# CH.8 Décidabilité

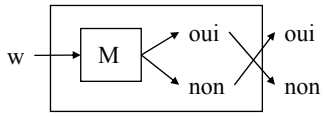
- 8.1 Les langages récursifs
- 8.2 La machine de Turing universelle
- 8.3 Des problèmes de langages indécidables
- 8.4 D'autres problèmes indécidables



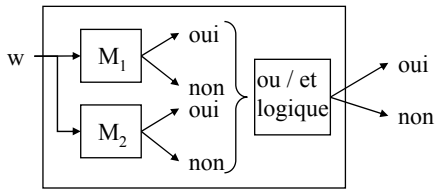
## 8.1 Les langages récursifs

Propriétés des langages récursifs :
Fermés par complémentation, union et intersection.

Complémentation :




Union / intersection

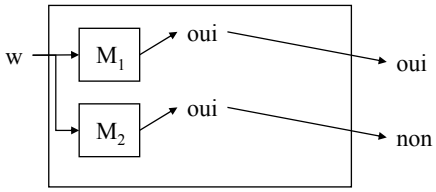



Propriétés des langages récursivement énumérables :
Fermés par union mais pas par intersection.

**Théorème :**
Le langage L est récursif si et seulement s'il est récursivement énumérable et son complémentaire aussi.

$M_1$ et $M_2$ reconnaissent L et son complémentaire.

La machine ci-contre s'arrête toujours.




Utile pour montrer qu'un langage n'est pas récursif. On montre que son complémentaire n'est pas récursivement énumérable.



## 8.2 La machine de Turing universelle

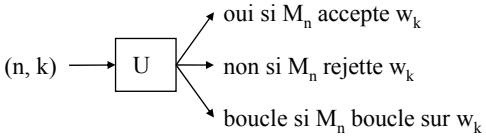
Numérotage des programmes permet de numéroter les machines de Turing (voir la machine comme un programme).
Pour tout entier n, on peut parler de la machine $M_n$. Cette machine peut être construite explicitement au moyen d'un programme (donc d'une machine de Turing.
On peut aussi numéroter les mots. Connaissant k, le mot $w_k$ peut aussi être construit explicitement par une machine de Turing.

**Exemple :** le langage diagonal $L_d = \{ w_i : w_i \notin L(M_i)\}$.
Ce langage n'est pas récursivement énumérable.
Supposons $L_d = L(M_k)$, reconnu par la machine de numéro k.
Soit $w_k$. On a alors :
$( w_k \in L_d ) \Leftrightarrow ( w_k \notin L(M_k) )$ par définition de $L_d$ ;
$\Leftrightarrow ( w_k \notin L_d )$ par définition de $L(M_k)$.
Contradiction dans tous les cas, donc $M_k$ n'existe pas.

Conséquence :
le problème "étant donnés n et k, la machine $M_n$ accepte-t-elle le mot $w_k$ ?" est indécidable.
Si on pouvait le décider, on pourrait aussi décider "étant donné k, la machine $M_k$ accepte-t-elle le mot $w_k$ ?". Ceci impliquerait la récursivité de $L_d$.

En fait le problème est équivalent au problème de l'arrêt.

En assemblant numérotage des machines et numérotage des mots, on a la **machine de Turing universelle** U :




Interpréteur = réalisation pratique de machine universelle.



## 8.3 Des problèmes de langages indécidables

Beaucoup d'arguements d'indécidabilité font appel au lemme suivant.

**Lemme :**
Soit une machine M prenant en entrée deux entiers n et k. On fixe n. On obtient, pour chaque n, une machine prenant en entrée un entier k.
Cette machine a comme numéro g(n). Alors la fonction qui à n associe g(n) est récursive totale.

Si on voit les machines de Turing comme des programmes, la démonstration est facile. Lorsque n est donné, on remplace dans le programme de M l'instruction de lecture de n par la valeur de n. On obtient ainsi un programme qui peut servir de numéro g(n).
Par contre, rien n'exige que la machine fasse quoi que ce soit d'utile...

Un problème sur les langages peut toujours se ramener un énoncé du type "le langage L a-t-il la propriété **S** ? On peut voir **S** comme un sous-ensemble de l'ensemble **L** de tous les langages récursivement énumérables (le sous-ensemble constitué des langages ayant cette propriété). De cette manière, une propriété et un sous-ensemble de **L** sont la même chose.

Les propriétés triviales sont **L**, satisfaite par tous les langages récursivement énumérables, et $\varnothing$, satisfaite par aucun langage.

**Théorème (Rice) :**

Si **S** est une propriété non-triviale, alors le problème "étant donné n, le langage $L(M_n)$ a-t-il la propriété **S** ?" est indécidable.

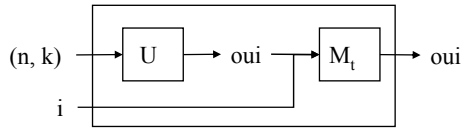
Démonstration :

Supposons qu'il existe une machine MS prenant en entrée n et décidant si $L(M_n) \in \mathbf{S}$. En intervertissant les sorties oui et non, on obtient une machine décidant $L(M_n) \notin \mathbf{S}$.



On suppose que le langage vide n'a pas la propriété **S**. Sinon, on s'intéresse à la propriété **L** – **S**. Puisque **S** n'est pas triviale, au moins un langage $L(M_t)$ possède la propriété **S**. On peut trouver t explicitement. Pour cela, on utilise MS en rentrant successivement tous les entiers 0, 1, ... La machine MS s'arrête toujours. Tant qu'elle s'arrête sur non, on continue. Puisque **S** est n'est pas triviale, elle finira par s'arrêter sur oui, pour un entier t.

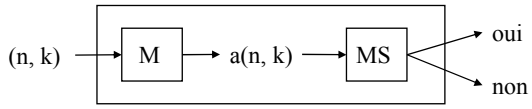
On construit la machine A suivante, avec 3 entrées, n, k et i :




Dans la machine A, la machine $M_t$ est lancée avec i en entrée seulement si la machine U s'est arrêtée sur oui.

En vertu du lemme, il existe une fonction a(n, k), récursive totale, calculée par la machine M telle que $A(n, k, i) = M_{a(n, k)}(i)$.

Supposons n et k donnés. Deux cas se présentent :
soit $w_k \in L(M_n)$, auquel cas $M_t$ démarre et répond oui si et seulement si $w_i \in L(M_t) = L$, c'est-à-dire que $L(M_{a(n, k)}) = L$ ;
soit $w_k \notin L(M_n)$, auquel cas $M_t$ ne démarre pas donc A n'accepte aucun i et $L(M_{a(n, k)}) = \varnothing$.
Soit maintenant la machine B suivante :




Avec les mêmes n et k donnés, deux cas se présentent de nouveau :
soit elle s'arrête sur oui, ce qui signifie $L(M_{a(n, k)}) \in \mathbf{S}$. On a vu plus haut que soit $L(M_{a(n, k)}) = L$, soit $L(M_{a(n, k)}) = \varnothing$ ; comme le langage vide n'est pas dans S, alors $L(M_{a(n, k)}) = L$, et donc $w_k \in L(M_n)$ ;
soit elle s'arrête sur non, ce qui signifie $L(M_{a(n, k)}) \notin \mathbf{S}$. La seule possibilité est maintenant $L(M_{a(n, k)}) = \varnothing$, et donc $w_k \notin L(M_n)$.

La machine B déciderait si $M_n$ accepte $w_k$, ce qui est indécidable...



Les conséquences du théorème de Rice sont innombrables ; sont indécidables :
"$L(M_n) = \varnothing$ ?", "$L(M_n)$ est fini ?", "$L(M_n)$ est récursif ?", "$L(M_n)$ est rationnel ?", "$\varepsilon \in L(M_n)$ ?", "$L(M_n) = L$ donné ?" ;
du dernier, on déduit "$L(M_n) = L(M_k)$ ?", et, de façon analogue, "$L(M_n) \cap L(M_k) = \varnothing$ ?", etc.

Ce résultat peut être raffiné en n'exigeant pas une machine qui décide (oui-non, récursif) mais une machine répondant oui si la propriété est vraie et boucle sinon (récursivement énumérable).
Par exemple, on peut trouver une telle machine pour le problème "$L(M_n) \neq \varnothing$ ?". Il suffit d'essayer tous les mots par taille croissante. Ce programme s'arrête lorsqu'il trouve un mot accepté. Ceci montre aussi qu'on ne peut pas faire de même pour "$L(M_n) = \varnothing$ ?", les deux problèmes étant complémentaires l'un de l'autre.

**8.4 D'autres problèmes indécidables**

Le problème de Post.

On donne deux suites finies A et B de mots sur un alphabet.

$A = w_1, w_2, \ldots, w_k$ et $B = x_1, x_2, \ldots, x_k$.

Probleme de Post :

Etant données les suites A et B, existe-t-il une suite $i_1, i_2, \ldots, i_m$ telle que $w_{i_1} w_{i_2} \ldots w_{i_m} = x_{i_1} x_{i_2} \ldots x_{i_m}$ ?"

Par exemple, sur l'alphabet {0, 1}, le problème avec

$A = \{1, 10111, 10\}$ et $B = \{111, 10, 0\}$ est vrai : $w_2 w_1 w_1 w_3 = x_2 x_1 x_1 x_3$.

Mais le problème avec

$A = \{10, 011, 101\}$ et $B = \{101, 11, 011\}$ n'a pas de solution.

On peut montrer que le problème de Post est indécidable, en montrant qu'il est équivalent à celui de l'appartenance d'un mot à un langage L(M).



On en déduit que le caractère intrinsèquement ambigu d'une grammaire algébrique est indécidable.

Le théorème du "point fixe"

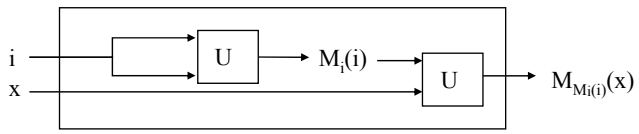
**Théorème :**

Soit σ une fonction récursive totale d'une variable entière. Alors il existe un entier $x_0$ et deux machines de Turing équivalentes de numéros $x_0$ et $\sigma(x_0)$.

En d'autres termes, $M_{x_0}(x) = M_{\sigma(x_0)}(x)$ quelle que soit l'entrée x.

Démonstration :

On construit la machine de Turing G suivante :




On considère la fonction g qui, à i fixé, donne le numéro de la machine ci-dessus, avec la seule entrée x (lemme vu plus haut) :
$M_{g(i)}(x) = G(i, x)$

La fonction g est récursive totale, donc $\sigma$ o g aussi, calculable par la machine de Turing $M_t$ : pour tout x, on a $M_t(x) = \sigma(g(x))$.
Soit $x_0 = g(t)$, qui est bien défini. On vérifie qu'il convient.
En effet,
$M_{x_0}(x) = M_{g(t)}(x) = G(t, x) = M_{M_t(t)}(x) = M_{\sigma(g(t))}(x) = M_{\sigma(x_0)}(x)$.

Le théorème est donc démontré.



On a calculé un tel $x_0$. Mais la machine correspondante n'a pas d'autre propriété. peut-être qu'elle ne produit jamais aucun résultat...

On peut maintenant évoquer un théorème "à la Gödel".
On a un ensemble fini d'axiomes F et on dispose du calcul logique des prédicats pour fabriquer des démonstrations de propositions. Si une proposition dérive logiquement des axiomes, on dit qu'elle est démontrable dans F. On suppose aussi que F est non-contradictoire, ou encore que toutes les propositions qu'on peut démontrer à partir de F sont vraies.

**Théorème :**
Il existe une machine de Turing M qui ne s'arrête sur aucune entrée et pour laquelle, quelle que soit l'entrée, il n'existe pas de démonstration dans F qu'elle ne s'arrête pas.

Démonstration :

On peut numéroter toutes les démonstrations (suites finies de symboles).

Etant donnée une démonstration $D_i$ et une proposition, on peut, à l'aide d'un algorithme, vérifier si la proposition est démontrée par cette démonstration.

Soit donc d(i, j) =1 s'il existe une démonstration que la machine $M_i$ ne s'arrête pas sur le mot $w_j$ ; d(i, j) est indéfini sinon.

On peut calculer d : la proposition est que $M_i(j)$ s'arrête. On énumère toutes les démonstrations et on vérifie si chacune démontre cette proposition. S'il existe une démonstration, l'algorithme s'arrête et on renvoie la valeur 1. Sinon, on bouclera indéfiniment. Donc d est récursive partielle. En vertu du lemme du paragraphe 8.3, il existe une fonction récursive totale g telle que, pour tout j, $M_{g(i)}(j) = d(i, j)$.



D'après le théorème du point fixe, on construit un entier $i_0$ tel que, pour tout j, on a $M_{i_0}(j) = M_{g(i_0)}(j) = d(i_0, j)$.

Supposons que $M_{i_0}(j)$ s'arrête. Alors $d(i_0, j)$ est défini, donc vaut 1 et donc il existe une démonstration que $M_{i_0}(j)$ ne s'arrête pas. Puisque à partir de F on ne peut rien démontrer de faux, on arrive à une contradiction.

Donc, $M_{i_0}(j)$ ne s'arrête pas. Par conséquent $d(i_0, j)$ n'est pas défini. Il n'existe donc pas de démonstration que $M_{i_0}(j)$ ne s'arrête pas.

On a trouvé la machine M cherchée.

Le théorème d'incomplétude de Gödel précise les axiomes (arithmétique de Peano), la numérotation des démonstrations et peut du coup se passer des machines de Turing en ramenant tout à des propriétés arithmétiques des entiers.