Automates et Calculabilité

– Universalité et diagonalisation –

Leçon 11 : 12 jan 2004

Uwe Nestmann

EPFL-I&C

Le problème de l'arrêt (I)

$$\text{HP} \stackrel{\text{def}}{=} \{\ (M,x) \mid \begin{array}{ll} & M \text{ est une MdT} \\ \wedge & x \text{ est un mot d'entrée pour } M \\ \wedge & \boxed{M \text{ s'arrête pour } x} \end{array} \}$$

$$\text{MP} \stackrel{\text{def}}{=} \{\ (M,x) \mid \begin{array}{ll} & M \text{ est une MdT} \\ \wedge & x \text{ est un mot d'entrée pour } M \\ \wedge & \boxed{x \in L(M)} \end{array} \}$$

L'ensemble HP, est-il récursif ? (Et MP ?)

En d'autres mots : existe-t-il une MdT totale qui accepte HP ?

En d'autres mots : l'arrêt d'une machine M quelconque pour n'importe quel mot d'entrée x valable est-il décidable ?

Universalité

L'idée est déjà vieille : si un langage de programmation est assez puissant, il est possible d'écrire un compilateur pour ce langage dans ce langage même.
On parle aussi de « *bootstrapping* ».
Exemple répandu : écrire un interprétateur pour LISP en LISP.
→ Programmation IV

Normalement, un programme traite des données.
Par exemple, une MdT M exécute son algorithme qui est déterminé par sa fonction de transition (noter qu'il s'agit toujours du même algorithme) sur un ensemble infini de données.

L'idée de l'***universalité*** est de construire une machine U qui est capable de simuler toute autre machine M ;
pour ce faire, il faut ***encoder ces autres machines comme données*** d'entrées.

Une MdT Universelle (I)

Une machine universelle U prend deux types de données en entrée:

- ☐ une représentation d'une MdT M ;
- ☐ un mot x à accepter par la machine M.

Pour pouvoir simuler l'exécution de M sur x par U, il faut

1. trouver un codage qui transforme toute MdT M en mot d'entrée $[M]$ pour la machine universelle U ;
2. coder les entrées x de M en entrée $[x]$ acceptable pour U.

On donne le mot $[M]\#[x]$ en entrée pour U . Le symbole $\#$ dans l'alphabet d'entrée de U ne nous sert que comme séparateur entre $[M]$ et $[x]$. Le but :

$$\begin{array}{lcl} U([M]\#[x]) \text{ s'arrête} & \text{ssi} & M \text{ s'arrête pour } x. \\ [M]\#[x] \in L(U) & \text{ssi} & x \in L(M). \end{array}$$

Une MdT Universelle (II)

Soit $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$ une MdT.

En utilisant l'alphabet d'entrée $\{0, 1\}$ pour la machine U, on peut coder une machine par le mot :

$$0^n\ 1\ 0^m\ 1\ 0^k\ 1\ 0^s\ 1\ 0^t\ 1\ 0^r\ 1\ 0^u\ 1\ 0^v\ 1\ \ldots$$

où $n = \|Q\|$, $m = \|\Gamma\|$, $k = \|\Sigma\|$,
et où $\vdash$ et $\sqcup$ sont les $u^{\text{ème}}$ et $v^{\text{ème}}$ symboles,
suivi par une série de mots de la forme :

$$\ldots\ 0^p\ 1\ 0^a\ 1\ 0^q\ 1\ 0^b\ 1\ 0\ 1\ \ldots$$

pour représenter les paires $((p, a), (q, b, L)$ de la fonction δ
où le dernier bit indique le sens de déplacement L ou R.

Comment représenter des mots d'entrée x de la MdT M ?

Une MdT Universelle (III)

Fonctionnement de U :

- ☐ U peut, par exemple, travailler sur 3 rubans :
  - un ruban pour contenir le codage de M comme entrée,
  - un ruban pour contenir le codage du ruban de M,
  - un ruban pour contenir l'état de M et la position de la tête de M.
- ☐ U vérifie d'abord si l'entrée $[M]\#[x]$ est valable.
- ☐ U simule une transition de M par plusieurs transitions :
  - lire l'état de M, la position de sa tête, et le symbole,
  - chercher la valeur de δ pour cette configuration,
  - établir la nouvelle configuration de M ...
- ☐ Par définition, U accepte/rejette ssi M accepte/rejette.

Le problème de l'arrêt (II)

$$\text{HP} \stackrel{\text{def}}{=} \{\ [M]\#[x] \mid \begin{array}{ll} & M \text{ est une MdT} \\ \wedge & x \text{ est un mot d'entrée pour } M \\ \wedge & \boxed{M \text{ s'arrête pour } x}\ \} \end{array}$$

L'ensemble HP, est-il récursif ?

Observer que U n'est pas totale : si M ne s'arrête pas pour x,
alors U ne s'arrête pas pour $[M]\#[x]$.

Peut-on construire une MdT U' améliorée telle que :

- ☐ U' accepte $[M]\#[x]$ si M accepte x ;
- ☐ U' rejette $[M]\#[x]$ si M rejette x ;
- ☐ U' ***rejette*** $[M]\#[x]$ si M ***boucle*** pour x ;

Diagonalisation (Cantor, $19^{\text{ème}}$ siècle)

Une technique de preuve, inventée pour démontrer qu'il n'existe pas de surjection $f : \mathbb{N} \to 2^{\mathbb{N}}$. ($2^{\mathbb{N}} = \{C \mid C \subseteq \mathbb{N}\}$.)
L'argument s'applique à tous les ensembles A à la place de $\mathbb{N}$.

Par contradiction :

- ☐ Supposons qu'il existe une surjection $f : A \to 2^A$.
- ☐ Définissons $B \stackrel{\text{def}}{=} \{x \in A \mid x \notin f(x)\}$.
- ☐ Comme $B \subseteq A$, nous avons aussi $B \in 2^A$.
- ☐ Comme f est surjective, il existe $y \in A$ tel que $f(y) = B$.
- ☐ La contradiction apparaît par :

$$\begin{array}{lcll} y \in f(y) & \Longleftrightarrow & y \in B & \text{par } f(y) = B \\ & \Longleftrightarrow & y \notin f(y) & \text{par définition de } B \end{array}$$

Mais, où est la « diagonalisation » ?

Diagonalisation pour $\mathbb{N}$!

Soit $f : \mathbb{N} \to 2^{\mathbb{N}}$ une surjection.
Notons tous les éléments de $2^{\mathbb{N}}$ dans un tableau de $\mathbb{N} \times \mathbb{N}$

$$\begin{array}{cc|cccccc} & T & 0 & 1 & 2 & \ldots & i & \rightarrow \\ \hline f(0) & 0 & & & & & & \\ f(1) & 1 & & & & & & \\ \vdots & \vdots & & & & & & \\ f(j) & j & & & & & & \\ \vdots & \downarrow & & & & & & \end{array} \qquad T(i,j) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{si } i \in f(j) \\ 0 & \text{si } i \notin f(j) \end{cases}$$

Ceci représente tous les éléments de $2^{\mathbb{N}}$, donc ***tous les sous-ensembles*** $f(j)$ ***de*** $\mathbb{N}$, par une ligne j du tableau T.

Contradiction : construire, par « inversion de la diagonale », un sous-ensemble de $\mathbb{N}$ pour lequel il n'existe pas de ligne dans le tableau. Comparer avec l'ensemble $B \stackrel{\text{def}}{=} \{ x \in A \mid x \notin f(A) \}$.

Diagonalisation, problème de l'arrêt (I)

Soit $[\cdot]$ un codage *injectif* qui transforme des MdT avec alphabet d'entrée $\{0,1\}^*$ en mots du même alphabet, donc $[M] \in \{0,1\}^*$. Notons qu'il est possible que ce codage ne soit *pas surjectif.*

Soit $x \in \{0,1\}^*$. M_x défini le « décodage » de x, alors la MdT satisfaisant $[M_x] = x$ si elle existe. S'il n'existe pas de M telle que $[M] = x$, on associe avec x une machine triviale M^{triv}.

Cela détermine une énumération de toutes les MdT sur $\{0,1\}^*$:

$$M_\epsilon, M_0, M_1, M_{00}, M_{01}, M_{10}, M_{11}, M_{000}, M_{001}, \ldots$$

Établir un tableau T sur $\{0,1\}^* \times \{0,1\}^*$ tel que

$$T(x,y) \stackrel{\text{def}}{=} \begin{cases} H & \text{si } M_y \text{ s'arrête pour } x \\ L & \text{si } M_y \text{ boucle pour } x \end{cases}$$

Diagonalisation, problème de l'arrêt (II)

Par contradiction :

- ☐ Supposons qu'il existe une MdT ***totale*** K avec $L(K) =$ HP.
- ☐ Pour tout $x, y \in \{0,1\}^*$,
  - K s'arrête et accepte $[M_y]\#[x]$ si M_y s'arrête pour x ;
  - K s'arrête et rejette $[M_y]\#[x]$ si M_y boucle pour x.

  Comme K est totale, elle calcule $T(x,y)$ en temps fini !
- ☐ Définissons ***par diagonalisation*** la MdT D telle que :
  - D accepte x si K rejette $[M_x]\#[x]$ (M_x boucle pour x);
  - D boucle pour x si K accepte $[M_x]\#[x]$ (M_x s'arr. pour x);
- ☐ Il existe y tel que $[D] = y$, alors $D = M_y$.
- ☐ M_y s'arr. pour $y \iff K$ rej. $[M_y]\#[y]$ par déf. de D
  $\iff M_y$ boucle pour y par déf. de K

MP ?

On démontre que MP n'est pas récursif
par réduction au problème d'arrêt,
et par le fait que l'on sait déjà que HP n'est pas récursif.

Par contradiction :

- ☐ Supposons qu'il existe une MdT ***totale*** M^{MP} telle que $L(M^{\text{MP}}) =$ MP.
- ☐ Supposons, nous cherchons si une MdT M s'arrête pour x.
- ☐ Construire de manière uniforme la machine $\widehat{M}$ qui accepte x dans les cas ou M ***accepte ou rejette*** x.
- ☐ Alors, $\widehat{M}$ accepte x ssi M s'arrête pour x.
- ☐ Si MP était récursif, on pourrait décider si M s'arrête pour x en construisant $\widehat{M}$ et en utilisant la machine M^{MP} pour décider si $\widehat{M}$ accepte x.