Les problèmes indécidables

Rappel et notations :
Le langage L(M) reconnu par la machine M est l'ensemble des mots acceptés par la machine. Le problème (ou langage) L est décidable s'il existe une machine M dont tous les calculs s'arrêtent et qui reconnaît ce langage : L=L(M).
On note <M> le code de la machine M.

Y a t-il des langages non décidables ?

Par un simple argument de comptage oui, car il n'y a pas assez de machine de Turing. On regarde tous les codes de machines : ce sont des mots booléens, il y en a un nombre dénombrable. De plus tous les mots ne sont pas nécessairement des codes, et plusieurs machines peuvent calculer la même chose.
Combien y a t-il de langages de {0,1}^* ? Il y en a autant que d'éléments dans P(N) car ce sont les sous ensembles de {0,1}^*, et nous avons vu grâce à un argument diagonal que c'est strictement plus que d'éléments dans N.

Donc il existe des langages non décidables. On peut même dire que la plupart le sont car il y en a un nombre non dénombrable !

Comment montrer qu'un langage n'est pas décidable ?

Pour montrer qu'un langage n'est pas décidable, on réduit un langage dont on sait déjà qu'il n'est pas décidable à celui-ci. C'est une réduction dans un sens un peu différent de celui qu'on utilise en complexité. On ne demande pas que la fonction soit calculée efficacement, mais juste qu'elle soit calculable :

Définition : une réduction (calculable) d'un langage F à un langage G est une fonction calculable phi de {0,1}^* dans {0,1}^* telle que pour tout mot booléen x, F(x)=G(phi(x))

Théorème : Soit G un langage booléen. Soit F un langage booléen non décidable. S'il existe phi calculable de {0,1}^* dans {0,1}^* telle que F(x)=G(phi(x)) alors G n'est pas décidable.

Preuve : Si G est décidable, alors on peut décider F(x) en calculant G(phi(x)) !

C'est très similaire au cas des problèmes NP-complet dont vous a parlé Tim, et nous avons également le problème de la poule et de l'oeuf ! Il nous faut trouver un langage non décidable pour amorcer le processus.

Nous allons en construire un appelé le **problème de l'arrêt** :
La donnée : le code <M> d'une machine de Turing et une entrée m pour la machine
La réponse : oui (1) si le calcul M(m) de la machine M sur l'entrée m s'arrête, non (0) sinon

Définition : Problème de l'arrêt H={<<M>,m>| le calcul de la machine M sur l'entrée m s'arrête}

<<M>,m> représente ici le code booléen du couple de mots (<M>,m). On choisit un codage calculable classique pour les couples.

La décidabilité du problème de l'arrêt est une question très naturelle à se poser : quand on écrit un programme pour répondre à une question, la moindre des choses est que le calcul s'arrête en temps fini ! Peut-on vérifier cela algorithmiquement ? Malheureusement (ou heureusement pour les informaticiens !) la réponse est non.

Théorème : H n’est pas décidable

Preuve : Nous allons utiliser un procédé diagonal. Supposons que le problème soit décidable.
On considère la machine M qui sur l’entrée x fait la chose suivante :

- Décide si la machine de code x s’arrête sur l’entrée x
- Si la réponse est oui, alors M boucle, sinon M s’arrête

Cette machine existe. En effet, nous avons supposé H décidable, et grâce aux machines universelles nous pouvons simuler le calcul d’une machine avec une autre machine.
La machine M a un code c. Que se passe t-il si on donne en entrée c à la machine M ? Le calcul s’arrête t-il ? Supposons que le calcul s’arrête. Alors par définition de M, la machine boucle et ne s’arrête pas. De même, si on suppose que le calcul de M sur c ne s’arrête pas, alors par définition la machine s’arrête.

Nous avons donc montré que H est indécidable.

Montrons maintenant une variante du problème de l’arrêt :
H’={<M>| le calcul de la machine M sur l’entrée vide s’arrête}

Ce problème est un sous-problème du précédent, il est donc plus simple. Nous avons une réduction évidente de H’ à H avec la fonction phi(x)=<x,0> mais cela ne nous donne pas le résultat. Pour montrer que H’ est indécidable, il nous faut une réduction dans l’autre sens.

Soit phi la fonction qui à <<M>,m> associe le code <M’> de la machine M’ qui sur une entrée quelconque, simule la machine M sur l’entrée m. On a alors bien <<M>,m> appartient à H ssi <M’>=phi(<<M>,m>) appartient à H’. H’ est donc indécidable.

Problème ouvert

Pour finir, il y a des problèmes pour lesquels dont on ne sait pas encore s’ils sont décidables. Citons par exemple le problème de Pisot : il s’agit de savoir si une suite récurrente linéaire à coefficients entiers a un zéro. La donnée est un entier k>0 (l’ordre de la récurrence), n entiers relatifs qui représentent les conditions initiales u_0, …, u_{k-1} et n+1 entiers relatifs pour les coefficients a_0,...,a_k. Est-ce que la suite récurrente linéaire définie par
a_0 u_n + a_1 u_{n+1}+... +a_k u_{n+k}=0 a un élément nul ? On peut décider si l’ensemble des éléments vide est fini ou infini, mais savoir s’il est vide est encore une question ouverte et très étudiée. On sait cependant que si cet ensemble est décidable sa complexité sera au moins NP-difficile.

Les problèmes indécidables

Notation et définition :

si M est une machine, $L(M)$ langage reconnu par M

$L(M) = \{x \in \{0;1\}^* \ / \text{le calcul de } M \text{ sur } x \text{ accepte}\}$

L est décidable s'il existe M dont tous les calculs s'arrêtent sur toutes les entrées et tq $L = L(M)$

On note $\langle M \rangle$ le code de la machine.

<u>Est-ce qu'il existe des langages décidables ?</u>

Combien de machines de Turing ? un nombre dénombrable donc un nombre dénombrable de langages décidables.

$L \subseteq \{0;1\}^*$ $\quad L \in \mathcal{P}(\{0;1\}^*)$ non dénombrable

<u>Comment montrer qu'un langage n'est pas décidable !</u>

<u>Définition</u> : une réduction calculable d'un langage F à un langage G est une fonction calculable $\varphi : \{0;1\}^* \to \{0;1\}^*$ tq $\forall x \in \{0;1\}^* \quad F(x) = G(\varphi(x))$

<u>Théorème</u> : soit G et F sont des langages, φ une réduction calculable de F à G, si F n'est pas décidable, alors G non plus.

Preuve : $\quad F(x) = G(\varphi(x))$

donc G décidable $\Rightarrow F$ décidable.

Le problème de l'arrêt (des machines de Turing) :

donnée : code $\langle M \rangle$ d'une machine de Turing,
et un mot booléen m

réponse : oui si le calcul de M sur m s'arrête
non sinon

Définition Le problème de l'arrêt est
$H = \{ \langle \langle M \rangle, m \rangle \mid$ le calcul $M(m)$ s'arrête $\}$

Théorème : H n'est pas décidable

Preuve : Soit la machine M qui sur l'entrée x fait :
- décide si la machine de code x s'arrête sur l'entrée x
- si la réponse est oui, alors M boucle
  sinon M s'arrête

Cette machine a un code $\langle M \rangle$.
Quel est le calcul de M sur $x = \langle M \rangle$?
si le calcul s'arrête alors cela veut dire que M sur $\langle M \rangle$ ne s'arrête pas
si le calcul ne s'arrête pas alors le calcul de M sur $\langle M \rangle$ s'arrête.

donc H n'est pas décidable $\square$

Variante $H' = \{ \langle M \rangle \mid$ le calcul de M sur l'entrée vide s'arrête $\}$

réduction $H' \to H$ $\qquad \langle M \rangle \xmapsto{\varphi} \langle \langle M \rangle, \varepsilon \rangle$

$\qquad\qquad\qquad\qquad x \qquad\qquad \langle x, \varepsilon \rangle$

réduction $H \to H'$
à une machine M et un mot m, on associe le code la machine M' :

M' : [sur une entrée x,
M' simule le calcul de M sur m

$M'(\varepsilon)$ s'arrête ssi $M(m)$ s'arrête.

$\varphi : \langle\langle M\rangle, m\rangle \mapsto \langle M'\rangle$ et calculable $\square$

<u>Problème ouvert</u> problème de Pisot :

$L = \{ \langle k, u_0 \dots u_{k-1}, a_0 \dots a_k \rangle \mid$

$k \in \mathbb{N}^*, u_0 \dots u_{k-1} \in \mathbb{Z}, a_0 \dots a_k \in \mathbb{Z}$

la suite définie par $a_0 u_n + a_1 u_{n+1} + \dots + a_k u_{n+k}$ a un zéro $\}$