

# Calculabilité et Complexité - Nicolas Ollinger

Alexandre Masson

15 Janvier 2013

## Table des matières

<b>1</b>	<b>Fondements de l'informatique</b>	<b>3</b>
<b>2</b>	<b>Calculabilité</b>	<b>4</b>
<b>3</b>	<b>Motivation informelle(approche naïve)</b>	<b>4</b>
<b>4</b>	<b>Machines de Turing</b>	<b>7</b>
4.1	Définitions . . . . .	7
<b>5</b>	<b>Thèse de Chruch-Turing-Kleene</b>	<b>9</b>
<b>6</b>	<b>la machine universelle</b>	<b>9</b>
<b>7</b>	<b>Indécidabilité</b>	<b>9</b>
<b>8</b>	<b>Interlude</b>	<b>11</b>
<b>9</b>	<b>Evaluation</b>	<b>12</b>
<b>10</b>	<b>Corollaire</b>	<b>14</b>
<b>11</b>	<b>Points Fixes et récursion</b>	<b>16</b>
11.1	Auto-référence . . . . .	16
11.2	Théorème de récursion : . . . . .	18
<b>12</b>	<b>Notions</b>	<b>19</b>

# 1 Fondements de l'informatique

le But du cours est de nous donner quelques éléments culturels pour comprendre comment certaines choses seront toujours valable dans 50 ans et pourquoi c'est de la science.

Computability en Anglais , on s'intéresse à la propriété d'être calculable.

Qu'est ce qui est calculable ?

Complexité s'intéresse à la meilleur complexité que l'on peut obtenir avec un algorithme. Existe il des problèmes intrinsèquement plus complexes que d'autres (ex : tri , pas possible de faire mieux que  $O(n \log(n))$ ).

Il est possible ici de faire un parallèle entre Programmation et Algorithmique. Ces deux choses travaillent sur : comment trouver de programme (ou comment construire un programme qui résous un problème) et calculabilité et complexité détermine l'existence d'un programme pour résoudre un problème.

thèse de Church-Turing(rien d'intuitif, quelque-soit le langage de prog, on ne pourra pas calculer plus de choses).

Les programmes qui calculent des fonctions partielles , peuvent ne pas répondre à des problèmes.

Un interprète, prend un programme et une donnée et renvoi le résultat du programme sur la donnée. Une fonction universelle existe dans tout modèle de calcul(Travaux de Alan Turing).

Un programme qui affiche son propre code dans son exécution, existe dans tous les langages de programmation. Théorème de récurrence, pour toute fonction , il existe une fonction auto-référente( qui se prend elle même en paramètre) qui prend e code en paramètre et affiche son code.

il n'existe pas de programme qui prend un autre programme en paramètre et dit si celui-ci s'arrête ou pas.

## 2 Calculabilité

**Définition** Études des modèles de calcul et des propriétés qui sont indépendantes du modèle.

On s'intéresse :

- aux programmes vus à la fois comme :
  - chaîne de caractères (code source) : un mot sur un alphabet fini
  - un objet qui code une fonction (soit des chaînes  $\rightarrow$  les chaînes, soit des Entiers  $\rightarrow$  les Entiers).

**Remarque** on confondra allègrement et sans scrupules les ensemble  $E^*$  (ensemble de tous les mots d'un alphabet),  $N^*$  (ensemble des couples d'entiers de taille\*) ,  $N$  ,... qui sont codables bijectivement et récursivement (codage et décodage calculables par un programme) les uns en les autres.

- problèmes de décision : problème auquel on répond par oui ou non, c'est un langage (donc un sous ensemble de  $E^*$ ).

**Intuitivement** Une fonction  $f$  est :  $E^* \rightarrow E^*$  partielle est calculable si il existe un programme tel que :  $\forall x \in \sigma^*$  .

Un problème de décision  $P \subset \sigma^*$  est décidable si  $f_P : x \rightarrow$  oui si  $p \in P$  est calculable , non sinon.

## 3 Motivation informelle(approche naïve)

Fixons un langage de programmation impératif IMP idéalisé. Code Source, entrée, sortie  $\in \text{UTF8}^*$ . Prenons  $\text{eval}(p, x) = y$  si le programme sur l'entrée  $x$  s'arrête en renvoyant  $y$ ,  $\perp$  sinon.

$$\begin{aligned}\phi p : \sigma^* &\rightarrow \sigma^* \\ \phi(x) &= \text{eval}(p, x).\end{aligned}$$

Question : Est ce que toute fonction est IMP-calculable? NON, il suffit de compter!

**Prop** : Toutes les fonctions ne sont pas IMP-calculable.

**Démonstration** : Comparons le nombre des fonctions et le nombre des programmes,  $\sigma^*$  est un ensemble infini dénombrable.

$\sigma^* \rightarrow \sigma^*$  n'est pas dénombrable.

donc il existe des fonctions non IMP-calculable.  $\sigma^*$  est dénombrable : il suffit d'énumérer par taille croissante et ordre lexicographique à taille fixée.

$\sigma^* \rightarrow \sigma^*$  n'est PAS dénombrable.

Procédé de Cantor :

par l'absurde, on suppose que  $\sigma^* \rightarrow \sigma^*$  est dénombrable : énumération( $\phi$  i)  
tel que  $\phi i = \sigma^* \sigma^*$

**Remarque :** l'entrée est  $\langle p, x \rangle$  codage dans la paire(p,x).

**Proposition :** Il existe un programme universel U tel que  $\forall p \forall x U(\langle p, x \rangle)$   
 $= \text{eval}(p, x)$ , il suffit de prendre pour U un interprète IMP codé en IMP.

**Proposition :** IMP-arrêt est IMP-indécidable.

**Démonstration :** Par l'absurde. Supposons que IMP-arrêt soit IMP-décidable, idem qu'il existe un programme debug tel que  $\phi \text{ debug } (\langle p, x \rangle) = \text{oui}$  si  $\phi p(x) \neq \perp$ , NON sinon.

On construit un programme malicieux comme suit : on lit X, on execute debug, si debug dis oui boucle infinie while 1, sinon return, masi alors,  $\phi$  malicieux(malicieux) répond oui si imp-arrêt dis non et non si imp arrête dis oui, donc ce programme ontredit IMP-arrêt CONTRADICTOIRE!!! donc IMP-arrêt n'est pas IMP-calculable.

Comment utiliser ce résultats pour démontrer que d'autres problèmes sont indécidables

**imp-arrêt** entrée : code source P.

question : Est ce que  $\phi(\epsilon) \neq \perp$ .

On aimerait montrer que IMP-arrêt0 est lui aussi IMP-indécidable en montrant que IMP-arrêt est plus simple que IMP-arrêt0. On va construire un programme qui utilise IMP-arrêt0 pour résoudre imp-arrêt.

Est ce que  $\phi_p(x) \neq \perp$

On a montré que IMP-arrêt est indécidable on va dire que  $\text{IMP-arrêt} \leq \text{imp IMP-arrêt0}$ .

On va construire un programme qui utilise un programme qui résout IMP-arrêt0 pour résoudre IMP-arrêt. Dans IMP-arrêt, on a en entrée P et x, on doit manipuler P et x, pour contruire un programme P' qui va nourrir imp-arrêt0.

Démonstration : Par l'absurde, on suppose qu'il existe un programme qui décide imp-arrêt0.

On construit alors le programme suivant :

```

Read Z ;
P :=  $\pi_1(Z)$ 
X :=  $\pi_2(Z)$ 
Q := le programme P où la première ligne [read Y] est remplacée par [read Y ;
Y := X] ;
R := IMP-arrêt0(Q) ;
write(R) ;

```

Ce programme résout imp-arrêt, qui est imp-indécidable : CONTRADICTION.  
Donc IMP-arrêt0 est indécidable.

Cette preuve utilise une technique d'évaluation partielle.

**Proposition :** Il existe une fonction IMP-calculable  $s$  qui à  $\langle P, x \rangle$  associe le code source d'un programme qui ignore son entrée et simule le programme  $P$  sur l'entrée  $x$ , c'est à dire :  $\forall y, \phi_s(\langle P, x \rangle)(Y) = \phi_P(x)$ .

**Proposition :** Il existe une fonction IMP-calculable  $s$  qui à  $\langle P, x \rangle$  associe le code source d'un programme qui sur une entrée  $Y$ , simule le programme  $P$  sur l'entrée  $\langle X, Y \rangle$  :  $\phi_s(\langle P, x \rangle)(Y) = \phi_P(\langle X, Y \rangle)$ .

**Démonstration :** Idem, mais remplacer [read Y] par [read Y, Y :=  $\langle X, Y \rangle$ ].

Prop : il existe un programme IMP qui retourne son propre programme code source.

**Démonstration :**

Soit  $R$  le programme suivante :

```

Read(Z) ;
X :=  $\pi_1(Z)$ 
Y :=  $s'(\langle X, X \rangle)$ 
write Y

```

donc le programme  $s'(\langle R, R \rangle)$  écrit son propre code source.

**Bilan :** Ce qui précède ne dépend pas vraiment de IMP.

Quelques principes sous-jacents :

- dénombre de manière calculable les programmes ;
- possibilités d'enchaîner, concaténer, composer des programmes.
- existence d'un crochet  $\langle \cdot, \cdot \rangle$  et de projection  $\pi_1, \pi_2$  calculables.
- existence d'un programme universel ;
- existence de fonctions d'évaluation partielle :  $s, s', \dots$

La calculabilité s'intéresse aux propriétés des fonctions calculables qui sont indépendantes du modèles de calcul considéré

## 4 Machines de Turing

Les MT sont le modèle de calcul de référence du cours. Il s'agit d'un modèle de machine mécanique abstrait. C'est le modèle de référence en complexité. A.M. Turing : On computable numbers with an application to the Entscheidungsproblem. 1936

### 4.1 Définitions

Le ruban est coupé en case, quantité finie d'information, on suppose qu'on a un alphabet fini qui nous permet d'écrire dans les cases du ruban, et le contenu d'un case c'est une lettre sur un alphabet fini.

L'opérateur est une flèche qui pointe sur une case, il a le droit de mémoriser une quantité finie d'informations et est dans un certain état, cet état est lui aussi dans un alphabet fini, l'ensemble  $Q$  des états, il a différentes opérations possibles : lire et écrire dans la case pointée, il peut aussi changer d'état, il peut déplacer le pointeur vers la gauche ou la droite. Le programme c'est : quoi faire quand on commence, que faire quand on a un couple état/lettre dans la case.

**Définition :** Une MT est un octuplet (ensemble d'états( $Q$ ),  $\Sigma$ (alphabet d'entrée/sortie),  $\Gamma$ , lettre blanche ( $B$ ), état initial( $Q_0$ ), un état fini succès( $Q_{oui}$ ), un ensemble d'état finaux échec( $Q_{non}$ ) et  $\delta$  : la fonction de transition).

**Définition :** Une configuration d'une MT  $M$  est un triplet  $(q, c, p) \in Q * \gamma \exp Z \times Z$ .

**Définition :** Il existe une transition de la configuration  $(q, v, p)$  à la configuration  $(q', c', p')$  de  $M$ , notée  $(q, c, p \vdash (q', c', p'))$  si  $\delta(q, c(p)) = (q', a, d)$  où  $\forall k \in Z$   $c'(k) = a$  si  $k = p$ ,  $c(k)$  sinon, et  $p' = p-1$  si  $d = <$ ,  $p$  si  $d =$  bas, et  $p+1$  si  $d = >$

**Remarque :** Dans une utilisation normale d'une MT le ruban est  $B$  presque partout. On préférera noter uqv une configuration  $(q, c, p)$  telle que :  $u$  : ce qu'il y a de pas blanc avant la case,  $q$  l'état de la MT, et  $V$  ce qu'il y a après sur la bande.

**Définition :** La configuration initiale de  $M$  sur l'entrée  $u \in \Sigma^*$  est la configuration  $q_0u$ .

**Définition** une configuration acceptante de  $M$  est une configuration  $Uq_{oui}V$ .

**Définition :** une configuration de refus de  $M$  est une configuration  $Uq_{non}V$ .

La machine s'arrête lorsqu'elle atteint un état d'arrêt, c'est à dire l'état acceptant  $q_0$  ou l'état de refus  $q_{non}$ .

**Remarque :** On notera  $| \cdot |^n$  n transitions successive,  $| \cdot |^* = \text{union des transitions}$  ( $| \cdot |^0$  l'identité).

**Définition :** Le langage  $L(M)$  d'une MT  $M$  est l'ensemble des mots acceptés :  $L(M) = \{ u \in \Sigma^* \mid q_0 \xrightarrow{*} u \text{ } q_{oui} \}$ .

**Définition :** Un langage est récursivement énumérable s'il est le langage d'une MT  $M$ , ie  $L = L(M)$ .

**Définition :** Un langage  $L$  est récursif s'il est le langage d'un MT qui s'arrête sur toute entrée.

**Exemple :** Langage des palindrome :  $L = \{ u \in a,b^* \mid u \text{ est un palindrome} \}$ .

**Définition** un problème est décidable si son langage est récursif.

exemple : le problème de l'arrêt :

l'entrée : une MT  $M$  et  $u$  un mot  $u \in \Sigma^*$ . Question Est que  $\Phi_M(u) = \perp$  ?

Langage associé :  $K = \{ \langle M, u \rangle \mid \Phi_M(u) = \perp \}$

**Définition** Une fonction partielle  $f : \Sigma^* \rightarrow \Sigma^*$  est calculable si il existe une MT  $M$  qui calcule  $F$ , c'est à dire que  $\forall u \in \Sigma^*$

- $q_0 \xrightarrow{*} v_{oui} f(u) \text{ si } f(u) \neq \perp$
- $\neg q_0 \xrightarrow{*} v_{oui} f(u) \text{ si } f(u) = \perp$

**Proposition** Un langage  $L \subseteq \Sigma^*$  est récursif si et seulement si  $L$  est récursivement énumérable et  $L$  est complémentirement récursivement énumérable

**Remarque :**  $L$  est co-r.e signifie  $\Sigma^* \setminus L$  est r.e.

**Démonstration dans le sens  $\Rightarrow$**   $L$  est récursif donc il existe une MT  $M$  totale telle que  $L(M) = L$ .

$L$  est r.e car  $M$  reconnaît  $L$ .

$L$  est Co-r.e car  $M'$ , copie de  $M$  où  $q_{oui}$  et  $q_{non}$  sont échangés, reconnaît  $\Sigma^* \setminus L$

**Dans le sens  $\Leftarrow$**  Soient  $M$  une MT qui reconnaît  $L$  et  $N$  une MT qui reconnaît  $\Sigma^* \setminus L$ , on construit une MT  $Z$  qui, sur une entrée  $u$ , simule en parallèle  $M$  et  $N$  sur l'entrée  $u$ . Si  $M$  s'arrête alors  $Z$  répond oui. Si  $N$  s'arrête alors  $Z$  répond non.  $Z$  est totale et reconnaît  $L$ .

**Exercice :** Montrons que  $L$  est récursif si et seulement si  $\chi_L$  est calculable, où  $\forall u \in \Sigma^* \chi_L(u) = \text{OUI}$  si  $u \in L$  et NON sinon



## 5 Thèse de Church-Turing-Kleene

- 1936 A.Turing définit le calcul/ par les MT. <- complètement mécanique
- 1936 A.Church définit le calcul par la  $\lambda$ -*definissibilité*, i.e le  $\lambda$ -calcul
- <- modèle fonctionnel.

**Thèse** La notion intuitive de ce qui est calculable correspond exactement à la notion de fonction calculable par MT.

**Idée :** essayer de construire d'autre modèle de calcul "raisonnables" : la notion de calculables coïncide toujours (en tout cas de 1936 à 2013).

## 6 la machine universelle

**Proposition** Il existe une machine de Turing U telle que :  
 $\forall M \forall u \Phi_U(\langle M, u \rangle) = \Phi_M(u)$

**Proposition** La fonction STEP suivant est calculable.

$\forall M \forall u \in \Sigma^* \forall k \in \mathbb{N} :$

$\text{STEP}(\langle M, u, k \rangle) =$

- TERMINÉ(q,v) si M sur l'entrée u arrête avant k étapes avec sortie v dans l'état q
- ENCOURS sinon.

**Principe de fonctionnement de U** Sur l'entrée  $\langle M, u \rangle$  U doit simuler le calcul de M sur l'entrée u. Simuler M c'est lire le ruban , écrire, déplacer la tête , modifier l'état.

U organise sa mémoire, son ruban , en zones, pour coder

- l'état de M
- le ruban de M
- la table de transitions de M

U se contente de chercher l'état et la lettre correcte dans la table des transitions, de les remplacer et de déplacer le marqueur. Ajouter des détails techniques pour initialiser et arrêter le calcul.

STEP fonctionne pareil en limitant le nombre de pas de simulation à k.

## 7 Indécidabilité

**Proposition** Il existe des langages non rékursifs. Preuve par dénombrement : Il y a  $\omega$  MT et  $2^\omega$  fonctions partielles de  $\Sigma^* \rightarrow \Sigma^*$ .

$K = \{ \langle M, u \rangle \mid \Phi_M(u) \neq \perp \}$

**Remarque :** K est r.e il suffit d'utiliser U pour tester  $\Phi_M(u)$ .

**Théorème de l'arrêt** K n'est pas récursif, ie le pb de l'arrêt n'est pas décidable.

**Démonstration** Supposons qu'il existe une MT H telle que

$\forall M \forall u \Phi_H(<M, U>) =$

- OUI si  $\Phi_M(u) \neq \perp$
- NON si  $\Phi_M(u) = \perp$

On diagonalise H en construisant une MT qui s'appelle D (comme diagonale) qui sur l'entrée M (où M est le code d'une machine) :

- exécuter H sur l'entrée  $<M, M>$
- si H a répondu OUI, on boucle
- si H a répondu NON, on s'arrête.

$\forall M, \Phi_D(M) = \perp$  si  $\Phi_M(M) \neq \perp$ , arrêt si  $\Phi_M(M) = \perp$

Donc, si  $L = D : \Phi_D(D) = \perp$  si  $\Phi_D(D) \neq \perp$ , arrêt si  $\Phi_D(D) = \perp$  : CONTRADICTION!!!

Donc H n'existe pas, K n'est pas récursif, K n'est pas co-r.e.

**Théorème de Rice** Soit P une propriété non triviale des fonctions partielles  $f : \Sigma^* \rightarrow \Sigma^*$ .

L'ensemble des machines de Turing qui calculent une fonction qui satisfait P n'est pas récursif. P sélectionne parmi les fonctions partielles un sous-ensemble, non-triviale, indique qu'il existe deux fonction f et g différentes telles que P(f) est vrai et P(g) est faux.  $M \mid P(\Phi_M)$  n'est pas récursif.

exemple de propriété :

- l'ensemble des fonctions totales
- l'ensemble des fonctions définies nulle part
- etc...

5 Février 2013

Pour toute propriété non triviale sur les fonctions partielles savoir si une MT calcule une fonction qui satisfait est indécidable : cette fonction est triviale, si il existe une fonction f qui la satisfait, et une fonction g :  $f \neq g$  qui ne la satisfait pas. Exemple :  $\{ M \mid P(\Phi_M) \}$  n'est pas récursif.  $\{ M \mid \phi_M \neq \perp \}$

$$K = \{ \langle M, x \rangle \mid \Phi_M(x) \neq \perp \}$$

**Démonstration** Par l'absurde, supposons qu'il existe une propriété non-triviale  $P$  calculable par MT  $T_P$ . soit  $\perp$  une fonction définie nulle part. Supposons sans perte de généralité, que  $\perp \notin P$  et soit  $S$  une MT telle que  $\Phi_S \in P$ .

On construit une machine de Turing  $F$  comme suit :

- Sur l'entrée  $\langle M, x \rangle$
- Soit  $T$  le codage de la MT suivante :
  - Sur l'entrée  $y$  :
    - simuler  $M$  sur l'entrée  $x$ .
    - si  $M$  s'arrête , on simule  $S$  sur l'entrée  $y$
    - si  $S$  s'arrête, on s'arrête.
- Simule  $T_P$  sur l'entrée  $T$ 
  - Si  $T_P$  répond OUI, accepter  $\langle M, x \rangle$
  - Si  $T_P$  répond NON, rejeter  $\langle M, x \rangle$

$F$  est totale.

$\Phi_F(\langle M, x \rangle) = \{ \text{OUI si } T_P(T) = \text{OUI, ie si } \Phi_M(x) \neq \perp$   
 et NON si  $T_P(T)$ , ie si  $\Phi_M(x) = \perp \}$

$F$  résout le problème de l'arrêt ! CONTRADICTION

Les problèmes indécidables sont légion :

- le typage au second ordre du  $\lambda$ -calcul ;
- mortalité des produits de matrices carrées
- problème du mot pour les groupes finiment engendrés.
- exemple : existence d'une solution entière à une équation polynomiale à coefficient entiers (10<sup>ème</sup> problème de Hilbert).
- Pavabilité du plan par un jeu de tuiles de Wang (tuiles divisées en quater avec des couleurs au coins, pas le droit de les tourner , et on les collent avec deux couleurs identiques aux arrêtes adjacentes) ;
  - entrée : famille de tuiles de Wang  $T \subseteq \Sigma^4$
  - Question :  $T$  pave-t-il le plan , ie,  $\exists c : \mathbb{Z}^2 \rightarrow T$  telle que les contraintes soient satisfaites.

#### Problème de correspondance de Post

- entrée :  $u_1, u_2, \dots, u_n, v_1, v_2, \dots, v_n \in \Sigma^*$
- question :  $\exists (ik) u_1, u_2, \dots, u_n = v_1, v_2, \dots, v_n$  ?

## 8 Interlude

**Définition** Une machine linéairement bornée (MTLB) est une MT qui s'interdit d'écrire sur  $B$ .  $\forall q, q', a, d \sigma(q, B) = (q', a, d) \rightarrow a = B$  car on ne peut pas écrire sur blanc.

**Proposition** L'arrêt des MTLB est décidable.

**Démonstration** On construit le graphe fini des configuration accessibles à partir de l'entrée  $x$  pour la machine  $M$ . Il y a environ  $|Q| * (|x| + 2) * |\Gamma|^{|x|}$

$\exists_{MTLB}$

- entrée :  $M$  une MTLB
- question : Existe-t-il une entrée  $x$  acceptée par  $M$ ?

**Proposition**  $\exists_{MTLB}$  est indécidable

**Démonstration** Par l'absurde, on suppose que  $\exists_{MTLB}$  est décidable, et on construit un programme qui résoud l'arrêt des MT.

IL s'agit étant donnés une MT  $M$  et une donnée  $x$ , de calculer une MTLB  $f(\langle M, x \rangle)$  qui accepte une entrée si et seulement si le MT  $M$  s'arrête sur l'entrée  $x$ . Notre MTLB est un vérifieur/parseur/accepteur de trace de MT : il considère son entrée comme une suite de configuration valide de  $M$  qui commence pas  $q_0^x$  et termine par une configuration d'arrêt :

$BB\#q_0x1x2...xn\# \dots \# \dots\#BB$

$$\begin{aligned}\Sigma' &= Q_M \cup \Gamma_M \cup \{\#\} \\ \Gamma' &= \Sigma' \cup \{B\} \cup \text{alphabet Entouré}\end{aligned}$$

**Principe de fonctionnement de la MTLB :**

- vérifier que l'entrée commence par  $\#q_0x1x2...xn\#$
- vérifier que l'entrée termine par  $\# \dots q_{OUI} \dots \#$
- pour chaque paire successive  $\#u\#v\#$  avec  $u, v \in \Sigma'^* \setminus \{\#\}$ 
  - vérifier que  $u \vdash_M v$  en faisant des aller retours en marquant les position avec l'alphabet entouré.

MPCP est la variante de PCP dnas laquelle on est obligé de commencer par le premier domino ( $i_1 = 1$ ).

**Théorème** MPCP est indécidable

**Démonstration** Idée : comme pour  $\exists_{MTLB}$  on va construire un reconnaisseur de trace de MT.  $\langle M, x \rangle \mapsto$  jeu de dominos  $T(M, x)$  tel que  $T(M, x)$  à une solution si et seulement si  $\Phi(x) \neq \perp$  et même mieux la solution de  $T(\langle M, x \rangle)$  écrit une trace de  $M$  sur l'entrée  $x$ .

**Voici les dominos de  $T(\langle M, x \rangle)$**  INSERER ICI LES IMAGES DU  
**TABLEAU**  $T(\langle M, x \rangle)$  admet pour solutions les traces acceptables de  $M$  sur l'entrée  $x$ . MPCP est indécidable

## 9 Evaluation

2 CC

- 1h30 calculabilite mer. 6 Mars
- construire une MT

- expliquer une MT
  - démontrer qu'un problème est décidable.
  - ensuite une correction
  - 1h30 de complexité
    - raisonner sur les classes
    - démontrer qu'un problème est NP-complet
  - ensuite une correction
- et pour finir une CT de 2h sur tout

## 10 Corollaire

PCP est indécidable , c'est une sorte de dominos qui se collent les uns des autres.

**Démonstration** On va transformer récursivement une instance donnée de MPCP  $\tau$  en une instance  $\tau'$  de PCP de sorte que  $\tau$  possède une solution si et seulement si  $\tau'$  en possède une.

Dans MPCP, on est obligé de poser en premier le premier domino de la liste. Il existe donc des solution de PCP qui ne sont pas solution de MPCP.

$$\left[\frac{aa}{a}\right] \left[\frac{bb}{ab}\right] \left[\frac{b}{bb}\right] \left[\frac{bb}{b}\right]$$

$$\left[\frac{aa}{a}\right] \left[\frac{bb}{ab}\right] \left[\frac{b}{bb}\right] \text{ est une solution MPCP}$$

$$\left[\frac{b}{bb}\right] \left[\frac{bb}{b}\right] \text{ solution de PCP , mais pas MPCP}$$

**idée :**  $\left[\frac{*a*a}{*a*}\right] \left[\frac{*b*b}{a*b*}\right] \left[\frac{*b}{b*b*}\right] \left[\frac{*b*b}{b*}\right] \left[\frac{*\$}{\$}\right]$  comme ça on est obligé de prendre la première tuile en premier et celle avec des dollars en dernier.

$$\forall u \in \Sigma^*$$

- $\bullet u = *u1 * u2... * un$
- $u\bullet = u1 * u2 * ...un*$
- $\bullet u\bullet = *u\bullet = \bullet u*$

À partir de  $\tau = (u1, v1), ..., (un, vn)$

On construit  $\tau'$  qui contient les  $n + 2$  dominos :

- $(\bullet u_1, \bullet v_1 \bullet)$ ,
- $\forall i \in 1, .., n (\bullet u_i, v_i \bullet)$ ,
- $(\bullet \$, \$)$

avec  $\$$  et  $*$  des nouvelles lettres

Une solution PCP de  $\tau'$  commence nécessairement par  $(\bullet u_1, \bullet v_1 \bullet)$  et se termine par  $(\bullet \$, \$)$ .

une plus courte solution contient entre ces tuiles uniquement  $(\bullet u_i, v_i \bullet)$

Les lettres en position impaires sont des  $*$  , en position paires des lettres de  $u_i/v_i$

$\tau$  possède une solution de MPCP si  $\tau$  possède une solution de PCP.

La transformation de  $\tau$  en  $\tau'$  est totale et calculable donc PCP est indécidable.

**Réductions** pour montrer qu'un problème est indécidable on a souvent construit une transformation calculable totale d'une instance A en instance B qui préserve les solutions.

**Définition** Soient A et B deux problèmes/langages  $A, B \subseteq \Sigma^*$ , on notera  $A \leq_m B$  si il existe une fonction  $F : \Sigma^* \rightarrow \Sigma^*$  calculable et totale telle que :  
 $\forall x \in \Sigma^* x \in A \iff F(x) \in B$  On dit que A est plus simple que B, que B est plus compliqué que A ou A est réductible à B.

**Proposition**  $\leq_m$  est un pré-ordre

**Démonstration**

- $\leq_m$  est réflexive :  $\forall A \subseteq \Sigma^* A \leq_m A$ , en effet l'identité (a vers a) est calculable et totale.
- $\leq_m$  est transitive :  $\forall A, B, C \subseteq \Sigma^* A \leq_m B$  et  $B \leq_m C \implies A \leq_m C$ 
  - si  $A \leq_m B$  il existe F calc.tot. tq  $\forall x \in \Sigma^* x \in A \iff F(x) \in B$
  - si  $B \leq_m C$  il existe G calc.tot. tq  $\forall x \in \Sigma^* x \in B \iff G(x) \in C$
  - Donc  $\forall x \in \Sigma^* x \in A \iff F(x) \in B \iff G(F(x)) \in C$ .
  - G o F qui est calculable totale réduit A à C.

**EX :**  $ARRET \leq_m MPCP \leq_m PCP$  donc  $ARRET \leq_m PCP$

**Proposition** si  $A \leq_m B$  et A indécidable, alors B est indécidable

**Démonstration** Soient  $A \leq_m B$  avec A indécidable.

supposons B décidable. Alors il existe une MT  $\mu$  totale qui reconnaît B.

Comme  $A \leq_m B$  il existe F calculable telle que  $\forall x \in \Sigma^* x \in A \iff F(x) \in B$

alors il existe une MT F tot qui calcule f.

À partir de F et  $\mu$  on construit une MT  $\alpha$  qui reconnaît A :

- sur l'entrée x
- $\alpha$  applique F pour obtenir f(x)
- puis  $\alpha$  appelle  $\mu$  sur l'entrée f(x)

Par construction  $\forall x \in \Sigma^* x \in A \iff \Phi_A(x) = OUI$

$\alpha$  est totale, on en déduit que A est décidable, CONTRADICTION!! Donc B est indécidable.

**Proposition** si  $A \leq_m B$  alors  $\neg A \leq_m \neg B$ .

**Idée**  $\forall x \in \Sigma^* x \in A \iff f(x) \in B$

donc  $\forall x \notin \Sigma^* x \in A \iff f(x) \notin B$

$\forall x \in \Sigma^* x \in \neg A \iff f(x) \in \neg B$

**Proposition** si  $\neg A \leq_m \neg B$  et B est rec.enum, alors A est rec.enum

**Proposition** si  $!A \leq_m !B$  et  $A$  n'est pas rec.enum alors  $B$  n'est pas rec.enum  
 $ARRET$  rec.enum et pas co.rec.enum.  
 $!ARRET$  co.rec.enum et non-rec.enum

### EQTM : égalité entre machines de Turing

- Entrée :  $\langle M1, M2 \rangle$  paire de MT
- question : Est ce que  $L(M1) = L(M2)$  ?

**Théorème** EQTM n'est ni rec.Enum , si co-rec.enum.

**Proposition** si  $A \leq_m B$  et  $A$  non rec.enum alors  $b$  non rec.enum

**Corollaire** si  $ARRET \leq_m !A$  alors  $A$  est non rec.enum.

**Démonstration** On sais que  $!ARRET$  n'est pas rec.enum , et si  $!ARRET \leq_m A$  alors  $a$  non rec.enum , on a  $ARRET \leq_m !A$  ssi  $!ARRET \leq_m A$

### Démonstration du théorème

- montrons que  $ARRET \leq_m !EQTM$  , i.e EQTM non rec.enum.  
 À  $\langle M, x \rangle$  on associe récursivement la paire
  - $M1$  : le MT qui calcule  $\perp$
  - $M2$  :
    - sur l'entrée  $y$
    - simuler  $M$  sur  $x$
- Montrons que  $ARRET \leq_m EQTM$ , i.e  $!EQTM$  non rec.enum.  
 À  $\langle M, x \rangle$  on associe récursivement la paire
  - $M1$  : la MT qui s'arrête tout le temps et répond CAKE.
  - $M2$  :
    - sur l'entrée  $y$
    - simuler  $M$  sur  $x$
    - répondre CAKE.

**Définition**  $A \cong$  si  $A \leq_m B$  et  $BT \leq_m A$

## 11 Points Fixes et récursion

**Paradoxe** Les êtres vivants sont des machines.  
 Les êtres vivants sont capables de s'(auto)-reproduire.  
 Les machines ne peuvent pas se reproduire.  
 Ce dernier point est faux.

### 11.1 Auto-référence

**proposition** Il existe une MT  $T$  qui produit, quelle que soit l'entrée ne description d'elle même.



**lemme 1 :** Il existe une fonction calculable  $q : \Sigma^* \rightarrow \Sigma^*$  totale que à  $x$  associe une MT  $P_x$  qui sur toute entrée produit  $x$  et s'arrête.

**Démonstration** la MT  $Q$  suivante calcule  $q$  :

- Sur l'entrée  $x$  :
  - soit  $P_x$  la MT suivante
    - sur l'entrée  $y$  :
      - effacer  $y$
      - écrire  $x$
      - s'arrêter
  - écrire  $P_x$
  - s'arrêter

**Démonstration**

$S$  est décomposée en deux MT  $A$  et  $B$  exécutées l'une après l'autre :

- $A$  produit une description de  $B$
- $B$  combine une description de  $A$  et de  $B$ .
- cela produit une description de  $A.B$  i.e  $S$

**A** c'est exactement  $P_B$ .

**B** on s'interdit d'utiliser  $A$  dans la définition de  $B$ , sinon auto-référence et problème. Par contre on a le droit d'utiliser l'entrée de  $B$ , c'est à dire la sortie de  $A$  c'est à dire une description de  $B$ . À partir de  $B$ , on calcule  $q(B) = P_B = A$ , on le combine à  $B$  : c'est  $S$  ! On l'écrit et on arrête.

**la machine A** c'est  $P_B$

**la machine B**

- Sur l'entrée  $M$ 
  - calculer  $q(M)$
  - combiner  $q(M)$  et  $M$
  - écrire la MT obtenue
  - s'arrêter.

$S$  = Combinaison de  $A$  et  $B$ , ou  $S$  = résultat de l'exécution de  $B$  sur  $B$ .

À l'exécution,  $S$  se comporte ainsi :

- 1 On exécute  $A$ , i.e, écrit  $B$  sur le ruban.
- 2 On exécute  $B$  sur la sortie de  $A$ , i.e la description de  $B$ .
  - calculer  $q(B)$
  - Combiner  $q(B)$  et  $B$ , i.e  $S$
  - écrire  $S$

## 11.2 Théorème de récursion :

**Théorème** Soit  $T$  une machine de Turing qui calcule une fonction  $t$  : qui prend deux paramètres :  $\Sigma^* x \Sigma^* \rightarrow \Sigma^*$ . Il existe une MT  $R$  qui calcule  $r : \Sigma^* \rightarrow \Sigma^*$  qui vérifie :

$$\forall x \in \Sigma^* \quad r(x) = t(\langle R, x \rangle)$$

**Démonstration :** Même schéma de preuve que précédemment. c'est  $B$  qui fait tout le travail.

**B :**

- sur l'entrée  $\langle B, T \rangle, x \rangle$  :
  - Calculer  $A = q'(\langle B, T \rangle)$
  - combiner  $A, B$ , et  $T \rightarrow R$
  - exécuter  $T$  sur l'entrée  $\langle R, x \rangle$

$$R \cong P'_{\langle B, T \rangle} \rightarrow B \rightarrow T \rightarrow \text{ruban}$$

**Moralité :** On a le droit maintenant de décrire des MT en ajoutant «obtenir une description de  $M$ » dans la description de  $M$ .

ex :

- $S$  sur l'entrée  $x$  :
  - obtenir une description de  $S$
  - l'écrire
  - s'arrêter

ex : Une preuve alternative de l'indécidabilité de l'arrêt.

- Supposons l'arrêt décidable par une MT  $\tilde{h}$
- on construit alors une MT  $\wp$  comme suit
  - sur l'entrée  $x$  :
    - Obtenir une description de  $P$
    - exécuter  $\tilde{h}$  sur l'entrée  $\langle P, x \rangle$
    - si la réponse est OUI alors boucler
    - si la réponse est NON s'arrêter

Sur l'entrée  $x$ , par construction  $P$  s'arrête si et seulement si  $\tilde{h}(\langle P, x \rangle) = \text{NON}$ , ie si et seulement si  $P$  ne s'arrête pas sur l'entrée  $x$ . CONTRADICTION!!!

$\tilde{h}$  n'existe pas, l'arrêt est indécidable.

Pour la saison 2 : aller voir la référence qui s'appelle Classical Recursive Theory

## 12 Notions

Machines de Turing à 1 ruban ,déterministe.  
Fonction calculable  $f : \Sigma^* \rightarrow \Sigma^*$  (partielle)  
Langage  $L \subseteq \Sigma^*$  récursif, reconnu par une MT TOTALE au sens fort  
Langage récursivement énumérable, reconnu au sens faible  
Problème décidable , langage récursif  
Problème indécidable, langage non-récursif  
On a montré que le problème de l'arrêt est indécidable  
Théorème de Rice : tout problème qui caractérise les MT dont la fonction satisfait une certaine propriété est soit trivial, soit indécidable  
Notion de réduction ,  $\leq_m$ . Pour montrer P indécidable, on montre  $ARRET \leq_m P$ .  
Théorème de récursion.