La machine de Turing

Pour comprendre plus précisément ce qu'est un algorithme, Turing regarde ce que devrait faire un calculateur qui exécute un algorithme. Cela lui donne une idée des instructions que la machine doit recevoir, et donc de ce qu'est un programme.

Pour cela il imagine que le calculateur est un humain avec une feuille de papier, un cerveau, des yeux, une main avec un crayon et une gomme. Le papier sert à la fois à poser le problème, à écrire la réponse et comme brouillon pour écrire les calculs intermédiaires. La donnée du problème est un nombre fini de symboles sur la feuille de papier.

Prenons un exemple précis : une addition

Exercice : calculer 123+789
Ou bien la version problème de décision : est-ce que 123+789=812 ?

Les yeux et la main se déplacent sur la feuille pour faire le calcul en suivant les instructions.

Celles-ci sont de la forme :
État du cerveau, symbole lu |-----> état du cerveau, écriture d'un symbole, déplacement des yeux et de la main

On va maintenant définir ce qu'est une machine de Turing :

L'espace de travail est un ruban bi-infini vers la droite et la gauche avec des cases indexées par les entiers relatifs Z
L'espace des états internes de la machine, qui correspond à l'ensemble des états du cerveau, est un ensemble fini Q
Parmi ces états nous avons des états q_0 (état initial), q_a (état qui accepte, répond oui à la question posée) et état q_r qui rejette (répond non à la question posée)
Sigma est un ensemble fini appelé alphabet qui sert à exprimer la donnée (123+789 dans le cas de l'addition, donc des chiffres et le symbole +)
Gamma est l'alphabet de travail qui contient Sigma ainsi qu'un symbole B pour les cases blanches. Il est fini. Dans notre exemple il contient par exemple les chiffres marqués par la retenue
Nous avons enfin une fonction de transition delta, qui est une fonction de Q(privé de q_a, q_r)xGamma dans QxGammax{G,S,D}. C'est un ensemble fini de règles de transition.

Formellement, une machine de Turing est un octuplet (Σ, Γ ,B, Q, q0 , qa , qr ,δ) avec les propriétés précédentes. Remarquez que c'est un objet combinatoire fini, qu'on peut coder avec un mot booléen. (voir par exemple le livre de Sylvain Perifel)

Qu'est-ce qu'un calcul d'une machine de Turing sur un mot d'entrée m ? C'est une suite de configurations finies telles que la première est la configuration initiale avec le mot m et on peut passer d'une configuration à la suivante par une application de la fonction de transition.

Plus en détail : Dans la configuration initiale, le ruban contient le mot d'entrée m (une lettre par case en commençant par la première), toutes les autres cases contiennent le symbole blanc B, l'état de la machine est q_0 et la tête de lecture écriture pointe sur la première case.

On effectue ensuite un pas de calcul. Par exemple si delta(q_0,m_1)=(q_1,0,D) on a une nouvelle configuration.

Le calcul s'arrête quand on est dans l'état q_a ou q_r. Il peut ne jamais s'arrêter !
On donne un exemple.

Si le calcul s'arrête, on dit que la machine accepte le mot si l'état final est q_a, qu'elle le rejette si l'état final est q_r. Le langage reconnu par la machine est l'ensemble des mots acceptés. Le problème (ou langage) est décidable s'il existe une machine dont tous les calculs s'arrêtent et qui reconnaît ce langage.

Exemple : l'ensemble des mots sur {0,1} qui ne contiennent que des 0 est décidable.

Exercice : Donner une machine qui reconnaît le langage des mots formés de 0 suivi de 1

Des remarques :

- Si on veut un résultat et pas juste accepté / refusé, comme par exemple un résultat d'addition, on peut définir le résultat du calcul comme le contenu du ruban à la fin du calcul
- On peut définir la complexité du calcul : le temps de calcul est le nombre de pas de calcul, ie d'application de la règle de transition. L'espace de calcul est le nombre de cases utilisées lors du calcul
- La notion de machine de Turing est très robuste et on peut changer le nombre de rubans utilisés, ou limiter la taille de l'alphabet
- Ce qui est remarquable, c'est que les ordinateurs actuels sont encore très proches de ce modèle théorique, qui a été pensé alors qu'on n'avait aucune idée des moyens techniques pour en fabriquer en version physique !
- Une des différence avec les ordinateurs actuels est l'accès par adresse de la mémoire. Sur un ruban de papier, on se déplace de proche en proche. C'est différent d'une RAM (random access memory) où on peut accéder à la mémoire par adresse.

Une notion très importante pour les machines de Turing : la machine de Turing universelle

Pour l'instant on a vu comment définir une machine pour un problème donné. Mais on sait bien qu'en pratique ce n'est pas ce qu'on veut ! On ne veut pas définir un ordinateur par problème, mais on veut programmer notre ordinateur. Autrement dit, on veut pouvoir entrer le programme comme une donnée.
Nous avons vu qu'une machine de Turing est un objet combinatoire fini, on peut donc le coder par des mots booléens.

Théorème : Il existe une machine de Turing universelle U, c'est à dire une machine qui, avec pour entrée le code c d'une machine M et un mot m, simule le calcul de c sur l'entrée m (le calcul ne s'arrête pas, l'état final est q_a ou l'état final est q_r)

Remarque : c'est ce qu'on appelle exécuter un programme. Cette simulation d'une machine par une autre est ce qui permet d'utiliser un programme comme sous-programme d'un autre.
Oublié dans la vidéo : Cette machine de Turing universelle est un interpréteur de machines de Turing

Revenons à ce dont nous avons parlé dans l'introduction du cours. Nous avons vu que les machines de Turing étaient une des façons de définir ce qui est calculable, avec aussi par exemple le lambda-calcul ou les fonctions récursives. Mais votre langage de programmation préféré est probablement aussi Turing complet, c'est à dire qu'il permet de calculer les mêmes fonctions. Ce n'est pas le cas pour des langages trop simples, comme celui pour programmer le thermostat de votre chauffage, ou votre lave-linge. Mais nos ordinateurs, si nous imaginons que leur mémoire peut être augmentée à l'infini, sont des machines universelles que nous pouvons programmer. Pour vous convaincre que votre langage préféré est Turing-complet, vous pouvez le programmer pour simuler le calcul d'une machine de Turing universelle.

Pour en savoir plus sur les machines de Turing, je vous recommande le chapitre 1 du très bon livre de Sylvain Perifel qui est disponible gratuitement en ligne.

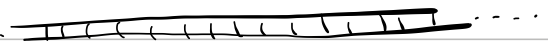
La machine de Turing

calculer
$$\begin{array}{r} {\scriptstyle 1\ 1}\\ \sqcup 1\ 2\ 3\\ +\sqcup 7\ 8\ 9\\ \hline 9\ 1\ 2 \end{array}$$

est-ce que $123+789=812$? non

forme des instructions :
état du cerveau, symbole lu → état du cerveau, écriture d'un symbole, déplacement

Définition d'une machine de Turing :

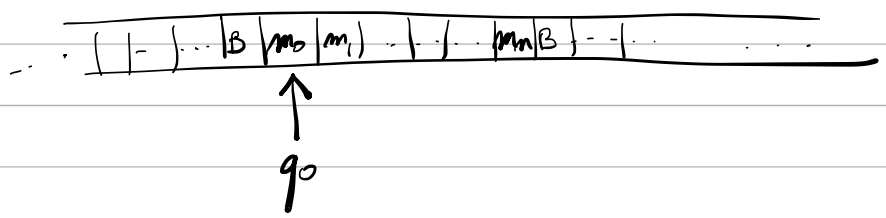
l'espace de travail : ruban bi-infini 
cases indexées par $\mathbb{Z}$

Q fini : les états internes de la machine
$q_0 \in Q$: l'état initial
$q_a \in Q$: l'état qui accepte
$q_r \in Q$: l'état qui rejette
Σ ensemble fini : l'alphabet pour exprimer la donnée
$\Gamma \supseteq \Sigma \cup \{B\}$ fini : l'alphabet de travail
δ fonction de transition : $Q \setminus \{q_a, q_r\} \times \Gamma \to Q \times \Gamma \times \{G, S, D\}$

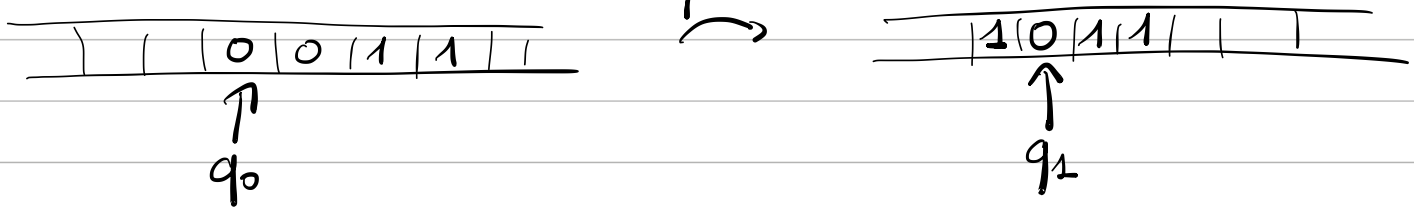
Une machine de Turing est un octuplet $(\Sigma, \Gamma, B, Q, q_0, q_a, q_r, \delta)$ avec les propriétés précédentes.

Qu'est-ce que le calcul d'une machine de Turing sur l'entrée m :

configuration initiale



un pas de calcul avec δ
par exemple si $\delta(q_0, 0) = (q_1, 1, D)$



Un calcul est une suite de configurations qui commence par la configuration initiale.
Il se termine quand l'état est q_a ou q_r.

ex de calcul qui ne s'arrête pas

$P = \Sigma = \{0, 1\}$ $\quad Q = \{q_0, q_a, q_r\}$
$\delta : (q_0, 0) \mapsto (q_0, 0, D)$
$\delta : (q_0, 1) \longmapsto (q_0, 1, D)$
$\delta : (q_0, B) \longmapsto (q_0, B, D)$

def si sur l'entrée m la machine s'arrête dans l'état q_a (resp q_r) on dit qu'elle accepte (resp rejette) l'entrée.

Le langage reconnu par la machine est l'ensemble des mots acceptés. Le problème (ou langage) est décidable s'il existe une machine dont tous les calculs s'arrêtent et qui reconnait ce langage.

ex : $L = \{ m \in \{0;1\}^* \mid m = 0^n \text{ pour } n \in \mathbb{N} \}$

δ : $(q_0, 0) \to (q_0, 0, D)$
$(q_0, B) \to (q_a, B, S)$
$(q_0, 1) \to (q_r, 1, S)$

$Q = \{q_0, q_a, q_r\}$

Théorème il existe une machine de Turing universelle U, c'est à dire une machine, qui avec pour entrée le code c d'une machine M et un mot m, simule le calcul de M sur m.