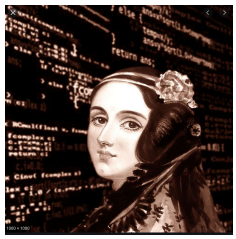
Notion de programme en tant que donnée. Calculabilité et décidabilité

Table des matières

1) Quelques repères historiques

- Wilhelm Schickard (1592 – 1635) , professeur à l'Université de Tübingen (Allemagne), aurait dessiné les plans de la première machine à calculer (mécanique). Cette machine n'a pas été construite.
- Blaise Pascal (1623 – 1662) , mathématicien et philosophe, construit à l'âge de 19 ans la Pascaline, première machine à calculer opérationnelle du XVIIè siècle .
- Gottfried Wilhelm Leibniz (1646 – 1716), mathématicien et philosophe, développe aussi une machine à calculer. Il préconise des idées très modernes : la machine de calcul universelle, le schéma "entrée-calcul-sortie", la base 2 pour la représentation des nombres.
- Le métier à tisser de Joseph Marie Jacquard (1752 – 1834) est basé sur l'utilisation de cartes perforées, et est à l'origine des premiers programmes de calcul.
- Charles Babbage (1791 – 1871), professeur à Cambridge, construit la machine différentielle et imagine les plans de la machine analytique (machine programmable). La dernière peut être considérée comme précurseur des ordinateurs modernes, consistant d'une unité de contrôle, une unité de calcul, une mémoire, ainsi que l'entrée-sortie.
- Ada Lovelace (1815 - 1852) , est une pionnière de la science informatique. Elle est principalement connue pour avoir réalisé le premier véritable programme informatique, lors de son travail sur un ancêtre de l'ordinateur : la machine analytique de Charles Babbage. Dans ses notes, on trouve en effet le premier programme publié, destiné à être exécuté par une machine, ce qui fait considérer Ada Lovelace comme le premier programmeur du monde. Elle a également entrevu et décrit certaines possibilités offertes par les calculateurs universels, allant bien au-delà du calcul numérique et de ce qu'imaginaient Babbage et ses contemporains. Elle est assez connue dans les pays anglo-saxons et en Allemagne, notamment dans les milieux féministes ; elle est moins connue en France, mais de nombreux développeurs connaissent le langage Ada, nommé en son honneur.

Blaise Pascal



Ada Lovelace

- David Hilbert (1862 – 1943) , mathématicien allemand. En 1900, Hilbert propose 23 problèmes dont certains ne sont pas résolus à ce jour. Pour voir la liste des problèmes. Il présente en 1920 un programme de recherche visant à clarifier les fondements des mathématiques : "tout énoncé mathématique peut être soit prouvé ou réfuté". Plus tard il énonce le "Entscheidungsproblem" : montrer de façon "mécanique" si un énoncé mathématique est vrai ou faux. Il faudra attendre 1936 pour qu'Alan Turing s'intéresse au problème n°10 avec Church (dont il était le doctorant). Ils définissent rigoureusement la notion d'algorithme.

— Kurt Gödel (1906 – 1978) , un des logiciens les plus fameux de l'histoire, répond en 1931 négativement quant au programme proposé par Hilbert, en montrant que tout système formel suffisamment puissant est soit incomplet ou incohérent. Il montre ceci en construisant une formule qui exprime le fait qu'elle n'est pas démontrable.


David Hilbert


Kurt Gödel


Alan Turing

— Alan Turing (1912 – 1954) et Alonzo Church (1903 – 1995) montrent indépendamment, en 1936, l'indécidabilité de l'Entscheidungsproblem. Turing propose "la machine de Turing" comme modèle mathématique de calcul, et Church le lambda calcul. Ils énoncent le principe selon lequel tout ce qui est calculable peut être calculé sur un de ces deux modèles ("thèse de Church-Turing"). La Machine de Turing est inventée pour répondre au problème mathématiques de la décidabilité proposé par Hilbert. Une machine de Turing a pour but de décrire les algorithmes. Il faut savoir que Turing ne verra pas de son vivant une réalisation concrète de sa "machine".
Pour plus d'information sur la vie d'Alan Turing, voir cette vidéo.

2) Machine de Turing

2.1) Définition

Il s'agit d'une machine imaginaire inventée par Alan Turing en 1936 pour expliquer la notion de "procédure mécanique" : l'équivalent de d'un algorithme. Cette machine est la plus élémentaire possible destinée à mettre en œuvre ces mécanismes de calcul, numériques ou symboliques, comme le font notamment les ordinateurs.

Une machine de Turing est un appareil disposant :

— d'un ruban infiniment long divisé en cases, dans lesquels la machine peut écrire des symboles ;
— une tête qui peut lire et écrire sur ce ruban ;
— une table de transition. Chacune des lignes de cette table est associée à un état et spécifie les actions à effectuer quand la machine est dans cet état, en fonction du symbole présent sous la tête de lecture. Ces actions peuvent être l'écriture d'un symbole (ici un 0 ou un 1) et le déplacement du ruban d'une case à droite ou à gauche. La table spécifie également le nouvel état après exécution de ces actions. La machine s'arrête quand un état marqué comme final est atteint.

Cette machine fonctionne comme une machine à calculer en binaire. Turing envisage le cas particulier où les symboles utilisés sont 0 et 1.

L'entrée du programme est un liste de symboles binaires. Une fois effectué, c'est sur ce même ruban que sera écrit la sortie du programme.

Remarque 1 : En 2012, pour célébrer le centenaire de la naissance d'Alan Turing, huit étudiant en master de l'École Normale Supérieure (ENS) de Lyon ont fabriqué en LEGO la première machine réelle (purement mécanique) du modèle de Turing. Voici le site du projet

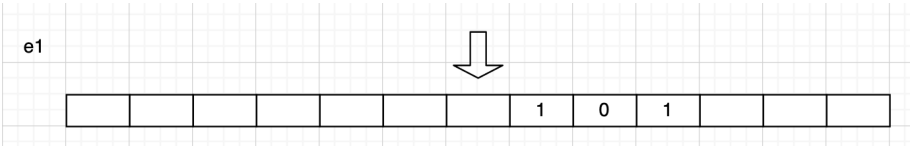
2.2) Exemple

La machine peut être dans deux états e1 et e2. Si la machine est dans l'état e1 et contient un blanc, elle écrit un blanc. Le rubanc se déplace vers la gauche et la machine est dans l'état e2. Si la machine est dans l'état e2 et qu'elle lit un 0, elle écrit un 1 et si elle lit un 1 elle écrit un 0. Dans les deux cas, le ruban se déplace vers la gauche et la machine reste dans l'état e2. Enfin, si elle lit un blanc, elle écrit un blanc et la machine s'arrête.

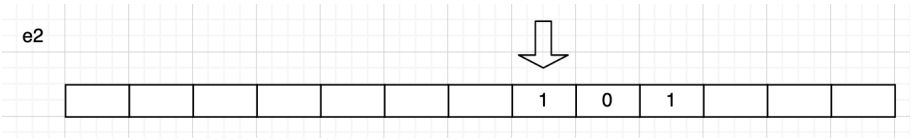
Ce qui nous donne la table de transition suivante :

État	Lecture	Écriture	Déplacement	État suivant
e1	blanc	blanc	gauche	e2
e2	0	1	gauche	e2
e2	1	0	gauche	e2
e2	blanc	blanc	gauche	Fin

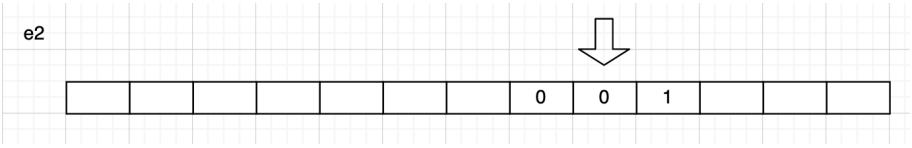
Les étapes successives sont représentées ci-dessous. Initialement la machine se trouve dans l'état e1.



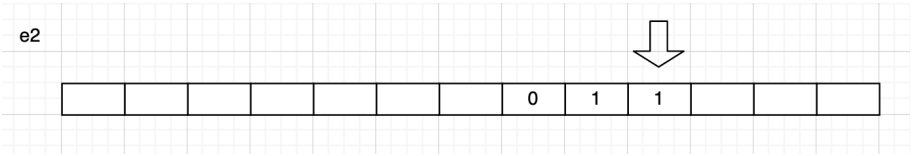
La machine lit un blanc, passe dans l'état e2 et le ruban se déplace vers la gauche.



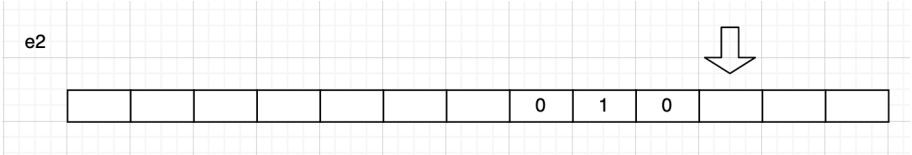
La machine lit un 1, donc écrit un 0, reste dans l'état e2 et ruban se déplace vers la gauche



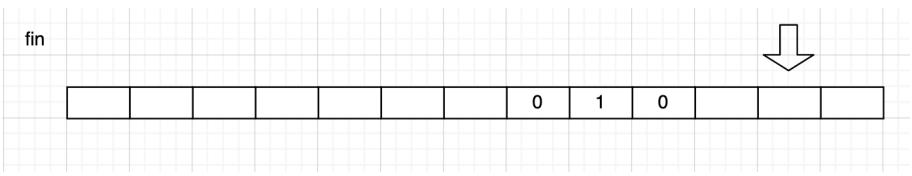
La machine lit un 0, donc écrit un 1, reste dans l'état e2 et le ruban se déplace vers la gauche.



La machine lit un 1 donc écrit un 0, reste toujours dans l'état e2 et le ruban se déplace vers la gauche.



La machine lit un blanc, écrit un blanc, le ruban se déplace encore vers la gauche et la machine s'arrête.



Remarque 2 : Voici un émulateur de cette machine de Turing

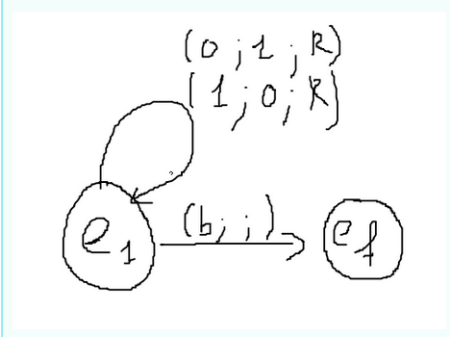
2.3) Diagramme

On peut construire un diagramme de la machine (on parle aussi d'automate).
Pour définir un diagramme, il faut :
— un alphabet utilisé par la machine (liste des caractères utilisés). Exemple '0','1','b'.
— des états sont symbolisés par des cercles et écrits sous la forme e1, e2, etc (ou encore q1, q2, etc).
— un état final représenté par ef (ou qf) mais parfois indiqué par une chaine de caractères ou un caractère particulier ("final").
— représenter les déplacements par les lettres L et R (left et right).
— utiliser des triplets (lecture ; écriture ; déplacement) et des flèches qui indiquent l'état suivant.

Exemple 1 :
Liste des caractères possiblement présents sur le ruban $\{'0';'1';'b'\}$
— A l'état $e1$, si la tête lit un 0 alors elle écrit 1, se déplace à droite et reste à l'état e1.
— A l'état e1, si la tête lit un 1 alors elle écrit 0, se déplace à droite et reste à l'état e1.
— A l'état e1, si la tête lit un blanc alors elle n'écrit rien et passe à l'état ef (elle s'arrête).
Ce qui nous donne la table de transition suivante :

État	Lecture	Écriture	Déplacement	État suivant
e1	0	1	droite	e1
e1	1	0	droite	e1
e1	blanc	rien	droite	ef

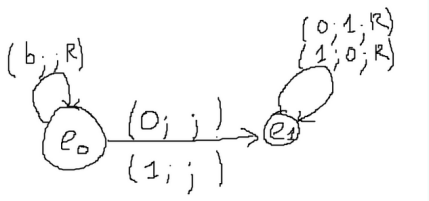
Et donc le diagramme suivant :



Exemple 2 :
Liste des caractères possiblement présents sur le ruban $\{'0';'1';'b'\}$
— si la tête lit un blanc alors elle n'écrit rien et se déplace à droite en restant à l'état e0.
— A l'état e0, si la tête lit un 0 alors elle n'écrit rien, ne se déplace pas et passe à l'état e1.
— A l'état e0, si la tête lit un 1 alors elle n'écrit rien, ne se déplace pas et passe à l'état e1.
— A l'état e1, si la tête lit un 0 alors elle écrit 1, se déplace à droite et reste à l'état e1.
— A l'état e1, si la tête lit un 1 alors elle écrit 0, se déplace à droite et reste à l'état e1.
Ce qui nous donne la table de transition suivante :

État	Lecture	Écriture	Déplacement	État suivant
e0	b	rien	droite	e0
e0	0	rien	pas de dplt	e1
e0	1	rien	pas de dplt	e1
e1	0	1	droite	e1
e1	1	0	droite	e1

Et donc le diagramme suivant :



3) Calculabilité et décidabilité

3.1) Calculabilité

Une fonction **calculable** est une fonction que l'on peut écrire sous forme d'algorithme.

Quelques remarques : Les machines de Turing sont une réponse à la recherche de définition d'une fonction calculable.

Toute fonction calculable est calculable grâce à une machine de Turing. C'est la thèse de Church : "Toute fonction physiquement calculable est calculable par une machine de Turing". Les fonctions calculables par d'autres modèles de calculs suffisamment riches (machines à registres, machine RAM, lambda calcul...) sont les mêmes que les fonctions calculables par des machines de Turing.

Il existe même une machine de Turing universelle qui simule toutes les autres machines de Turing. Les machines de Turing sont en fait des modèles universels de calcul.

Fonctions non-calculables : contrairement à ce que l'on pourrait imaginer, il existe des fonctions qui ne sont pas calculables. Il y en a même une infinité.

Remarque : La calculabilité ne dépend pas du langage utilisé.

Si une fonction est calculable, il existe un algorithme dans le langage que vous utilisez qui permet de la calculer.

3.2) Décidabilité

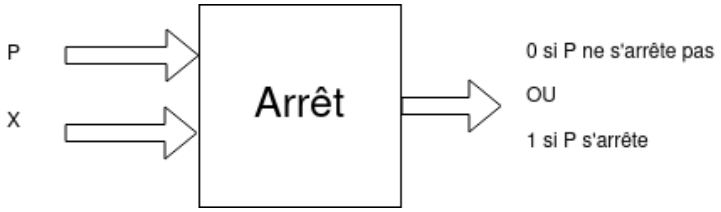
Une propriété est décidable s'il existe un algorithme qui la confirme ou l'infirme en un nombre fini d'étapes. L'algorithme doit répondre par oui ou par non à la question posée par le problème. On parle dans ce cas de problème de décision.

3.3) Le problème de l'arrêt

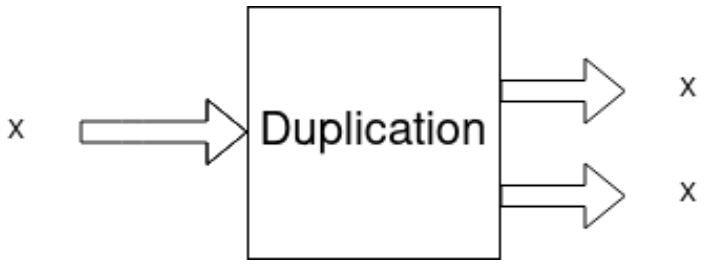
Il nous est arrivé à tous d'écrire une boucle infinie. C'est le drame car notre programme ne s'arrête jamais. A l'échelle de l'industrie ce problème devient un problème essentiel. Pensez aux problèmes de sécurité (centrale nucléaire, aviation, transports, etc).

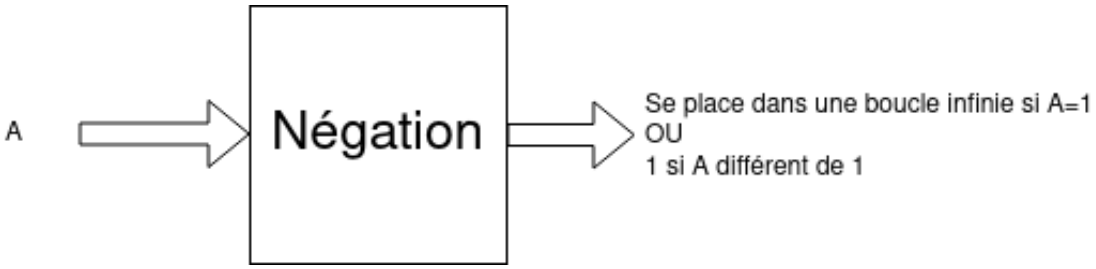
Le problème de l'arrêt est un problème classique en décidabilité. Il fait partie des problèmes qui n'ont pas de solution. Autrement dit, il est impossible d'écrire un algorithme qui résout ce problème. La plupart des autres problèmes indécidables reviennent à ce problème.

Nous supposons l'existence d'un programme, appelé "Arrêt", ayant deux entrées : le première représente le programme P et la seconde représente une donnée X du programme P (X est une donnée utilisée par le programme P). Le programme "Arrêt" détermine sans jamais se tromper si l'exécution du programme P s'arrête ou pas. Il donnera la valeur 0 si le programme P ne s'arrête pas et la valeur 1 si il s'arrête.

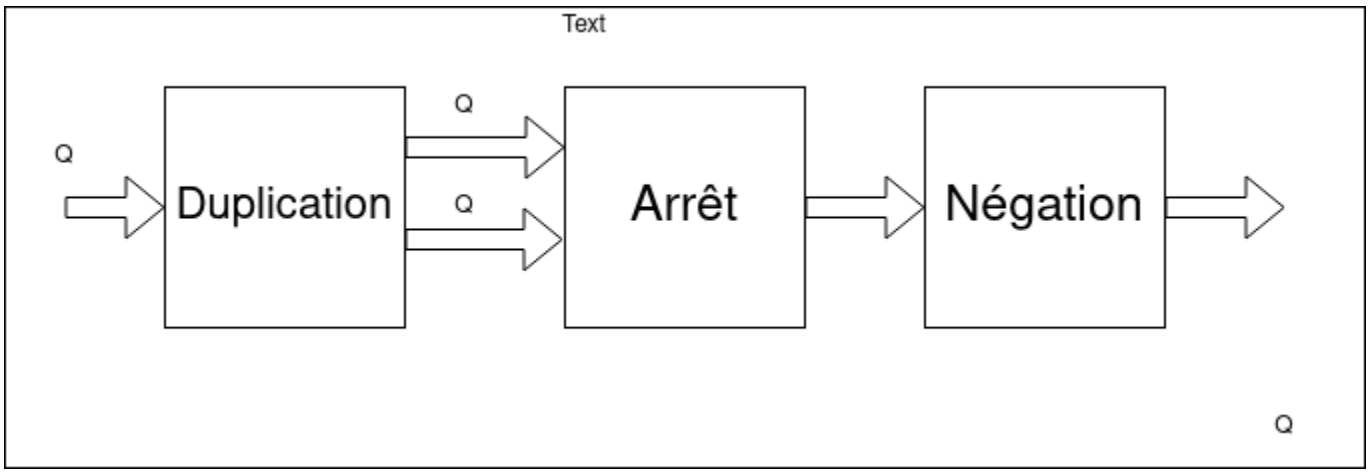


Nous pouvons alors construire un autre programme que l'on appelle Q constitué de 3 programmes : le premier programme est l programme "Duplication", le deuxième est le programme "Arrêt" t le troisième est le programme "Négation".





L'idée est d'exécuter le programme Q avec comme entrée le programme Q lui-même. Cela revient à la situation suivante :



Il faut alors distinguer 2 cas :

— En sortie du programme "Arrêt", on a la sortie 1 si Q s'arrête. Ce qui nous amène en sortie du programme Q une situation où l'on se trouve dans une boucle infinie, ce qui veut dire que Q ne s'arrête pas. Ce qui est contradictoire.

— En sortie du programme "Arrêt", on a la sortie 0 si le programme Q ne s'arrête pas. Ce qui nous amène à la sortie du programme Q le résultat 1, ce qui signifie que Q s'arrête. Ce qui est là aussi contradictoire.

Conclusion : Nous aboutissons dans les deux cas à une contradiction. Ce qui montre l'impossibilité de l'existence du programme "Arrêt". Il n'existe donc pas de programme qui détermine d'une façon générale si un programme donné s'arrête se termine sur une entrée donnée.