Théorie des langages

Olivier Ridoux - 2020

Motivations

Les systèmes informatiques permettent de réaliser de façon automatique un éventail toujours plus varié de tâches. Celles-ci commencent souvent par lire des données selon un format d'entrée, pour ensuite en calculer d'autres selon un autre format. La **théorie des langages** (**TL**) permet d'étudier formellement dans quelles conditions ces tâches peuvent être automatisées. Comment lire une donnée et décider si elle est au bon format ? Comment produire une donnée qui respecte un format attendu ? Et à la base de tout, comment caractériser un format ?

Cette motivation d'ingénierie informatique n'est pas la motivation historique du développement de la théorie des langages dans les années 1930-50. Deux sources indépendantes en sont l'origine :

1. Formaliser des questions **linguistiques**. Comment caractériser une langue naturelle ? Comment reconnaître qu'une phrase est correctement formée ? Les premiers modèles formels qui ont été proposés dans ce but se sont révélés décevants, mais bien adaptés aux applications informatiques qui sont apparues plus tard. Ce sont ceux-ci qui sont présentés ici.

2. Formaliser des questions de **télécommunication**. Comment reconnaître qu'un message n'a pas été altéré par le bruit de transmission ? Comment décrire l'organisation des messages et de leur signalisation, numérotation, etc. ? Comment retrouver les éléments de signalisation dans un message ?

La TL répond à ces questions par des résultats formels très puissants qui ont permis de développer des outils qui sont dans tous les systèmes informatiques : compilateurs, gestionnaires de documents structurés, etc. La TL offre donc un magnifique exemple de **théorie qui marche** propre à une **bonne ingénierie**.

Ce module est accompagné de travaux pratiques utilisant des outils modernes comme **Xtext**.

Lettres, mots et langages

Comme toutes les mathématiques, la TL utilise des termes pré-existants pour désigner des concepts spécifiques. Il faut utiliser ces termes formels avec exactitude sans les confondre avec les termes homonymes du sens commun.

Lettre : Il s'agit de signes distincts deux à deux (on dit aussi **symbole**). Un **alphabet**, ou **vocabulaire**, souvent noté Σ ou V, est un ensemble de lettres, toujours fini dans la théorie décrite ici. Les lettres génériques sont souvent notées **a**, **b**, … On appelle **lettre-*X*** (**symbole-*X***) une lettre formelle qui est un *X* au sens commun.

Le concept formel de lettre évoque celui de lettre d'un alphabet au sens commun, mais on peut l'interpréter beaucoup plus largement, ex. par des fréquences sonores. L'important est que l'ensemble des signes soit fini et que les signes soient aisément distinguables. Ex. les **lettres-fréquences** pourront être 262 Hz, 277 Hz, 294 Hz, …, 523 Hz, mais pas le continuum des fréquences entre 262 et 523 Hz.

Mot : Un mot est une séquence de lettres, qui peut être vide, et qui dans la théorie décrite ici est toujours finie. Dans un mot non vide il y a une première lettre, une seconde, etc., et une dernière lettre. On note usuellement un mot générique par un **w** (pour ***word***), ou les lettres voisines, un mot particulier non vide par la suite de ses lettres, ex. **a** ou **abab**, et le mot vide par ε. On dira mot-*X* pour désigner des mots formels qui sont des *X* au sens commun. Ex. une séquence de **lettres-fréquences** pourrait former un **mot-mélodie**, et une séquence de **lettres-mots** pourrait former un **mot-phrase** (où on voit que le mot du sens commun n'est pas forcément un mot formel).

La théorie des langages confère une structure algébrique aux mots en les dotant d'une opération, appelée **concaténation** et notée $\bullet$. Elle consiste à mettre bout à bout deux mots pour en former un autre. Ex. **ab•cd = abcd**. La concaténation est associative, ce qui permet d'écrire **u•v•w** sans parenthèses, et même **uvw**. Elle a un élément neutre à gauche et à droite, qui est ε. Elle n'est **pas commutative**.

L'ensemble de tous les mots qu'il est possible de former avec les lettres d'un vocabulaire **V** (resp. Σ) est noté **V*** (resp. Σ^*). Cet ensemble est infini même si ses éléments sont des mots finis. Il est **dénombrable**.

Langage : Un langage est une partie de V*. Les langages sont habituellement notés $\mathcal{L}$. Étant donné un vocabulaire V, on peut former une infinité de langages, dont $\emptyset$ et V*. L'ensemble constitué de tous les langages **n'est pas dénombrable**.

La TL confère une structure algébrique aux langages issus d'un même vocabulaire en les dotant d'opérations. L'une d'entre elles, appelée **produit** et notée $\bullet$, permet de former un nouveau langage en concaténant les mots de deux langages :

$$\mathcal{L}_1\bullet\mathcal{L}_2 = \{\, w_1\bullet w_2 \mid w_1\in\mathcal{L}_1 \wedge w_2\in\mathcal{L}_2 \,\}.$$

Le produit est associatif et a $\{\varepsilon\}$ pour élément neutre à gauche et à droite. Il n'est **pas commutatif**. Le langage vide est un élément absorbant, $\mathcal{L}\bullet\emptyset = \emptyset\bullet\mathcal{L} = \emptyset$! Et ε n'est pas un langage. Confondre ε, $\{\varepsilon\}$ et $\emptyset$ constitue donc une grave erreur, comme le serait confondre le chiffre **'0'** et les nombres **0** et **1** pour la **multiplication** usuelle.

Une autre opération, appelée **fermeture** et notée *, permet de former un nouveau langage par itération de la concaténation :

$$\mathcal{L}^* = \{\varepsilon\} \cup \mathcal{L} \cup \mathcal{L}\bullet\mathcal{L} \cup \mathcal{L}\bullet\mathcal{L}\bullet\mathcal{L} \cup \dots$$

$\mathcal{L}^*$ est toujours infini, sauf si $\mathcal{L}$ est vide. Cet opérateur est aussi appelé **« étoile de Kleene »** du nom du mathématicien Stephen Kleene (1909-94) à qui on doit de nombreux résultats de la TL.

Les langages intéressants sont ceux qui ne sont ni $\emptyset$ ni V*, et qui sont gros voire infinis. On ne peut donc pas les représenter en extension, et c'est une contribution importante de la théorie des langages que de proposer des moyens finis et compacts pour représenter des langages gros voire infinis. Ex. un langage formel comme **Java** est l'ensemble des **mots-programmes** qui respectent la spécification de ce langage. Cet ensemble est gros, même infini, et correspond bien à la notion informelle de langage intéressant. Mais l'enjeu de la TL n'est pas simplement de caractériser Java, mais plutôt de caractériser des **moyens** de caractériser des langages, de **classifier** ces moyens, et d'en déterminer le **pouvoir d'expression**. Ex. les opérations ci-dessus suffisent-elles à caractériser le langage Java ?

Comment caractériser formellement un langage ?

La TL propose deux styles de caractérisation d'un langage :
1. Former des langages plus gros en assemblant des langages plus petits. C'est un style **extensionnel**, adapté aux humains.
2. Donner une procédure de décision qu'un mot appartient à un langage. C'est un style **intensionnel**, adapté aux machines.

L'idéal serait bien sûr d'avoir une **passerelle** entre les deux, de façon à ce qu'un humain puisse caractériser commodément, de son point de vue, un langage dans un style extensionnel, puis traduire cette caractérisation en style intensionnel plus commode pour une machine. Beaucoup d'exemples de telle passerelle existent, et c'est tout l'intérêt de la TL que de pouvoir le démontrer formellement (ex. page 5).

Expressions régulières (re, pour ***regular expression*****) :** Elles sont un exemple du style **extensionnel** pour caractériser les langages. La théorie décrit comment former des re, et quels langages celles-ci dénotent. On note une re générique **r**, et **RE** l'ensemble des re qu'il est possible de former. RE est **dénombrable**.

Soit **V = {a, b, …}**, les re de base sont $\underline{\varepsilon}$, **a̲**, **b̲**, … Elles décrivent les petits langages qui en composent de plus gros. Le soulignement sert ici à distinguer l'utilisation des lettres du **langage décrit** dans les re **qui le décrivent**. Soit **r**, r_1 et r_2 des re, $r_1|r_2$, $r_1\, r_2$ et **r*** sont aussi des re. On pourra utiliser des parenthèses pour marquer la portée des **|** et des *. Les re dénotent des langages de la façon suivante :

- $SEM(\underline{\varepsilon}) = \{\varepsilon\}$ $SEM(\underline{a}) = \{a\}$ $SEM(\underline{b}) = \{b\}$ …
- $SEM(r_1|r_2) = SEM(r_1) \cup SEM(r_2)$ union de deux langages
- $SEM(r_1\, r_2) = SEM(r_1) \bullet SEM(r_2)$ produit de deux langages
- $SEM(r^*) = SEM(r)^*$ itération d'un langage

RE constitue donc un langage qui décrit des langages ; c'est un **métalangage**.

Ex. la re **###4/4(CDFCBCACCBAFFCCB)*(EEEEFBBBBBBBBBBAAAAA)*CDECBCA** décrit les premières notes du *Yellow Submarine* des Beatles (C=Do, D=Ré, etc.) où **V** est **{#, 4, /, A, B, C, D, E, F, G}**. Ici, la re engendre des **mots-partitions**.

On qualifie de **rationnels** les langages qu'il est possible de caractériser par une re. L'ensemble des langages n'étant pas dénombrable mais RE l'étant, les langages ne sont pas tous rationnels ! Mais alors, lesquels le sont ? Lesquels ne le sont pas ?

Automates finis (fsa, pour ***finite state automata*****) :** Ils sont un exemple du style **intensionnel** de caractérisation des langages. La théorie décrit comment former des fsa, et comment ceux-ci **acceptent** des mots. On note un fsa générique **M** (pour ***Machine***). Chaque fsa est fini. Les fsa constituent un ensemble **dénombrable** : **FSA**.

Un fsa est défini par un vocabulaire V, un ensemble fini d'états, noté **Q**, contenant un état initial $q_0 \in Q$, et des états finaux $F \subseteq Q$. Il est aussi défini par une relation de transition δ constituée d'un ensemble de triplets $(q, \underline{a}, q')$, souvent notés $q \xrightarrow{a} q'$, où $a \in V$ et $q, q' \in Q$. Les fsa **acceptent** des mots de la façon suivante :

- $ACCEPT(\varepsilon, q)$ est vrai ssi $q \in F$
  les états finaux acceptent le mot vide, ils sont les seuls à le faire
- $ACCEPT(a\bullet m, q)$ est vrai ssi $\exists\, q'$ tq $q \xrightarrow{a} q'$. $ACCEPT(m, q')$
  un état quelconque accepte un mot si il permet une transition qui accepte sa première lettre et qui conduit à un état qui accepte le reste
- un fsa **accepte** un mot m ssi $ACCEPT(m, q_0)$.

Un fsa **reconnaît** un langage si il en accepte tous les mots, et seulement eux.

Théorème de Kleene : Les langages rationnels sont les langages reconnus par les fsa, et vice-versa. Cela rend possible la passerelle attendue entre les re, adaptées aux humains, et les fsa, adaptées aux machines. Cette passerelle est réalisée dans des outils comme **Perl**, le **grep** du shell UNIX, le **REGEXP** de SQL, **LEX** et **Xtext**.

On a vu que les langages ne pouvaient pas tous être rationnels ; comment distinguer ceux qui ne le sont pas ? Le lemme suivant donne un critère.

Lemme de l'étoile (simplifié) : Si un langage est rationnel alors chacun de ses mots plus longs qu'une limite fixée peut être découpé en 3 tronçons **u•v•w** tels que **u** n'est pas trop long et **{u}•{v}*•{w}** est inclus dans le langage. On démontre qu'un langage n'est pas rationnel en prenant la **contraposée** du critère. En fait, de nombreux langages ne sont pas rationnels :

- Langage des palindromes : le langage des mots qui se lisent dans les deux sens (ex. **252**, **kayak** et **ressasser**).
- Langage des parenthèses : le langage des mots balisés par des parenthèses bien imbriquées, ex. **(…)(((((…)))((…))))(…))**, quel que soit le degré d'imbrication.
- RE : à cause des parenthèses qui marquent la portée des **|** et des *. Le métalangage qui décrit les langages rationnels n'est donc pas rationnel. Et Java non plus, à cause de ses parenthèses et de ses accolades.

Avec le Lemme de l'étoile se clôt cette présentation des langages rationnels. Il faut surtout en retenir l'architecture générale : l'échafaudage **lettre-mot-langage**, les moyens de **caractériser des langages**, par **expression** (extensionnelle) et par **automate** (intensionnel), **l'équivalence** formelle entre ces moyens, l'identification de leur **puissance d'expression**, et la formalisation de cette puissance d'expression dans un **critère d'appartenance**.

La TL ne s'arrête pas aux langages rationnels. Que certains langages ne soient pas rationnels fait se demander si on peut construire le même genre d'édifice pour eux, ou une partie d'entre eux. Et la réponse est oui. Par exemple, on montre que le langage des parenthèses est l'archétype d'une famille de langages qui sont bien caractérisés par une autre forme d'expression, les **grammaires algébriques**, et par des **automates à pile**, que les deux ont exactement la même puissance d'expression, qui est bien testée par une variante du Lemme de l'étoile, le tout constituant la théorie des **langages algébriques** (on dit aussi **langage sans contexte**). On montre aussi que les langages ne sont pas tous algébriques, et on recommence en définissant des classes de langages toujours plus larges qu'on ordonne dans la **Hiérarchie de Chomsky**, du nom de Noam Chomsky (1928-…), autre grand nom de la TL. Dans cette hiérarchie, les classes de langages sont désignées par un numéro ; ex. les langages rationnels sont les langages de **type 3** (voir page 7).

Analyse syntaxique

Jusque là l'enjeu était de caractériser l'appartenance d'un mot à un langage. Mais ce n'est pas suffisant car les applications qui lisent des données formatées veulent en plus utiliser le format comme un guide pour leur calcul.

Depuis Chomsky, on privilégie des formes de caractérisation des langages qu'on appelle **grammaire**. On y trouve les re, avec les lettres d'un vocabulaire, plus un second vocabulaire qui donne des noms aux composants structurels d'un document. On appelle le premier le vocabulaire **terminal**, noté **T**, et le second le vocabulaire **non-terminal**, noté **NT**. T correspond au lexique d'une langue naturelle, ex. **je**, **qui**, **le**, **pomme**, **aime**, **Alice**, et NT à ses structures grammaticales, ex. **pronom**, **article**, **nom-commun**, **verbe**, **nom-propre**. Enfin, des **règles grammaticales** définissent comment chaque composant structurel est formé :

- phrase ⟶ sujet verbe complément
- sujet ⟶ pronom-sujet | nom
- nom ⟶ article nom-commun | nom-propre
- pronom-sujet ⟶ **je** | **tu** | **il** | …
- verbe ⟶ **aime** | **apprend** | …
- complément ⟶ nom (ε | **qui** verbe complément)

Noter comment **complément** est défini en fonction de lui-même. Cette forme de grammaire est appelée **grammaire algébrique**, noté **GA**. D'autres formes existent.

Une grammaire **engendre** un langage de la façon suivante :
- $SEM(nt) = \bigcup_{nt \to r} SEM(r)$ où $nt \in NT$
- SEM(r) = … comme pour les expressions régulières
- La grammaire **engendre w** ssi $w \in SEM(phrase)$

Cette grammaire engendre donc des mots-phrases comme **Alice aime le violon**, mais aussi **Bob aime Alice qui apprend le violon**. L'enjeu de l'analyse syntaxique est de reconnaître que dans la première phrase **Alice** est le **sujet** et **le violon** est le **complément**, mais que dans la seconde **Alice** est une partie du **complément**.

L'analyse syntaxique présente deux difficultés, l'une liée aux grammaires, l'autre au processus d'analyse :

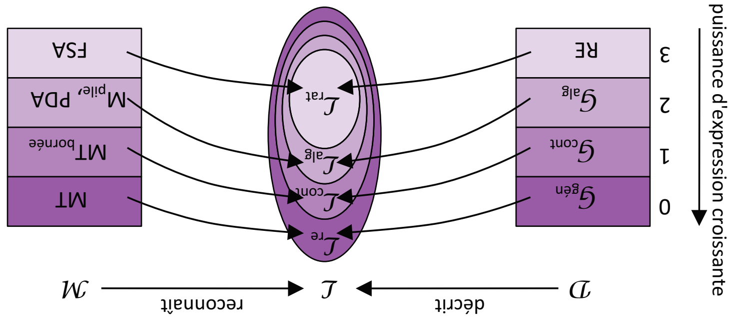
1. Ambiguïté : une grammaire peut donner plusieurs décompositions structurelles à la même phrase, comme dans **Alice a vu Bob avec sa longue-vue** (de quoi **avec sa longue-vue** est-il le **complément** ?). À éviter absolument !

2. Incomplétude : il existe des algorithmes qui réalisent bien l'analyse syntaxique dans tous les cas, mais on leur préfère souvent des variantes plus simples qui ne marchent que pour certaines familles de grammaires. Lire la documentation !

2×2 formalismes linguistiques

	sans imbrication	avec imbrication
ensemble (extensionnel)	expressions régulières : **CSV**	grammaires algébriques : **WHILE**, CSV
reconnaisseur (intensionnel)	automates finis : **CSV**	automates à pile : **WHILE**, CSV

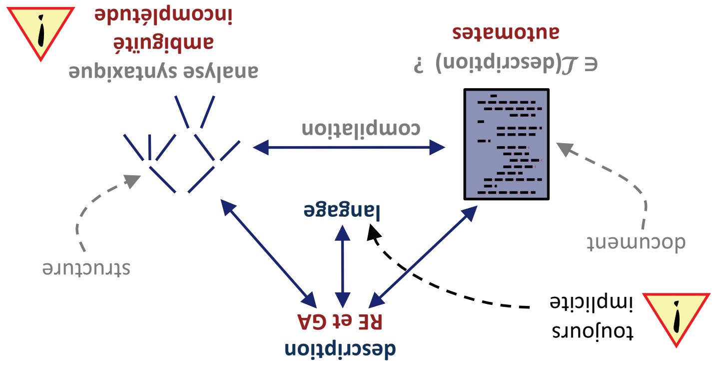
Hiérarchie de Chomsky



Type	Grammaire Chomsky, 1959	Automate	Langage
0	$(\Sigma_T \mid \Sigma_N)^+ \to (\Sigma_T \mid \Sigma_N)^*$	MT Turing, 1936	récursivement énumérable Post, 1947
1	$\alpha\, A\, \beta \to \alpha\, \gamma\, \beta$	MT bornée …, Kuroda, 1964	avec contexte
2	$\Sigma_N \to (\Sigma_T \mid \Sigma_N)^*$ **algébrique**	**à pile** Chomsky, 1962	**sans contexte** 1947-64
3	$\Sigma_N \to \Sigma_T \Sigma_N \mid \varepsilon$ **régulière (RE !)**	**fini** Kleene, 1951-56	**rationnel** Kleene, 1951-56

Conclusion

La TL, ce sont des mathématiques (définitions, théorèmes et preuves) qui marchent (algorithmes et systèmes automatiques efficaces) et qui sont utilisées partout. Abandonner la pensée magique !



Les expressions régulières et les grammaires **engendrent** des mots et des langages (mais pas de la même classe), et les automates **acceptent** des mots et **reconnaissent** des langages. Ils constituent les outils de base pour formaliser une grande variété de structures séquentielles organisées : programmes, documents, traces d'exécution, protocoles, etc., mais aussi architecture, croissance des plantes, musique, etc.

En pratique, utiliser la TL revient à **programmer** avec des expressions régulières, des grammaires ou des automates. Il faut toujours s'assurer que ce programme engendre le langage désiré, et pour cela il faut **absolument** le tester avec des cas positifs (pour détecter la **sous-génération**) et des cas négatifs (pour détecter la **sur-génération**). Enfin, il est **indispensable** de surveiller **l'ambiguïté** des grammaires et **l'incomplétude** des analyseurs !

Bibliographie

- « **Méthodes mathématiques pour l'informatique** », Vélu (Dunod, 2005)
- « **Mathématiques pour l'informatique** », Arnold et Guessarian (Dunod, 2005)
- « ***The New Turing Omnibus*** », Dewdney (Owl Books, 2001)
- « ***Formal Languages: Origins and Directions*** », Greibach (Annals Hist. Comp., 1981)

Prérequis : logique et théorie des ensembles - lire, apprendre, comprendre.
Pliage : recto visible (autre face), traits **gris** rentrants, traits **rouges** saillants. Découper selon le trait rouge entre les deux ● de l'autre face, puis achever le pliage.

① Théorie naïve des ensembles finis

Un **ensemble** est une collection d'objets où la position et la multiplicité de comptent pas. On appelle **éléments** les objets de la collection, et on dit qu'un élément **appartient** à un ensemble. Cela se note $e \in E$ pour « *e appartient à E* » ou $e, e' \in E$ pour « *e et e' appartiennent à E* ».

On note une collection entre accolades ($\{...\}$). Ex. **{a, b, c}** est une collection, et puisque la position ne compte pas, la collection **{b, c, a}** dénote le même ensemble, et puisque la multiplicité ne compte pas non plus, la collection **{a, b, a, c}** dénote encore le même ensemble.

Dans la théorie naïve des ensembles on ne considère que des objets qu'il est possible de distinguer les uns des autres, mais à part cela n'importe quoi peut être un objet, y compris un ensemble.

Une collection vide, **{}**, constitue un ensemble particulier qu'on appelle **l'ensemble vide**, et qu'on note $\emptyset$ ou $\{\}$. On appelle **singleton** un ensemble qui n'a qu'un élément. L'ensemble $\{\emptyset\}$ est donc un singleton. Ne pas confondre l'élément **a** et le singleton **{a}**. Les confondre revient à commettre une erreur de type en programmation, comme confondre **int** et **int []**. Ne pas confondre non plus $\emptyset$ et $\{\emptyset\}$. Ce serait comme confondre un pointeur nul et un pointeur sur un pointeur nul.

La collection des éléments d'un ensemble s'appelle **l'extension** de l'ensemble. On peut aussi spécifier les éléments d'un ensemble par une propriété qui les distingue de ceux qui n'appartiennent pas à l'ensemble. Cette propriété s'appelle **l'intension** de l'ensemble (l'intenSion n'a rien à voir avec l'intenTion, ou dessein). On exprime une intension soit par une phrase en prenant le risque d'être imprécis ou ambigu, soit par une formule logique en prenant le risque d'être fastidieux. N'importe quelle formule logique fausse (contradictoire ou absurde) est une intension de $\emptyset$.

Ex. ***« les lettres communes aux mots ``bancal'' et ``tabac'' »*** est une intension possible pour l'ensemble constitué de la collection **{a, b, c}**. Les ***« 3 premières lettres de l'alphabet »*** est une intension équivalente. Un ensemble peut avoir de nombreuses intensions équivalentes, certaines même difficiles à comparer.

Si tous les éléments d'un ensemble E_1 appartiennent à un ensemble E_2, on dit que E_1 est **inclus** dans E_2, et on le note $E_1 \subset E_2$. On dit aussi que E_1 est un **sous-ensemble** ou une **partie** de E_2. On note $E_1 \subsetneq E_2$ pour signifier l'inclusion sans égalité possible (on parle alors de sous-ensemble **strict**), et $E_1 \subseteq E_2$ pour insister sur l'égalité possible. $\emptyset$ est un sous-ensemble de chaque ensemble, et tout ensemble est sous-ensemble (non-strict) de lui même. Si un ensemble a **n** éléments, il a 2^n sous-ensembles (non-strict).

① Logique des prédicats

Dans sa définition la plus naïve, la **logique des prédicats** (ou **calcul des prédicats**) est la formalisation de la logique employée tous les jours dans les activités un tant soit peu scientifiques. Elle permet d'exprimer des **jugements** sous forme de formules, et de décider formellement si elles sont **vraies** ou **fausses**. Les formules de la logique des prédicats sont définies de la façon suivante :

Prédicats atomiques ou (**atomes**) : Ce sont des formules élémentaires, c-à-d. pas composées de sous-formules, qui expriment des jugements sur des objets. Nous ne nous prononçons pas sur la notation à employer dans les atomes ; cela n'a pas vraiment d'importance. Par exemple,
- **Rennes est une capitale**, où **Rennes** est **l'objet** et **être une capitale** est le **prédicat** du jugement.
- **Rennes est la capitale de la France**, où **Rennes** et la **France** sont des objets et **être la capitale de** est le prédicat.
- **x > y**, où **x** et **y** sont les objets, et où > (être plus grand que) est le prédicat.
- **1273 + 556 est un nombre premier**, où **1273 + 556** est l'objet et **être un nombre premier** est le prédicat.
- **Le Grand théorème de Fermat est vrai**, où le **Grand théorème de Fermat** est l'objet et **être vrai** est le prédicat.
- $\mathcal{P} \neq \mathcal{NP}$, où $\mathcal{P}$ et $\mathcal{NP}$ sont les objets et $\neq$ (être différent) le prédicat.

On peut attribuer une valeur de vérité, **Vrai** ou **Faux**, à un atome, en se référant à une réalité de terrain, comme pour **Rennes**, à des théories et des calcul, comme pour la primalité de **1273 + 556**, ou à des preuves externes, comme celle du **Grand théorème de Fermat**. Parfois on ne peut pas, par manque d'information, comme pour **x > y**, ou parce qu'on ne sait vraiment pas, comme pour $\mathcal{P} \neq \mathcal{NP}$.

Formules propositionnelles : Ce sont des formules qui sont constituées de sous-formules, élémentaires ou non, qui sont reliées par des connecteurs logiques, généralement $\wedge$,$\vee$, $\neg$, ou $\Rightarrow$.
- **conjonction, $\phi_1 \wedge \phi_2$:** relie deux formules pour en former une troisième qui n'est vraie que si les deux premières le sont. Le connecteur $\wedge$ est **commutatif**, **associatif**, et a pour **élément neutre** la valeur de vérité **Vrai**. Cela permet la notation de conjonction étendue comme $\bigwedge_{i \in [1,n]} \phi_i$. Si on convenait que **Faux < Vrai**, $\phi_1 \wedge \phi_2$ serait le minimum de ϕ_1 et ϕ_2 . Si on convenait que **Faux = 0** et **Vrai=1**, $\phi_1 \wedge \phi_2$ serait $\phi_1 \times \phi_2$.
- **disjonction, $\phi_1 \vee \phi_2$:** relie deux formules pour en former une troisième qui n'est vraie que si au moins une des deux premières l'est. On parle de disjonction **exclusive** si une et une seule de ces formules peut être vraie. Le connecteur $\vee$ est **commutatif**, **associatif**, et a pour **élément neutre** la valeur de vérité **Faux**. Cela permet la notation de disjonction étendue comme $\bigvee_{i \in [1,n]} \phi_i$. Si on convenait que **Faux < Vrai**, $\phi_1 \vee \phi_2$ serait le maximum de ϕ_1 et ϕ_2. Si on convenait que **Faux = 0** et **Vrai=1**, $\phi_1 \vee \phi_2$ serait $1 - (1-\phi_1) \times (1-\phi_2)$.
- **implication, $\phi_1 \Rightarrow \phi_2$:** relie deux formules pour en former une troisième qui est fausse ssi ϕ_1 est vraie alors que ϕ_2 est fausse. Le connecteur $\Rightarrow$ n'est **ni commutatif ni associatif**. Si on convenait que **Faux < Vrai**, $\phi_1 \Rightarrow \phi_2$ serait **Vrai** ssi $\phi_1 \leq \phi_2$.
- **négation, $\neg \phi$:** constitue une formule qui est fausse ssi ϕ est vraie. Si on convenait que **Faux = 0** et **Vrai=1**, $\neg \phi$ serait $1 - \phi$.

Formules quantifiées : Ce sont des formules, élémentaires ou non, dont la valeur de vérité s'évalue par rapport à un ensemble, plutôt que par rapport à des objets.
- **quantification universelle, $\forall x . \phi(x)$:** constitue une formule qui est vraie ssi ϕ est vraie de tous les éléments du domaine. Si le domaine est fini, la quantification universelle est juste une conjonction des applications de ϕ à tous les éléments du domaine : $\forall x \in \{e_1, e_2, ..., e_n\}. \phi(x)$ ssi $\bigwedge_{i \in [1,n]} \phi(e_i)$.
- **quantification existentielle, $\exists x . \phi (x)$:** constitue une formule qui est vraie ssi ϕ est vraie d'au moins un élément du domaine. Si le domaine est fini, la quantification existentielle est juste une disjonction des applications de ϕ à tous les éléments du domaine : $\exists x \in \{e_1, e_2, ..., e_n\}. \phi(x)$ ssi $\bigvee_{i \in [1,n]} \phi(e_i)$.

Parenthèses et priorité des opérateurs : Il est courant d'assigner des priorités d'opérateurs aux différents connecteurs et quantificateurs afin de spécifier comment se lisent les formules complexes en l'absence de parenthèses. Cependant, il n'est jamais très prudent de se reposer trop lourdement sur les priorités des opérateurs quand le coût de quelques parenthèses est si faible devant le coût d'une grosse erreur. Il vaut mieux ne pas être avare de parenthèses, et nous proposons d'utiliser les parenthèses rondes (**(...)**) pour structurer les connecteurs, et les parenthèses carrées (ou crochets, **[...]**) pour les quantificateurs.

N'oubliez jamais !

Un **modèle** est toujours **imparfait**, et tout ce qu'on peut lui demander est d'être **utile** (d'après George Box 1978, voir aussi a contrario les cartes à l'échelle 1/1 de Jorge Luis Borges 1946 et Lewis Carroll 1893 qui sont parfaites mais inutiles).

② Ensembles infinis

Quand un ensemble comporte un nombre fini d'éléments on appelle ce nombre la **cardinalité** de l'ensemble, mais ce concept doit être revisité pour les ensembles infinis car il n'y a pas de nombre naturel pour mesurer l'infinité.

Si on peut établir une bijection entre deux ensembles on dit qu'ils ont la même **cardinalité**. Si un ensemble a la même cardinalité qu'une partie stricte de lui-même, on dit qu'il est **infini**. Les autres ensembles sont dits **finis**. Par exemple, la fonction $x \mapsto 2 \times x$ forme une bijection entre l'ensemble des **entiers naturels** et l'ensemble des **entiers pairs**, qui en est une partie stricte. La cardinalité modélise la quantité d'éléments d'un ensemble, mais ne peut être assimilée à un entier naturel que pour les ensembles finis.

Les ensembles infinis n'ont pas tous la même cardinalité ; certains sont « plus gros que d'autres ». Si un ensemble a la même cardinalité que celui des entiers naturels, on dit qu'il est **dénombrable**. Il est **indénombrable** sinon.

Les principaux résultats sont les suivants :
- L'ensemble des entiers naturels est dénombrable (par définition), ainsi que l'ensemble des paires d'entiers naturels, des rationnels, des triplets et nuplets, des suites finies de nombres et de symboles, et des textes (et donc des expressions et des programmes), par un **théorème de Georg Cantor (1878)**.
- L'ensemble des fonctions sur les entiers naturels n'est pas dénombrable, ainsi que l'ensemble des parties des entiers naturels, et l'ensemble des nombres réels, par un autre **théorème de Georg Cantor (1874 et 1891)**.
- L'ensemble des parties d'un ensemble **E** infini est toujours **« plus gros » que E**.

On constate que beaucoup d'ensembles intéressants (ex. les réels ou les fonctions) ne sont pas dénombrables, alors que les programmes le sont. Cela indique que ces ensembles intéressants ne sont pas calculables par des programmes.

② Idiomes logiques

En pratique, on n'utilise pas n'importe quelle formule de la logique des prédicats. On a tendance à utiliser des expressions idiomatiques qu'il convient de reconnaître et interpréter correctement au premier coup d'œil.

Quantifications dans un domaine : Très souvent, on écrit des formules comme $\forall x \in E . \phi(x)$ ou $\forall x \text{ tq } \psi(x) . \phi(x)$. C'est une façon de dire dans quel domaine doit être évaluée la quantification. Ces formules **doivent** être lues $\forall x . [x \in E \Longrightarrow \phi(x)]$ et $\forall x . [\psi(x) \Longrightarrow \phi(x)]$. Une conséquence directe est qu'une quantification universelle sur un domaine vide est trivialement vraie. Cela paraît une situation étrange, mais c'est banal en informatique, spécialement quand on considère les cas d'initialisation dans les programmes : ex. **Tous les utilisateurs sont ...** lorsqu'il n'y a pas encore d'utilisateurs. De la même façon, on écrit des formules comme $\exists x \in E . \phi(x)$ ou $\exists x \text{ tq } \psi(x) . \phi(x)$. Ces formules **doivent** être lues $\exists x . [x \in E \wedge \phi(x)]$ et $\exists x . [\psi(x) \wedge \phi(x)]$. On voit alors qu'une quantification existentielle sur un domaine vide est trivialement fausse.

Cascades d'implications : On écrit parfois des formules comme $\phi_1 \Rightarrow \phi_2 \Rightarrow \phi_3$, qui pourrait être lue $(\phi_1 \Rightarrow \phi_2) \Rightarrow \phi_3$ ou $\phi_1 \Rightarrow (\phi_2 \Rightarrow \phi_3)$. Dans le premier cas, on exprime que $\phi_1 \Rightarrow \phi_2$ est une hypothèse qui suffit à démontrer ϕ_3. Dans le second cas, la formule est équivalente à $(\phi_1 \wedge \phi_2) \Rightarrow \phi_3$, donc $(\phi_2 \wedge \phi_1) \Rightarrow \phi_3$, et donc $\phi_2 \Rightarrow (\phi_1 \Rightarrow \phi_3)$. Conclusion, faire des économies de bouts de parenthèses avec discernement !

Formules de De Morgan (Augustus De Morgan, 1806-1871) : La négation a à voir avec le complémentaire d'une situation, mais le complémentaire d'une situation compliquée est souvent encore plus compliquée que la situation. Les formules de De Morgan (qui ne sont pas toutes dues à De Morgan) peuvent être mises à profits pour pousser les négations vers l'intérieur des formules où elles s'appliqueront à des formules plus petites donc plus simples.
- $\neg (\phi_1 \wedge \phi_2)$ est équivalent à $\neg \phi_1 \vee \neg \phi_2$, et $\neg (\phi_1 \vee \phi_2)$ est équivalent à $\neg \phi_1 \wedge \neg \phi_2$.
- $\neg (\phi_1 \Rightarrow \phi_2)$ est équivalent à $\phi_1 \wedge \neg \phi_2$.
- $\neg\neg \phi$ est équivalent à ϕ . Pose beaucoup plus de difficultés que la taille de l'identité ne le laisse penser, mais est vrai pour l'usage usuel de la logique. Se rappeler combien nous humains sommes peu doués pour les doubles négations.
- $\neg \forall x . \phi(x)$ est équivalent à $\exists x . \neg \phi (x)$, et $\neg \exists x . \phi (x)$ est équivalent à $\forall x . \neg \phi(x)$.

Cascades de quantifications : On imbrique souvent les quantifications. Certaines imbrications doivent faire réfléchir, et d'autres moins.
- $\exists x . \exists y . \phi (x, y)$ est équivalent à $\exists y . \exists x . \phi (x, y)$ qu'on note souvent $\exists x, y . \phi (x, y)$.
- $\forall x . \forall y . \phi (x, y)$ est équivalent à $\forall y . \forall x . \phi (x, y)$ qu'on note souvent $\forall x, y . \phi (x, y)$.
- $\forall x . \exists y . \phi (x, y)$ exprime que pour chaque **x** il y a un **y**, **qui peut dépendre de x**, qui a la propriété désirée. Ex. dans $\forall x . \exists y . x \times y = 0$, le même **y** convient pour tous les **x**, mais dans $\forall x . \exists y . x \times y = 1$, à tout **x** correspond un **y** différent. Une variable existentielle est donc implicitement une fonction de toutes les variables universelles qui la précèdent.
- $\exists x . \forall y . \phi (x, y)$ exprime qu'un même **x** a la propriété ϕ pour tous les **y**. Ex. $\exists x . \forall y . x \times y = 0$ est vrai, mais pas $\exists x . \forall y . x \times y = 1$.
- $\exists x . [\psi \curlywedge \phi(x)]$ est équivalent à $\psi \curlywedge \exists x . \phi(x)$ et $\forall x . [\psi \curlywedge \phi(x)]$ est équivalent à $\psi \curlywedge \forall x . \phi(x)$ si **x** n'apparaît pas dans ψ, et $\curlywedge$ représente l'un des connecteurs $\wedge$ ou $\vee$.

③ Algèbre des ensembles

Étant donné un **univers** $\mathcal{U}$ d'objets de référence, on peut former une structure algébrique sur les sous-ensembles de $\mathcal{U}$ avec les opérations suivantes :

Union (notée $\bigcup$) : $A \cup B$ est l'ensemble formé de la collection des éléments de A et de ceux de B. Comme la multiplicité ne compte pas, il est sans conséquence qu'un élément appartienne aux deux ensembles ou seulement à l'un d'entre eux, et alors auquel des deux. Par exemple, $\{a, b\} \cup \{c, b\} = \{a\} \cup \{b, c\} = \{a, b, c\}$. L'union est **commutative**, **associative**, **idempotente** et a $\emptyset$ pour **élément neutre**. Cela autorise la notation d'union étendue comme $\bigcup_{i \in [1,n]} E_i$. L'intension d'une union d'ensembles est la disjonction des intensions de ces ensembles. Noter que $A \subseteq B$ ssi $A \cup B = B$ et que $\bigcup_{i \in \emptyset} E_i = \emptyset$.

Intersection (notée $\bigcap$) : $A \cap B$ est l'ensemble formé de la collection des éléments de A qui appartiennent aussi à B. Par exemple, $\{a, b\} \cap \{c, b\} = \{b\}$. L'intersection est **commutative**, **associative**, **idempotente** et a $\mathcal{U}$ pour **élément neutre**. Cela autorise la notation d'intersection étendue comme $\bigcap_{i \in [1,n]} E_i$. L'intension d'une intersection d'ensembles est la conjonction des intensions de ces ensembles. Noter que $A \subseteq B$ ssi $A \cap B = A$ et que $\bigcap_{i \in \emptyset} E_i = \mathcal{U}$.

L'intersection et l'union sont **distributives** l'une par rapport à l'autre ; pour tout triplet d'ensembles **A**, **B** et **C**, $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ et $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$. C'est une propriété fondamentale de la théorie des ensembles.

Complémentation (notée $\complement$) : $\complement_A B$ est l'ensemble formé de la collection des éléments de A qui n'appartiennent pas à B : le **complémentaire** de B dans A. On note parfois $A \backslash B$ et même $A - B$. Il n'est pas nécessaire que B soit inclus dans A ! Quand l'ensemble de référence est $\mathcal{U}$ on peut se dispenser de le noter : $\complement B = \complement_{\mathcal{U}} B$. Noter que $\complement \complement B = B$. L'intension du complémentaire d'un ensemble est la conjonction de l'intension de l'ensemble de référence et de la négation de l'intension de l'ensemble. La complémentation est une sorte de négation et suit les **lois de De Morgan** : $\complement_A(B \cup C) = \complement_A B \cap \complement_A C$ et $\complement_A(B \cap C) = \complement_A B \cup \complement_A C$.

Même si l'une fait penser à l'addition et l'autre à la soustraction, **l'union et la complémentation ne sont pas les opposées l'une de l'autre** ; pour certaines paires d'ensembles A et B, on a $(A \backslash B) \cup B \neq A$ ou $(A \cup B) \backslash B \neq A$. Ex. $(\{a\} \backslash \{b\}) \cup \{b\} = \{a, b\}$ et $(\{a, b\} \cup \{b\}) \backslash \{b\} = \{a\}$.