à une mesure de probabilité calculable si elle n'est pas contenue dans l'ensemble effectivement négligeable maximal.

Proposition 8.1.21. *Une suite calculable $\omega \in \Omega$ est typique par rapport à une probabilité μ, si, et seulement si, $\mu(\{\omega\}) > 0$.*

Démonstration. Si $\mu(\{\omega\}) > 0$, alors ω ne peut appartenir à aucun négligeable, donc *a fortiori* dans aucun effectivement négligeable.

Réciproquement, si $\mu(\{\omega\}) = 0$, nous allons montrer que $\{\omega\}$ est effectivement négligeable. Soit $\omega\restriction_n$ la suite des préfixes de ω. Nous avons alors $\cap_{n\in\mathbb{N}}\Omega_{\omega\restriction_n} = \{\omega\}$ et la suite des ensembles $\Omega_{\omega\restriction_n}$ est décroissante. Par continuité monotone de la probabilité, on a donc $\mu(\Omega_{\omega\restriction_n}) \to 0$. Puisque tant ω que μ sont calculables, cette convergence est effective, i.e. pour tout $\delta > 0$ rationnel, nous pouvons construire algorithmiquement un segment initial $\xi_\delta \preceq \omega$ t.q. $\mu(\xi_\delta) < \delta$. Or Ω_{ξ_δ} est un recouvrement de $\{\omega\}$. On conclut que $\{\omega\}$ est effectivement négligeable. □

Théorème 8.1.22 (Levin [40] et Schnorr [59])**.** *Une suite est typique si, et seulement si, elle est chaotique.*

En résumant, nous avons

$$\text{chaoticité} \iff \text{typicité} \implies \text{stochasticité}.$$

8.2 Réalisation de l'aléa par un procédé physique

Les paragraphes précédents ont établi l'impossibilité fondamentale de générer une suite de nombres aléatoires par un algorithme classique. La question subsiste cependant si on peut générer une telle suite par un procédé physique classique fini. Nous verrons dans le cours de *Fondements mathématiques de la mécanique quantique et applications* que la physique classique est équivalente à une théorie des probabilités enrichie d'une transformation dynamique. Il est donc intuitivement clair qu'il sera aussi impossible de générer une vraie suite aléatoire en lançant successivement une pièce honnête.

Le mouvement d'une pièce (vue comme un corps solide) est soumis aux lois du mouvement de la mécanique classique. Pour simplifier les calculs, on considère, comme dans [34], que la pièce a la forme d'un disque (sans épaisseur) de rayon R et son centre de masse coïncide avec le centre géométrique. Une impulsion initiale sur la pièce consiste à lui imputer une vitesse verticale initiale v_z et une vitesse angulaire α autour d'un axe de rotation sur le plan du disque et passant par son centre [5]. Dans la suite, la pièce évolue sous l'action de la pésenteur (dont l'accéleration est g) et sans action de force ou de moment. On note $z(t)$ la position verticale du centre de masse et $\theta(t)$ la angle (de la normale à la pièce par rapport à la verticale) à l'instant t.

Le mouvement est décrit par les équations différentielles :

$$\frac{\mathrm{d}^2 z}{\mathrm{d}t^2}(t) = -g, \text{ avec conditions initiales : } z(0) = R, \frac{\mathrm{d}z}{\mathrm{d}t}(0) = v_z,$$
$$\frac{\mathrm{d}^2 \theta}{\mathrm{d}t^2}(t) = 0, \text{ avec conditions initiales : } \theta(0) = 0, \frac{\mathrm{d}\theta}{\mathrm{d}t}(0) = \alpha.$$

Les solutions de ces équations s'obtiennent de manière élémentaire :

$$z(t) = v_z t - \frac{1}{2} g t^2 + R; \;\; \theta(t) = \alpha t$$

et sont valables de $t = 0$ jusqu'au temps $t_0 > 0$ le premier instant où la pièce touche une surface totalement plastique (par exemple du sable, de la boue ou de la pâte à modéler) sur le plan $z = 0$. L'instant t_0 se calcule en cherchant le plus petite racine positive de l'équation $z(t_0) - R|\sin\theta(t_0)| = 0$ et la pièce atterrira sur la « pile » si

$$2n\pi - \frac{\pi}{2} < \theta(t_0) 2 < n\pi + \frac{\pi}{2}, n \in \mathbb{N}.$$

On appelle pré-images de « pile » les couples $(v_z, \alpha) \in \mathbb{R}^2_>$ qui donnent « pile », i.e. $\alpha t_0 \in [(2n - \frac{1}{2})\pi, (2n + \frac{1}{2})\pi]$. Lorsque αt_0 est sur un des bords de ces intervalles, on a $\sin\theta(t_0) = \pm 1$ et l'équation pour le point inférieur de la pièce devient $z(t_0) - R = v_z t_0 - \frac{1}{2} g t_0^2 = 0$, ou $t_0 = \frac{2v_z}{g}$. En introduisant la variable $\zeta = \frac{v_z}{g}$ (qui a des dimension de temps), la famille d'équations

$$\alpha = (2n \pm \frac{1}{2})\frac{\pi}{2}\zeta, n \in \mathbb{N}_>$$

délimite alternativement dans le plan (α, ζ) les lieux des paramètres où la pièce atterrit sur « pile » ou « face ». La machine décrite dans la figure 8.1 est physiquement réalisée et son fonctionnement décrit dans [23]. Son utilisation conduit aux résultats décrits dans la figure 8.2.

La raison profonde pour laquelle la pièce lancée tombe tantôt du côté « pile » tantôt du côté « face » est que nos doigts ne peuvent pas contrôler

5. Il est démontré dans [23] que c'est uniquement sous cette condition que la pièce est approximativement honnête.

FIGURE 8.1 – Le « lanceur de pièces » de Diaconis et al. [23].

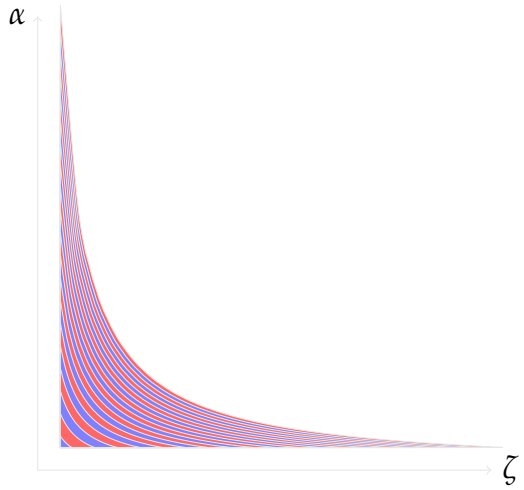


FIGURE 8.2 – L'espace de phases (α, ζ) (sous quelques conditions simplificatrices), où α est la vitesse angulaire initiale et $\zeta = \frac{v_z}{g}$, où v_z est la vitesse verticale initiale et g l'accélération de la pésanteur, est stratifié en régions rouges et bleues selon les résultats possibles (pile ou face). Seulement les premières strates sont présentées ; cependant, la stratification couvre tout le quadrant.

avec précision la condition initiale ; cette dernière contient des états chevauchant plusieurs strates de l'espace des phases (cf. figure 8.2). Le lanceur des pièces en contrôlant mieux la distribution de la condition initiale permet d'introduire une forte corrélation entre la face de lancement et d'atterrissage de la pièce. De point de vue fondamentale, rien n'interdit d'imaginer un appareil encore mieux contrôlé que le lanceur à ressort pour prédire de manière déterministe la suite.

En résumant les paragraphes 8.1 et 8.2, nous formulons le corollaire suivant.

Corollaire 8.2.1. *Il n'existe*
— *ni d'algorithme informatique classique (machine de Turing classique),*
— *ni de système physique classique fini*
permettant de jouer au « pile ou face » !

Ce corollaire ne siginifie pas qu'il n'y ait pas d'aléa dans la Nature ; il établit juste la **réductibilité de l'aléa classique**. Nous verrons au cours *Fondements mathématiques de la mécanique quantique et applications* que le seul vrai aléa dans la Nature est l'aléa quantique.

Bibliographie

[1] Peter B. Andrews. *An introduction to mathematical logic and type theory: to truth through proof*, volume 27 of *Applied Logic Series*. Kluwer Academic Publishers, Dordrecht, second edition, 2002. URL: http://dx.doi.org/10.1007/978-94-015-9934-4, doi:10.1007/978-94-015-9934-4. 47, 60

[2] A. Baker. Contributions to the theory of Diophantine equations. II. The Diophantine equation $y^2 = x^3 + k$. *Philos. Trans. Roy. Soc. London Ser. A*, 263:193–208, 1967/1968. 3

[3] Garrett Birkhoff. *Lattice theory*, volume 25 of *American Mathematical Society Colloquium Publications*. American Mathematical Society, Providence, R.I., third edition, 1979. 60

[4] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 2010. 4th edition. URL: https://doi.org/10.1017/CBO9781107050884, doi:10.1017/CBO9781107050884. 52

[5] George Boole. *An investigation of the laws of thought*. Cambridge Library Collection. Cambridge University Press, Cambridge, 2009. On which are founded the mathematical theories of logic and probabilities, Reprint of the 1854 original, published by Walton and Maberly, Cambbridge (1854). URL: https://doi.org/10.1017/CBO9780511693090.024, doi:10.1017/CBO9780511693090.024. 41

[6] Émile Borel. Les probabilités dénombrables et leurs applications arithmétiques. *Rend. Circ. Mat. Palermo*, 27:247–271, 1909. 110

[7] Georg Cantor. Über eine Eigenschaft des Inbegriffes aller reelle algebraische Zahlen. *Journal für die reine und angewandte Mathematik*, pages 258–262, 1874. 4

[8] Georg Cantor. Beiträge zur Begründung der transfiniten Mengenlehre. *Math. Ann.*, XLVI(2):481–512, 1897. doi:10.1007/BF01444205. 4

[9] Georg Cantor. Beiträge zur Begründung der transfiniten Mengenlehre. *Math. Ann.*, XLIX(2):207–246, 1897. URL: https://doi.org/10.1007/BF01444205, doi:10.1007/BF01444205. 4

[10] Brian F. Chellas. *Modal logic*. Cambridge University Press, Cambridge-New York, 1980. An introduction. 52

[11] Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2, 1956. doi:10.1109/tit.1956.1056813. 9

[12] Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2):137 – 167, 1959. URL: http://www.sciencedirect.com/science/article/pii/S0019995859903626, doi:10.1016/S0019-9958(59)90362-6. 9, 31, 32

[13] Alonzo Church. A set of postulates for the foundation of logic. *Annals of Mathematics, Series 2*, 33:346–266, 1932. 61

[14] Alonzo Church. A note on the *Entscheidungsproblem*. *The Journal of Symbolic Logic*, 1(1):40–41, 1936. URL: http://www.jstor.org/stable/2269326. 8

[15] Alonzo Church. *Introduction to mathematical logic*. Princeton Landmarks in Mathematics. Princeton University Press, Princeton, NJ, 1996. Reprint of the second (1956) edition, Princeton Paperbacks. 85

[16] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM. URL: http://doi.acm.org/10.1145/800157.805047, doi:10.1145/800157.805047. 103

[17] René Cori and Daniel Lascar. *Logique mathématique: Calcul propositionnel, algèbres de Boole, calcul des prédicats. Cours et exercices. I*. Axiomes. Collection de Logique Mathématique. [Axioms. Collection in Mathematical Logic]. Dunod, Paris, 2003. re-impression de l'édition de 1993, Masson. Préface de J.-L. Krivine. 47, 60

[18] René Cori and Daniel Lascar. *Logique mathématique: Fonctions récursives, théorème de Gödel, théorie des ensembles, théorie des modèles. Cours et exercices. II*. Dunod, Paris, 2003. Re-impression de l'édition de 1993, Masson. Préface de J.-L. Krivine. 47, 60

[19] Martin Davis, editor. *The undecidable. Basic papers on undecidable propositions, unsolvable problems and computable functions*. Edited by Martin Davis. Raven Press, Hewlett, N.Y., 1965. 8

[20] Martin Davis, Hilary Putnam, and Julia Robinson. The decision problem for exponential diophantine equations. *Ann. of Math. (2)*, 74:425–436, 1961. 3

[21] Andreas de Vries. Algebraic hierarchy of logics unifying fuzzy logic and quantum logic, 2007. 59 pages. URL: http://arXiv.org/abs/0707.2161. 60

[22] Patrick Dehornoy. *La théorie des ensembles*. Calvage et Mounet, Paris, 2017. 1

[23] Persi Diaconis, Susan Holmes, and Richard Montgomery. Dynamical bias in the coin toss. *SIAM Rev.*, 49(2):211–235, 2007. URL: http://dx.doi.org/10.1137/S0036144504446436, doi:10.1137/S0036144504446436. 116, 117

[24] Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Heidelberg, fourth edition, 2010. URL: http://dx.doi.org/10.1007/978-3-642-14279-6, doi:10.1007/978-3-642-14279-6. 11

[25] Apostolos Doxiadis and Christos H. Papadimitriou. *Logicomix*. Bloomsbury Press, New York, 2009. An epic search for truth, Character design and drawings by Alecos Papadatos, color by Annie Di Donna. 60

[26] Adolf Fraenkel. *Einleitung in die Mengenlehre*. Dover Publications, New York, N.Y., 1946. Reprint of the 3rd edition, Springer-Verlag (1928). 6

[27] Gottlob Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Verlag von Louis Nebert, Jena, 1879. Document disponible à la Bibliothèque nationale de France. 4

[28] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatsh. Math. Phys.*, 38(1):173–198, 1931. URL: http://dx.doi.org/10.1007/BF01700692, doi:10.1007/BF01700692. 6, 8, 59

[29] Kurt Gödel. *On formally undecidable propositions of Principia mathematica and related systems*. Dover Publications Inc., New York, 1992. Translated from the German and with a preface by B. Meltzer, With an introduction by R. B. Braithwaite, Reprint of the 1963 translation. 6

[30] Petr Hájek. *Metamathematics of fuzzy logic*, volume 4 of *Trends in Logic—Studia Logica Library*. Kluwer Academic Publishers, Dordrecht, 1998. URL: https://doi.org/10.1007/978-94-011-5300-3, doi:10.1007/978-94-011-5300-3. 52

[31] Michael A. Harrison. *Introduction to formal language theory*. Addison-Wesley Publishing Co., Reading, Mass., 1978. 31

[32] David Hilbert. Mathematische Probleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, pages 253–297, 1900. Vortrag, gehalten aus dem internationalen Mathematiker-Kongreß zu Paris 1900. URL: http://gdz.sub.uni-goettingen.de/dms/load/toc/?PPN=PPN252457811_1900. 2, 3

[33] D. A. Huffman. The synthesis of sequential switching circuits. I, II. *J. Franklin Inst.*, 257:161–190, 275–303, 1954. URL: https://doi.org/10.1016/0016-0032(54)90574-8, doi:10.1016/0016-0032(54)90574-8. 9

[34] Joseph B. Keller. The probability of heads. *Amer. Math. Monthly*, 93(3):191–197, 1986. URL: http://dx.doi.org/10.2307/2323340, doi:10.2307/2323340. 116

[35] S. C. Kleene. Representation of events in nerve nets and finite automata. In *Automata studies*, Annals of mathematics studies, no. 34, pages 3–41. Princeton University Press, Princeton, N. J., 1956. 9

[36] Kojiro Kobayashi. On the structure of one-tape nondeterministic Turing machine time hierarchy. *Theoret. Comput. Sci.*, 40(2-3):175–193, 1985. URL: https://doi.org/10.1016/0304-3975(85)90165-3, doi:10.1016/0304-3975(85)90165-3. 99

[37] A. N. Kolmogorov. Three approaches to the definition of the concept "quantity of information". *Problemy Peredači Informacii*, 1(vyp. 1):3–11, 1965. 108, 110, 111

[38] Andrei Nikolaevich Kolmogorov. Combinatorial foundations of information theory and the calculus of probabilities. *Uspekhi Mat. Nauk*, 38(4(232)):27–36, 1983. 9

[39] A. K. Lenstra and H. W. Lenstra, Jr., editors. *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1993. URL: http://dx.doi.org/10.1007/BFb0091534, doi:10.1007/BFb0091534. 2

[40] L. A. Levin. The concept of a random sequence. *Dokl. Akad. Nauk SSSR*, 212:548–550, 1973. 115

[41] Leonid Levin. Universal'nye zadachi perebora. *Problemy Peredachi Informatsii*, 9:115–116, 1973. Universal brute force search problems. 103

[42] Ming Li and Paul Vitányi. *An introduction to Kolmogorov complexity and its applications*. Texts in Computer Science. Springer, New York, third edition, 2008. URL: http://dx.doi.org/10.1007/978-0-387-49820-1, doi:10.1007/978-0-387-49820-1. 108

[43] Rudolf Lidl and Günter Pilz. *Applied abstract algebra*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, second edition, 1998. 11

[44] Michel Marchand. *Mathématiques discrètes*. Outils mathématiques pour l'informaticien. De Boeck, Bruxelles, 2005. 11

[45] Per Martin-Löf. The definition of random sequences. *Information and Control*, 9:602–619, 1966. 115

[46] Yuri V. Matiyasevich. *Hilbert's tenth problem*. Foundations of Computing Series. MIT Press, Cambridge, MA, 1993. Translated from the 1993 Russian original by the author, With a foreword by Martin Davis. 3

[47] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, January 1998. URL: http://doi.acm.org/10.1145/272991.272995, doi:10.1145/272991.272995. 113

[48] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, 5:115–133, 1943. 9

[49] George H. Mealy. A method for synthesizing sequential circuits. *Bell System Tech. J.*, 34:1045–1079, 1955. URL: https://doi.org/10.1002/j.1538-7305.1955.tb03788.x, doi:10.1002/j.1538-7305.1955.tb03788.x. 9

[50] Cristopher Moore and Stephan Mertens. *The nature of computation*. Oxford University Press, Oxford, 2011. 31, 99, 107

[51] Edward F. Moore. Gedanken-experiments on sequential machines. In *Automata studies*, Annals of mathematics studies, no. 34, pages 129–153. Princeton University Press, Princeton, N. J., 1956. 9

[52] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, Cambridge, 1995. URL: http://dx.doi.org/10.1017/CBO9780511814075, doi:10.1017/CBO9780511814075. 107

[53] Stefan Müller-Stach, editor. *Richard Dedekind: Was sind und was sollen die Zahlen? Stetigkeit und Irrationale Zahlen*. Klassische Texte der Wissenschaft. Springer Spektrum, 1 edition, 2017. 5

[54] Roger Penrose. *The emperor's new mind*. The Clarendon Press Oxford University Press, New York, 1989. Concerning computers, minds, and the laws of physics, With a foreword by Martin Gardner. 60

[55] Pavel Pták and Sylvia Pulmannová. *Orthomodular structures as quantum logics*, volume 44 of *Fundamental Theories of Physics*. Kluwer Academic Publishers Group, Dordrecht, 1991. Translated from the 1989 Slovak original by the authors. 60

[56] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Develop.*, 3:114–125, 1959. URL: https://doi.org/10.1147/rd.32.0114, doi:10.1147/rd.32.0114. 9

[57] Miklós Rédei. *Quantum logic in algebraic approach*, volume 91 of *Fundamental Theories of Physics*. Kluwer Academic Publishers Group, Dordrecht, 1998. 60

[58] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21(2):120–126, 1978. URL: http://dx.doi.org/10.1145/359340.359342, doi:10.1145/359340.359342. 2

[59] C.-P. Schnorr. Process complexity and effective random tests. *J. Comput. System Sci.*, 7:376–388, 1973. Fourth Annual ACM Symposium on the Theory of Computing (Denver, Colo., 1972). 115

[60] Henry Maurice Sheffer. A set of five independent postulates for Boolean algebras, with application to logical constants. *Trans. Amer. Math. Soc.*, 14(4):481–488, 1913. URL: https://doi.org/10.2307/1988701, doi:10.2307/1988701. 47

[61] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. 2

[62] Jan Šindelář and Pavel Boček. Kolmogorov complexity, pseudorandom generators and statistical models testing. *Kybernetika*, 38(6):747–759, 2002. URL: http://eudml.org/doc/33616. 112

[63] Michael Sipser. *Introduction to the theory of computation*. Cengage Learning, Boston, MA, 2012. Third edition. 31

[64] Peter Smith. *An Introduction to formal logic*. Cambridge University Press, 2003. 47

[65] Peter Smith. *An introduction to Gödel's theorems*. Cambridge Introductions to Philosophy. Cambridge University Press, Cambridge, second edition, 2013. URL: http://dx.doi.org/10.1017/CBO9781139149105, doi:10.1017/CBO9781139149105. 6, 58, 60

[66] B.A. Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, Oct 1984. doi:10.1109/MAHC.1984.10036. 103

[67] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc.*, S2-42(1):230, 1936. URL: http://dx.doi.org/10.1112/plms/s2-42.1.230, doi:10.1112/plms/s2-42.1.230. 8, 61, 85

[68] Alan M. Turing. Computability and λ-definability. *The Journal of Symbolic Logic*, 2(4):153–163, 1937. URL: http://www.jstor.org/stable/2268280. 8

[69] V. A. Uspenskiĭ, A. L. Semenov, and A. Kh. Shen'. Can an (individual) sequence of zeros and ones be random? *Uspekhi Mat. Nauk*, 45(1(271)):105–162, 222, 1990. URL: http://dx.doi.org/10.1070/RM1990v045n01ABEH002321, doi:10.1070/RM1990v045n01ABEH002321. 108

[70] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1–3):1–336, 2011. URL: http://dx.doi.org/10.1561/0400000010, doi:10.1561/0400000010. 107

[71] V. S. Varadarajan. *Geometry of quantum theory*. Springer-Verlag, New York, second edition, 1985. Reprint of the original edition published in two volumes, volume I in 1968 and volume 2 in 1970. 60

[72] Richard von Mises. *Wahrscheinlichkeit, Statistik und Wahrheit*. Springer-Verlag, Vienna-New York, 1972. Vierte Auflage, durchgesehen von Hilda Geiringer, Library of Exact Philosophy, Vol. 7. 108

[73] John von Neumann. Various techniques used in connection with random digits. *J. Res. Nat. Bur. Stand. Appl. Math. Series*, 3:36–38, 1951. 107

[74] Alfred North Whitehead and Bertrand Russell. *Principia mathematica, Volume 1*. Cambridge at the University Press, Cambridge, 1910. 6

[75] Alfred North Whitehead and Bertrand Russell. *Principia mathematica, Volume 2*. Cambridge at the University Press, Cambridge, 1927. 6

[76] Alfred North Whitehead and Bertrand Russell. *Principia mathematica, Volume 3*. Cambridge at the University Press, Cambridge, 1927. 6

[77] Alfred North Whitehead and Bertrand Russell. *Principia mathematica to *56*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, 1997. Reprint of the second (1927) edition. URL: http://dx.doi.org/10.1017/CBO9780511623585, doi:10.1017/CBO9780511623585. 6, 85

[78] E. Zermelo. Untersuchungen über die Grundlagen der Mengenlehre. I. *Math. Ann.*, 65(2):261–281, 1908. URL: https://doi.org/10.1007/BF01449999, doi:10.1007/BF01449999. 5

Index

algorithme
  Monte Carlo, 100
alphabet, 12
ancêtre, 21
application
  calculable, 106
arbre
  arbre
    des dérivations, 36
  enraciné, 21
  hauteur de l'–, 22
  nœud de –, 21
  non-dirigé, 24
  ordonné, 22
  positionnel, 24
  racine de –, 21
  sous-, 23
automate, 31
  fini déterministe, 64
  fini non-déterministe, 67
  déterministe, 35
  digraphe de l'—, 64, 67
  fini non-déterministe généralisé, 71
  langage reconnu par —, 64, 67
  non-déterministe, 35

base, 20

chaoticité, 106
clôture
  monoïdale, 12
  transitive, 20
complétude
  NP, 99
complexité
  algorithmique temporelle, 93
  classe de — P, 94
  classe de — NP, 98
  de Kolmogorov, 106, 107
concaténation, 12

degré
  entrant, 17
  sortant, 17
descendant, 21
digraphe, 17
  d'un automate, 67
  d'un automate fini, 64

ensemble
  effectivement négligeable, 110
  néglgeable, 109
entropie (de Kolmogorov), 107
expression régulière, 69

feuillage, 20
fonction
  calculable, 99, 107
  Turing-calculable, 89
formule
  satisfaisable, 49
formule booléenne, 99
  k-satisfaisable, 100
formule propositionnelle, 47

grammaire, 31, 33
  ambigue, 37
  axiome d'une –, 33
  productions d'une –, 33
graphe
  connexité de –, 18
  d'une relation, 17
  dirigé, 17
  trajectoire, 17

langage, 31, 34
  décidé par machine de Turing, 81
  reconnu par machine de Turing, 81
  reconnu par un automate, 64, 67
  réductible, 99
  régulier, 65
  Turing-décidable, 81
  Turing-reconnaissable, 81

machine de Turing, 80
  décideuse, 81

non-déterministe, 95
probabiliste, 100
vérificatrice, 98
modele
modèle, 49
monoïde
générateur de –, 32
libre, 32
mot, 12

paire composable, 18
prédécesseur, 21
préfixe, 33

quantificateur
existentiel, 52
universel, 52

R-chemin, 16
cycle, 16
hamiltonien, 16
sans recoupement, 16
simple, 25
relation
acyclique, 20
anti-réflexive, 15
anti-symétrique, 15
circulaire, 18
clôture d'une –, 19
complémentaire, 13
composition, 14
conjonction, 14
cyclique, 18
d'accessibilité, 20
d'ordre partiel, 18
d'ordre strict, 18
disjonction, 14
domaine de, 13
identique, 13
image de, 13
indice de –, 79
réciproque, 13
réflexive, 15
représentation d'une —, 13
restriction d'une —, 14
simplement acyclique, 25
symétrique, 15
transitive, 18

semi-groupoïde, 18
semigroupe
générateur de –, 32
libre, 32
stabilité fréquentielle, 104
successeur, 21
suffixe, 33
suite
chaotique, 108
légale, 105
stochastique, 105
symboles
non-terminaux, 33
terminaux, 33

temps
de calcul, 93

valuation, 49