

Mode d'emploi :

- Lire cette page de documentation (**histoire**, **prérequis logiques**, **opinion**), comprendre, apprendre le cas échéant.
- Préparer les plis (voir traits **gris** rentrants et traits **rouges** saillants au verso de cette page).
- Découper selon le trait rouge entre les deux ● du verso de cette page, puis achever le pliage.
- S'occuper du découpage de la machine de Turing tangible *lowcost* en dernier.

Numérique, digital, binaire, …, ou symbolique ?

Aujourd'hui, on ne dit pas « l'informatique » mais « le **numérique** » ! Ce terme fait suite à **digital** et **binaire**. Dans tous les cas, l'idée de nombre est mise en avant. De même, on ne programme plus, mais on code, ce qui par voisinage de sens ramène à chiffrer, et encore aux nombres.

Nous pensons que cette mode renforce une perception ésotérique de l'informatique, une forme de **numérologie**, et qu'elle contribue par là à l'incompréhension de ce qu'est l'informatique, **Ah vous êtes informaticien.ne ! Vous voulez vraiment représenter tout avec des nombres ? N'est-ce pas réductionniste ? Vous ne pourrez jamais représenter la vibration des couleurs des tableaux de Turner avec des nombres !**

…alors qu'on peut passer une vie d'informaticien.ne sans jamais se soucier de coder quoi que ce soit avec des nombres.

Les travaux de Turing montrent un visage beaucoup plus intéressant ; celui du raisonnement sur les symboles et les lois qui les organisent, selon la tradition très ancienne de recherche d'éléments premiers et de lois de production d'éléments dérivés. C'est ce que font les physiciens et chimistes et leurs ancêtres alchimistes depuis l'antiquité, les linguistes depuis Panini (grammaire du sanskrit, il y a environ 6000 ans), les biologistes, physiologistes et médecins depuis toujours, et bien sûr les mathématiciens. Tous créent des symboles pour donner des noms aux choses, et cherchent des lois pour les organiser. Ce qui est propre à l'informatique est qu'elle a fait de l'invention de symboles une industrie, et qu'elle formalise des calculs en tant que tels, et pas seulement leurs résultats.

Lire Turing est alors très éclairant. Il utilise autant de symboles qu'il veut. Il se passe complètement de système de numération pour définir ses différentes machines ; quand elles doivent compter, c'est avec des bâtons en répétant une marque autant de fois que nécessaire. Et ses premières machines manipulent soit des **programmes**, soit des **formules logiques** ! Bien sûr, le prétexte premier de son article est d'énumérer des nombres (*computable numbers*), mais ce n'est pas pour cela qu'on se souvient de son article ! Les origines du binaire, digital, numérique ne sont vraiment pas à aller chercher chez Turing, mais plutôt chez **Claude Shannon** et dans l'histoire des premiers ordinateurs, mais **l'informatique n'est pas plus la science des ordinateurs que l'astronomie n'est celle des télescopes** (Michael Fellows et Ian Parberry, 1993).

Alan Turing

Alan Mathison Turing est né en 1912 (Londres) et mort en 1954 (Wilmslow). Paradoxalement, c'est pour ses activités les moins publiques qu'il est le plus connu. Pendant la seconde guerre mondiale, il a participé à l'effort de guerre britannique en contribuant à l'analyse et au décodage des messages que la marine allemande chiffrait à l'aide de la machine ***Enigma***. Cet aspect de sa vie était couvert par le secret et n'a été déclassifié que vers les années 1970-80, mais plusieurs hommages lui ont été rendus pour son rôle éminent dans l'accélération de la fin de la guerre. Il est aussi connu pour sa vie privée et en particulier pour son homosexualité, qui était considérée comme un crime à l'époque. Il sera condamné en 1952 à une « castration chimique ». En 1954, il est retrouvé mort chez lui, des suites d'un suicide. En 2009, le gouvernement britannique reconnaîtra la brutalité et l'injustice du traitement imposé à Alan Turing, qui sera gracié à titre posthume en 2013.

Cependant, la facette de sa vie qui nous intéresse le plus ici est sa contribution majeure à la science informatique, et cette facette est complètement publique. Le modèle de calcul qu'il propose en 1936 pour répondre à une question de logique mathématique est toujours pertinent, et les premières conséquences que Turing en tire sont toujours des résultats phares de l'informatique théorique. À partir de la fin des années 1940, il participera aux recherches sur les premiers ordinateurs. Il se déclarera lui-même constamment surpris de ce qu'il est possible de leur faire faire, et cela le conduira à formuler la première idée d'**intelligence artificielle**. Il aura donc à la fois démontré formellement que **des tâches bien formalisées ne sont pas réalisables automatiquement**, et fait le pari que **des tâches mal formalisées, mais réputées intelligentes, le seront !**

③ Les réductions du *Entscheidungsproblem*

Sitôt posé le problème de la décision du calcul des prédicats (le *Entscheidungsproblem*), on a essayé de le réduire à des problèmes plus simples. Ce programme de recherche avait commencé avant la formalisation par Hilbert de ce problème, car celle-ci ne faisait que formaliser complètement la logique du 1^{er} ordre qui était dans l'air du temps depuis Frege en 1879. Et il a été poursuivi bien après le résultat négatif de Turing, car si celui-ci démontre qu'il n'existe pas de méthode complètement automatique pour décider tout le calcul des prédicats, cela n'empêche pas de rêver d'une méthode qui serait semi-automatique, ou d'une méthode automatique qui déciderait d'une partie seulement du calcul des prédicats, ou d'une méthode automatique qui saurait répondre **oui**, **non** ou **je-sais-pas** (le moins souvent possible).

En 1915-1920, **Leopold Löwenheim** puis **Thoralf Skolem** démontrent que si une formule du calcul des prédicats est satisfaisable dans un domaine infini, elle l'est pour toutes les cardinalités de l'infini. En particulier, elle l'est dans un domaine au plus dénombrable, c-à-d. pas plus gros que **N**.

En 1930, **Jacques Herbrand** montre qu'une formule du calcul des prédicats est satisfaisable si et seulement si elle l'est dans un domaine dénombrable qui a une certaine forme, facile à énumérer, appelé maintenant son **domaine de Herbrand**. En conséquence, une formule insatisfaisable dans son domaine de Herbrand est insatisfaisable tout court, et si **¬F** n'est pas satisfaisable dans son domaine de Herbrand alors **F** est une tautologie.

En 1965, **Alan Robinson** montre une transformation effective des formules du calcul des prédicats en des formules assez proches de celles du calcul des propositions qui permet d'appliquer une procédure automatique qui sait détecter si une formule n'est pas satisfaisable dans le domaine de Herbrand, et donc dans aucun modèle. Cette procédure, appelée **résolution**, peut ne pas détecter qu'une formule est vraie pour le modèle de Herbrand. On appelle cela un **semi-algorithme**. Cela illustre qu'un résultat d'infaissabilité, comme celui de Turing, ne doit pas être interprété comme une catastrophe absolue, mais plutôt comme un challenge pour approcher au plus près ce qu'il est impossible de faire.

① Calcul des propositions (ou logique des propositions)

Le **calcul des propositions** est une logique d'énoncés élémentaires auxquels on peut attribuer une valeur de vérité, **Vrai** ou **Faux**, et qu'on peut articuler en utilisant des **connecteurs logiques** qui jouent le rôle des **conjonctions de coordinations** en français pour former des énoncés complexes dont on peut calculer à leur tour la valeur de vérité.

Variables propositionnelles (ou **atomes**) : Ce sont les formules élémentaires. Dans la suite, on les note par des lettres minuscules, ex. **a**, **b**, **c**, mais absolument rien ne l'exige.

Formules propositionnelles : Ce sont des formules, atomiques ou non, qui sont reliées par des connecteurs logiques, généralement **∧**, **∨**, **¬**, ou **⇒**.

- conjonction**, **ϕ₁ ∧ ϕ₂** : relie deux formules pour en former une troisième qui vaut **Vrai** ssi les deux premières le sont.
- disjonction**, **ϕ₁ ∨ ϕ₂** : relie deux formules pour en former une autre qui vaut **Vrai** ssi au moins une des deux premières l'est.
- implication**, **ϕ₁ ⇒ ϕ₂** : relie deux formules pour en former une autre qui vaut **Faux** ssi **ϕ₁** vaut **Vrai** alors que **ϕ₂** vaut **Faux**.
- négation**, **¬ ϕ** : constitue une formule qui vaut **Faux** ssi **ϕ** vaut **Vrai**.

Tautologie et satisfaisabilité : les formules propositionnelles peuvent être vues comme des fonctions des variables qu'elles contiennent : **{Vrai, Faux}ⁿ → {Vrai, Faux}** pour une formule à **n** variables. Même si il existe une infinité de formules à **n** variables, il n'y a que **2^(2ⁿ)** fonctions différentes, et on peut représenter chacune d'entre elles par une **table de vérité**. C'est un tableau à **2ⁿ lignes**, et **n+1 colonnes**. Ex. **(a∨b)∧c**, **a∨¬a**, et **a∧¬a** sont représentées par les tables de vérité ci-contre. Dans leurs **n premières colonnes**, les lignes représentent toutes les entrées possibles et dans leur **n+1-ème colonne**, elles représentent ce que vaut la fonction pour chaque entrée.

On voit que toutes les lignes de la table de **a∨¬a** ont la valeur **Vrai** en position de résultat. Cette formule est une **tautologie** ; elle vaut **Vrai** pour toutes ses entrées, c-à-d. pour toutes les interprétations de ses variables. Noter que **a∨¬a** est tautologique sans qu'on sache qui de **a** ou de **¬a** est **Vrai**. Cela peut être jugé problématique et il faudra un autre cadre logique pour traiter cela : la logique **intuitionniste**. Toutes les lignes de la table de **a∧¬a** ont la valeur **Faux**. Cette formule est **contradictoire** ou **absurde**. Enfin, des lignes de la table de **(a∨b)∧c** peuvent avoir **Faux** mais au moins une a **Vrai**. Cette formule est **satisfaisable**. Noter que, pour tout **ϕ**, **ϕ** ou **¬ϕ** est satisfaisable, que **ϕ** et **¬ϕ** peuvent l'être toutes deux (ex. **(a∨b)∧c**), et que **ϕ** est tautologique ssi **¬ϕ** n'est pas satisfaisable (= est contradictoire).

② Calcul des prédicats (ou logique du 1^{er} ordre)

Le **calcul des prédicats** permet d'exprimer des **jugements** (ex. **∀X,Y,Z. X>Y ∧ Y>Z ⇒ X>Z** et **∀X,Y. père(X,Y) ∧avare(X) ⇒ prodigue(Y)**), et de décider formellement si ils sont **tautologiques** (c-à-d. vrais pour toutes les interprétations des symboles) ou non. Le calcul des prédicats n'a commencé à être formalisé qu'à la fin du XIX^e siècle, par **Gottlob Frege**, puis au début du XX^e siècle avec **Alfred Whitehead** et **Bertrand Russell**, **David Hilbert** et **Wilhelm Ackermann**, et **Kurt Gödel**. Les formules du calcul des prédicats sont formées d'énoncés élémentaires constitués d'un **prédicat**, c-à-d. un jugement (ex. **être avare**) appliqué à un ou plusieurs sujets, des connecteurs du calcul des propositions (ex. **∧**, **∨**, **¬**, et **⇒**) et des quantificateurs **∀** et **∃**. On constitue des ensembles de formules qu'on appelle **théories**. À proprement parler, la tautologie d'une formule s'évalue par rapport à une théorie, qui va par exemple énoncer les propriétés logiques de **<**, **père**, **avare** et **prodigue**. On veut savoir si une formule est une **conséquence logique** d'une théorie et pas seulement une conséquence d'une certaine interprétation des symboles.

Formules quantifiées : Ce sont des formules dont la valeur de vérité s'évalue par rapport à un ensemble d'objets, un **domaine**, plutôt que par rapport à un objet. Syntactiquement, une quantification lie une variable, comme le font **∫… dx** ou **∂…/∂x**. Sémantiquement, les quantifications sont définies comme suit :

- quantification universelle**, **∀x . ϕ (x)** : constitue une formule qui vaut **Vrai** ssi **ϕ (e)** vaut **Vrai** pour tout élément **e** du domaine. Si le domaine est fini, la quantification universelle est juste une **conjonction** des applications de **ϕ** à tous les éléments du domaine.
- quantification existentielle**, **∃x . ϕ (x)** : constitue une formule qui vaut **Vrai** ssi **ϕ (e)** vaut **Vrai** pour au moins un élément **e** du domaine. Si le domaine est fini, la quantification existentielle est juste une **disjonction** des applications de **ϕ** à tous les éléments du domaine.

Démonstration automatique dans la logique des prédicats : dès le début du XX^e siècle, on s'est demandé si il était possible de décider qu'une formule du calcul des prédicats était une tautologie en utilisant une méthode **purement algorithmique**, c-à-d. systématique, qui finit toujours avec un résultat décisif (c-à-d. **oui** ou **non**, mais pas **je-sais-pas**), et sans nécessité d'un jugement externe. On appelle cela une **procédure de décision**. Concernant la partie propositionnelle, on sait qu'une telle procédure existe, même si elle est très coûteuse dans le pire des cas (mais alors, on parle de **complexité algorithmique** et pas de **calculabilité**). Mais concernant les quantifications, la difficulté réside dans des conditions comme **Vrai pour tous les éléments du domaine** ou **Vrai pour au moins un élément du domaine**, et ce pour tous les domaines, y-compris ceux qui ne sont pas finis. Comment explore-t-on un domaine infini en un nombre d'opérations fini ? Comment le faire pour tous les domaines ? Voir le bloc ③ de ce document.

N'oubliez jamais !

Un **modèle** est toujours **imparfait**, parfois **utile**, et c'est tout ce qu'on peut lui demander (d'après George Box). Les **programmes** sont des modèles d'une forme de réalité.

L'informatique ne peut travailler que sur la **représentation** des choses. Ex., la machine de Turing universelle opère sur une représentation des programmes. Mais il faut des capteurs et des actionneurs pour toucher les vraies choses. Par contre, opérer sur des représentations de représentations de … est courant. Ex., **440** est une représentation d'un entier qui peut être la représentation de la mesure d'une fréquence, tout comme **1B8**, et **La₃** pourrait être une autre représentation de la même fréquence. Il faut toujours se demander quelle représentation a du sens par rapport à ce **pourquoi** est utilisé le modèle.

	Modèles de programme et de ruban à photocopier			
	ÉTAT DEPART	ÉTAT ARRIVÉE	SYMBOLE ÉCRIT	SENS DÉPLACEMENT
1		règle pour le symbole b		
		règle pour le symbole 0		
		règle pour le symbole 1		
2		règle pour le symbole b		
		règle pour le symbole 0		
		règle pour le symbole 1		
3		règle pour le symbole b		
		règle pour le symbole 0		
		règle pour le symbole 1		
4		règle pour le symbole b		
		règle pour le symbole 0		
		règle pour le symbole 1		
5		règle pour le symbole b		
		règle pour le symbole 0		
		règle pour le symbole 1		
6		règle pour le symbole b		
		règle pour le symbole 0		
		règle pour le symbole 1		
7		règle pour le symbole b		
		règle pour le symbole 0		
		règle pour le symbole 1		
8		règle pour le symbole b		
		règle pour le symbole 0		
		règle pour le symbole 1		
9		règle pour le symbole b		
		règle pour le symbole 0		
		règle pour le symbole 1		
10		règle pour le symbole b		
		règle pour le symbole 0		
		règle pour le symbole 1		
11		règle pour le symbole b		
		règle pour le symbole 0		
		règle pour le symbole 1		
12		règle pour le symbole b		
		règle pour le symbole 0		
		règle pour le symbole 1		

Le *Entscheidungsproblem*

En 1900, **David Hilbert** présente au congrès international de mathématiques ses vues sur ce que seront les grands challenges mathématiques du XX^e siècle. Il dresse alors une liste de 23 problèmes, qu'on appelle désormais **problèmes de Hilbert**. Certains ont pu être résolus au cours du XX^e siècle au prix d'un travail important, ex. en 1970, la non-décidabilité de la résolution des **équations diophantiennes** (équations polynomiales en nombre entier, 10^e problème de Hilbert) par **Youri Matyasevich** en s'appuyant sur des résultats de **Julia Robinson**. D'autres étaient si difficiles qu'ils sont devenus des problèmes du XXI^e siècle, et ce sont vus dotés d'un prix de 1 000 000 \$ pour qui les résoudreait. D'autres étaient tout simplement mal posés, et n'ont pas été de vrais challenges !

En 1928, Hilbert et **Wilhelm Ackermann** proposent dans leur **Principes de logique mathématique** une formalisation du calcul des prédicats qui pourrait servir d'intermédiaire pour le 2nd problème, la non-contradiction des axiomes de l'arithmétique. Se pose alors la question de l'automatisation de la démonstration dans le calcul des prédicats : le ***Entscheidungsproblem***. Au début, il allait de soi pour Hilbert que la question n'était pas de savoir si c'était possible, mais par quel moyen : ***Wir müssen wissen, Wir werden wissen***.