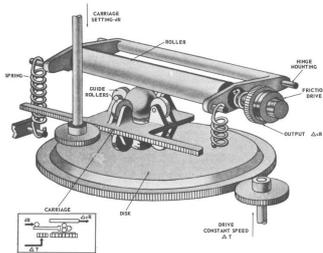


Cours 5: Thèse de Church. Indécidabilité.



Olivier Bournez
bournez@lix.polytechnique.fr

Ecole Polytechnique
INF412

1

Exprimez vous!!



Page du cours.



Commentaires, avis
sur les cours et les PCs.

■ Séances

- 4: (possibilité de la mise en place d'un tutorat).
- 5: demain, mercredi: repas avec les délégués.
- 6: mercredi dans une semaine: PC notée.

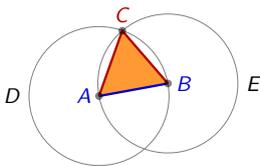
- Exprimez des commentaires, avis sur les cours et les PCs:
email à bournez@lix.polytechnique.fr, ou
www.enseignement.polytechnique.fr/informatique/INF412/AVIS.

2

Objectif

- Objectif de ce cours : répondre à la question suivante :

Qu'est-ce qu'un algorithme ?



Exemple d'algorithme :
Pour construire un *triangle équi-*
latéral ayant pour côté AB : tracer le cercle de centre A de rayon
AB ; tracer le cercle de centre B de rayon AB. Nommer C l'une des
intersections de ces deux cercles.
Le triangle ABC est la solution re-
cherchée.

3

Thèse de Church

- Historiquement, plusieurs modèles :

- ▶ Fonctions récursives, Kurt Gödel, 1931-34.
- ▶ λ -calcul, Alonzo Church, 1936.
- ▶ Machines de Turing, Alan Turing, 1936.
- ▶ Systèmes de Post,
- ▶ Machines RAM,
- ▶ Programmes JAVA, C, CAML, ...
- ▶ ...

- Il s'avère que ces modèles, très différents, ont exactement la même puissance

- ▶ **Thèse de Church/Turing** : une fonction est *calculable* si et seulement si elle est calculable par machine de Turing.

- (exprimée clairement pour la première fois par Stephen Kleene, étudiant en thèse d'Alonzo Church.)

4

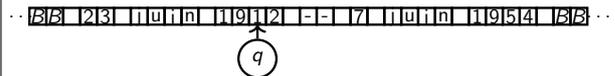
Parenthèse historique



- Théorie de la relativité : 1907/1915
- Théorème d'incomplétude : 1931

5

Alan M. Turing



- Machines de Turing : 1936

6

Parenthèse historique

- Ces modèles sont nés des suites du théorème d'incomplétude de Kurt Gödel :
- Et en fait de la question suivante "*Entscheidungsproblem*" :

Peut-on décider mécaniquement
si un énoncé est démontrable ou non ?

- Il s'avère que préciser "mécaniquement", c'est formaliser la notion d'algorithme.

7

Thèse de Church

- Thèse de Church:

Calculable dans un sens intuitif
correspond à
calculable par machine de Turing

8

Robustesse du modèle

- Le modèle de la machine de Turing est extrêmement robuste.
- On peut en effet envisager de nombreuses variantes autour du concept de machine de Turing,
 - ▶ mais chacune de ces variantes ne change pas ce que l'on arrive à programmer avec ces machines.

9

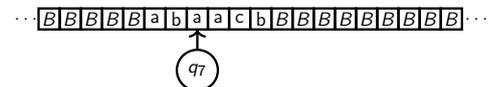
Restriction à un alphabet binaire

Théorème

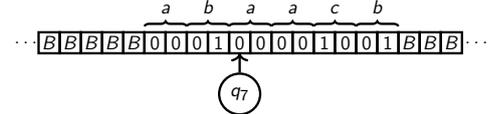
Toute machine de Turing qui travaille sur un alphabet Σ quelconque peut être simulée par une machine de Turing qui travaille sur un alphabet Σ avec uniquement deux lettres, avec $\Gamma = \Sigma \cup \{B\}$.

- Idée de la démonstration :

- ▶ Machine M sur l'alphabet $\{a, b, c\}$

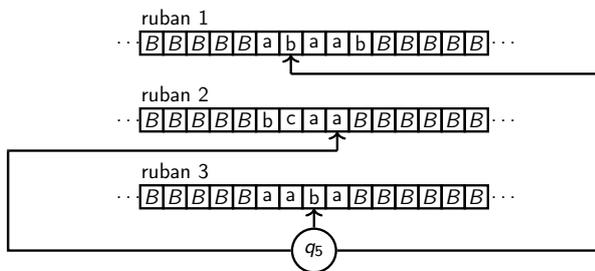


- ▶ Machine M' simulant M sur l'alphabet $\{0, 1\}$.



10

Machines de Turing à plusieurs rubans

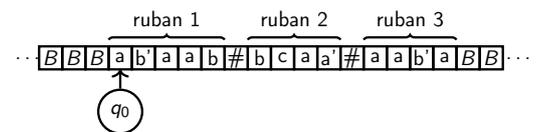


11

Théorème

Toute machine de Turing qui travaille avec k rubans peut être simulée par une machine de Turing avec un unique ruban.

- Idée de la démonstration :



12

- Le modèle de la machine de Turing peut paraître extrêmement rudimentaire.
- Il n'en demeure pas moins extrêmement puissant, et capable de capturer la notion de *calculable* en informatique.

13

Machines RAM (Random Access Machine)

- Un modèle de calcul qui ressemble beaucoup plus aux langages machine actuels, et à la façon dont fonctionnent les processeurs actuels.
 - ▶ (Similaire/Alternative à la Machine virtuelle du cours INF371)
- Une machine RAM possède des registres qui contiennent des entiers naturels (nuls si pas encore initialisés).
- Les instructions autorisées dépendent du processeur que l'on veut modéliser, mais elles incluent en général la possibilité de :
 1. copier le contenu d'un registre dans un autre ;
 2. l'adressage indirect : récupérer/écrire le contenu d'un registre dont le numéro est donné par la valeur d'un autre registre ;
 3. effectuer des opérations élémentaires sur un ou des registres, par exemple additionner 1, soustraire 1 ou tester l'égalité à 0 ;
 4. effectuer d'autres opérations sur un ou des registres, par exemple l'addition, la soustraction, la multiplication, division, les décalages binaires, les opérations binaires bit à bit.

14

Machines RISC (Reduced Instruction Set)

- Une machine RISC est une machine RAM dont les instructions sont uniquement de la forme :
 1. $x_0 \leftarrow 0$;
 2. $x_0 \leftarrow x_0 + 1$;
 3. $x_0 \leftarrow x_0 \ominus 1$;
 4. **if** $x_0 = 0$ **then** aller \ 'a l' instruction num\ 'ero j ;
 5. $x_0 \leftarrow x_j$;
 6. $x_j \leftarrow x_0$;
 7. $x_0 \leftarrow x_{x_j}$;
 8. $x_{x_0} \leftarrow x_j$.

15

Machines RISC vs Machines de Turing

Théorème

Toute machine RISC peut être simulée par une machine de Turing.

- (et réciproquement).

16

Idée de la démonstration

- La machine de Turing qui simule la machine RISC possède 4 rubans. Les deux premiers rubans codent les couples (i, x_i) pour x_i non nul. Le troisième ruban code l'accumulateur x_0 et le quatrième est un ruban de travail.
 - ▶ Pour un entier i , notons $\langle i \rangle$ son écriture en binaire.
- Plus précisément :
 - ▶ Le premier ruban code un mot de la forme

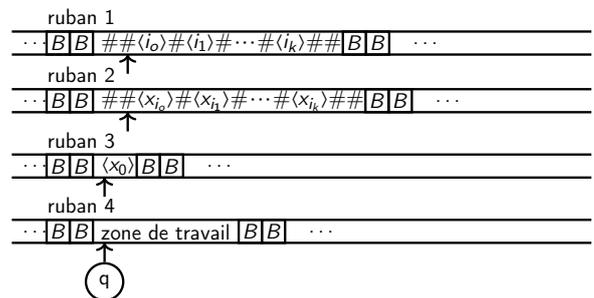
$$\#\#\langle i_0 \rangle \#\langle i_1 \rangle \#\dots \#\langle i_k \rangle \#\#\#;$$
 - ▶ Le second ruban code un mot de la forme

$$\#\#\langle x_{i_0} \rangle \#\langle x_{i_1} \rangle \#\dots \#\langle x_{i_k} \rangle \#\#\#;$$
 - ▶ Le troisième ruban code

$$\langle x_0 \rangle.$$
- On appelle **position standard** la position où les têtes de lecture des deux premiers rubans sont sur le deuxième $\#$, et la tête de lecture des autres rubans est tout à gauche.
- La simulation est décrite pour seulement 3 exemples.

17

$$x_0 := x_0 + 1$$



18

■ $x_0 \leftarrow x_0 + 1$:

1. on déplace la tête de lecture du ruban 3 tout à droite jusqu'à atteindre un symbole B .
2. On se déplace alors d'une case vers la gauche, et on remplace les 1 par des 0, en se déplaçant vers la gauche tant que possible.
3. Lorsqu'un 0 ou un blanc est trouvé, on le change en 1 et on se déplace (possiblement) à gauche pour revenir en position standard.

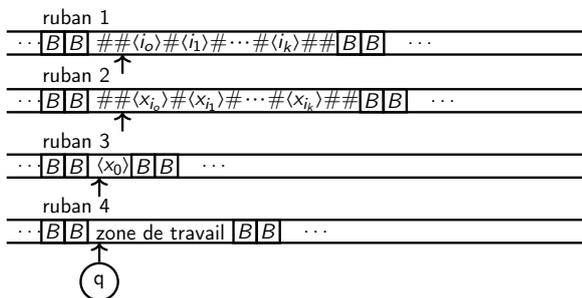
19

■ $x_{23} \leftarrow x_0$:

- ▶ on parcourt les rubans 1 et 2 vers la droite, bloc par bloc, jusqu'à atteindre la fin du ruban 1, ou ce que l'on lise un bloc $\#10111\#$ (10111 correspond à 23 en binaire).
- ▶ Si la fin du ruban 1 a été atteinte, alors l'emplacement 23 n'a jamais été vu auparavant. On l'ajoute en écrivant 10111 à la fin du ruban 1, et on recopie le ruban 3 (la valeur de x_0) sur le ruban 2. On retourne alors en position standard.
- ▶ Sinon, c'est que l'on a trouvé $\#10111\#$ sur le ruban 1. On lit alors $\langle x_{23} \rangle$ sur le ruban 2. Dans ce cas, il doit être modifié. On fait cela de la façon suivante :
 1. On copie le contenu à droite de la tête de lecture numéro 2 sur le ruban 4.
 2. On copie le contenu du ruban 3 (la valeur de x_0) à la place de x_{23} sur le ruban 2.
 3. On écrit $\#$, et on recopie le contenu du ruban 4 à droite de la tête de lecture du ruban 2, de façon à restaurer le reste du ruban 2.
 4. On retourne en position standard.

21

$$x_0 := x_{x_{23}}$$



22

■ $x_0 \leftarrow x_{x_{23}}$:

- ▶ En partant de la gauche des rubans 1 et 2, on parcourt les rubans 1 et 2 vers la droite, bloc par bloc, jusqu'à atteindre la fin du ruban 1, ou ce que l'on lise un bloc #10111# (10111 correspond à 23 en binaire).
- ▶ Si la fin du ruban 1 a été atteinte, on ne fait rien, puisque x_{23} vaut 0 et le ruban 3 contient déjà $\langle x_0 \rangle$.
- ▶ Sinon, c'est que l'on a trouvé #10111# sur le ruban 1. On lit alors $\langle x_{23} \rangle$ sur le ruban 2, que l'on recopie sur le ruban 4.
- ▶ Comme ci-dessus, on parcourt les rubans 1 et 2 en parallèle jusqu'à trouver # $\langle x_{23} \rangle$ # où atteindre la fin du ruban 1. Si la fin du ruban 1 est atteinte, alors on écrit 0 sur le ruban 3, puisque $x_{x_{23}} = x_0$.
- ▶ Sinon, on copie le bloc correspondant sur le ruban 1 sur le ruban 3, puisque le bloc sur le ruban 2 contient $x_{x_{23}}$, et on retourne en position standard.

23

Machines RAM vs Machines de Turing

- Dans une machine RAM, on peut autoriser aussi des instructions du type :
 1. $x_i \leftarrow x_j \text{ op } x_k$
 où op est une opération (addition, soustraction, ...)

- Avec le même principe :

Théorème

Toute machine RAM peut être simulée par une machine de Turing.

- ▶ (dès que les opérations op restent calculables par machine de Turing).

24

Machines de Turing : calcul de fonctions

- Fonctions sur les mots : $f : \Sigma^* \rightarrow \Sigma^*$.

- ▶ Une machine de Turing calcule la **fonction partielle** f , si pour tout mot $w \in \Sigma^*$ la machine s'arrête en acceptant avec son ruban qui contient $f(w)$ (entouré de B) si w est dans le domaine de f , et boucle ou refuse sinon.

- Fonctions $f : E \rightarrow F$:

- ▶ on fixe un codage des éléments de E et de F , et on exige que la fonction qui passe du codage de $e \in E$ au codage de $f(e) \in F$ soit calculable.

- ▶ Exemple :

- On peut coder $\vec{n} = (n_1, \dots, n_k) \in \mathbb{N}^k$ par $\langle \vec{n} \rangle = a^{n_1+1} b a^{n_2+1} b \dots a^{n_k+1}$.
- Une fonction $f : \mathbb{N}^k \rightarrow \mathbb{N}$ est calculable si elle est calculable sur ce codage.

25

Thèse de Church

■ Thèse de Church :

Calculable dans un sens intuitif
correspond à
calculable par machine de Turing

■ Conséquence :

- ▶ on va décrire des algorithmes "haut niveau" dans la suite (en français, JAVA, CAML, PYTHON, etc) ;
- ▶ en utilisant implicitement à chaque fois le fait qu'ils peuvent s'écrire comme des programmes de machines de Turing.

26

Interprètes

■ Un langage comme JAVA est interprété :

- ▶ un programme JAVA est compilé en un codage que l'on appelle **bytecode**.
- ▶ Lorsqu'on cherche à lancer ce programme, l'interprète JAVA simule ce bytecode.

■ Ce principe d'interprétation permet la portabilité :

- ▶ seul l'interprète dépend de la machine sur laquelle on exécute le programme.

27

■ Avec des machines de Turing, on peut aussi programmer des **interprètes**,

- ▶ c'est-à-dire des programmes qui prennent en entrée la description d'un autre programme et qui simulent l'exécution de ce programme.

■ Pour le voir,

- ▶ Il nous faut fixer une représentation des programmes des machines de Turing.

28

Codage d'une machine de Turing

■ Soit $M = (Q, \Sigma, \Gamma, B, \delta, q_0, q_a, q_r)$: une machine de Turing sur l'alphabet $\Sigma = \{0, 1\}$:

- ▶ On peut considérer que $Q = \{q_1, q_2, \dots, q_z\}$, avec la convention que $q_1 = q_0$, $q_2 = q_a$, $q_3 = q_r$, et que $\Gamma = \{X_1, X_2, \dots, X_s\}$, avec la convention que X_s est le symbole B , et que X_1 est le symbole 0 de Σ , et que X_2 est le symbole 1 de Σ .
- ▶ Pour $m \in \{\leftarrow, |, \rightarrow\}$, on définit $\langle m \rangle$ par $\langle \leftarrow \rangle = 1$, $\langle \rightarrow \rangle = 2$, $\langle | \rangle = 3$.
- ▶ Supposons que l'une des règles de transition du programme δ soit $\delta(q_i, X_j) = (q_k, X_l, m)$: le codage de cette règle est le mot $0^i 10^j 10^k 10^l 10^{\langle m \rangle}$ sur l'alphabet $\{0, 1\}$.

■ Un codage, noté $\langle M \rangle$, de la machine de Turing M est un mot de la forme

$$C_1 11 C_2 11 C_3 \dots C_{n-1} 11 C_n,$$

où chaque C_i est le codage d'une des règles de transition de δ .

29

Existence d'une machine de Turing universelle

- On peut construire un interpréteur, c'est-à-dire une **machine de Turing universelle** :

Théorème

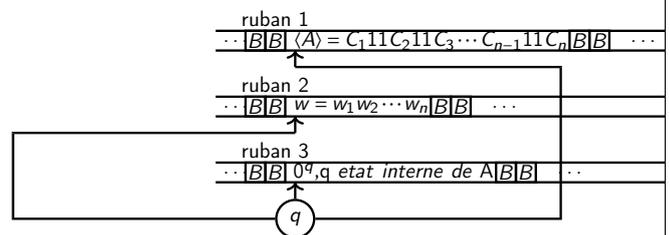
Il existe une machine de Turing M_{univ} telle, que sur l'entrée $\langle (M), w \rangle$ où :

- $\langle M \rangle$ est le codage d'une machine de Turing M ;
- $w \in \{0,1\}^*$,

la machine M_{univ} simule la machine de Turing M sur l'entrée w .

30

Principe : en autorisant 3 rubans



31

Problèmes de décision

- On va s'intéresser dorénavant presque uniquement aux problèmes dont la réponse est soit *vrai* soit *faux*.



- Un **problème de décision** \mathcal{P} est la donnée
 - d'un ensemble E , que l'on appelle **l'ensemble des instances**,
 - et d'un sous-ensemble E^+ de E , que l'on appelle **l'ensemble des instances positives**.
- La question à laquelle on s'intéresse est de construire (si cela est possible) une machine de Turing qui décide si une instance donnée est positive ou non :
 - c'est-à-dire, qui prend en entrée le codage d'un élément $e \in E$, et qui accepte si $e \in E^+$, et qui refuse si $e \notin E^+$.
 - notation : on écrira $e \in \mathcal{P}$ pour dire que $e \in E^+$.

32

Exemples

- Problème NOMBRE PREMIER:
 - Donnée:** Un entier naturel n .
 - Réponse:** Décider si n est premier.
- Problème CODAGE:
 - Donnée:** Un mot w .
 - Réponse:** Décider si w correspond au codage $\langle M \rangle$ d'une certaine machine de Turing M .
- Problème REACH:
 - Donnée:** Un triplet constitué d'un graphe G , d'un sommet u et d'un sommet v du graphe.
 - Réponse:** Décider s'il existe un chemin entre u et v dans le graphe G .

33

Remarque : problèmes vs langages

- On utilise indifféremment la terminologie problème ou langage,
 - ▶ car les deux concepts sont essentiellement les mêmes ;
 - ▶ formellement, à un langage correspond un problème (de décision) et réciproquement.
- Problèmes vers langages :
 - ▶ en effet, à un problème est associé généralement implicitement un codage : e est représenté par un mot $\langle e \rangle$ sur un alphabet Σ .
 - ▶ On peut donc voir E comme $\{w | w = \langle e \rangle, e \in E\}$.
 - ▶ Au problème de décision \mathcal{P} , correspond le langage $L(\mathcal{P}) = \{w | w = \langle e \rangle, e \in E^+\}$.
- Langages vers problèmes :
 - ▶ Réciproquement, on peut voir tout langage L comme le problème de décision :

Donnée : Un mot w .
Réponse : Décider si $w \in L$.

34

Problèmes décidables

Définition

- ▶ **Rappel :** Un langage $L \subset M^*$ est dit **décidable** s'il est décidé par une certaine machine de Turing.
- ▶ Un langage ou un problème décidable est aussi dit **récuratif**.
- ▶ Un langage qui n'est pas décidable est dit **indécidable**.

- On note D pour la classe des langages et des problèmes décidables.
- Par exemple :
 - ▶ Proposition. Les problèmes de décision NOMBRE PREMIER, CODAGE et REACH sont décidables.

35

Le langage universel

- On appelle **langage universel**, le problème de décision suivant :
 - ▶ Problème L_{univ} :

Donnée : • Le codage $\langle M \rangle$ d'une machine de Turing M
• et un mot w .

Réponse : Décider si la machine M accepte le mot w .

Théorème

Le problème L_{univ} n'est pas décidable.

36

■ Démonstration :

- ▶ Par l'absurde : si L_{univ} est décidé par une machine de Turing A , on peut alors construire une machine de Turing B qui fonctionne de la façon suivante :
 - B prend en entrée un mot $\langle C \rangle$ codant une machine de Turing C ;
 - B appelle la machine de Turing A sur la paire $(\langle C \rangle, \langle C \rangle)$ (c'est-à-dire sur l'entrée constituée du codage de la machine de Turing C , et du mot w correspondant aussi à ce même codage) ;
 - Si la machine de Turing A accepte ce mot, B refuse.
 - Si la machine de Turing A refuse ce mot, B accepte.
- ▶ Appliquons la machine de Turing B sur le mot $\langle B \rangle$, c'est-à-dire sur le mot codant la machine de Turing B :
 - Si B accepte le mot $\langle B \rangle$, cela signifie, par définition de L_{univ} et de A , que A accepte $(\langle B \rangle, \langle B \rangle)$. Mais si A accepte ce mot, B est construit pour refuser son entrée $\langle B \rangle$. Contradiction.
 - Si B refuse le mot $\langle B \rangle$, cela signifie, par définition de L_{univ} et de A , que A refuse $(\langle B \rangle, \langle B \rangle)$. Mais si A refuse ce mot, B est construit pour accepter son entrée $\langle B \rangle$. Contradiction.

37

Problèmes semi-décidables

Théorème

Le problème L_{univ} est toutefois semi-décidable :

- ▶ Un langage $L \subset \Sigma^*$ est dit **semi-décidable** (ou encore **récursivement énumérable**) s'il correspond à l'ensemble des mots acceptés par une machine de Turing.
- ▶ On note RE la classe des langages et des problèmes semi-décidables.
- Preuve :
 - ▶ Sur l'entrée $(\langle M \rangle, w)$, il suffit de simuler la machine de Turing M sur l'entrée w .
 - On arrête la simulation et on accepte si l'on détecte dans cette simulation que la machine de Turing M atteint son état d'acceptation.
 - Sinon, on simule M pour toujours.

38

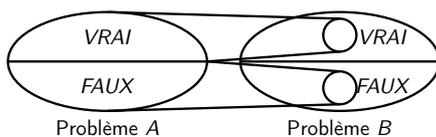
- Nous connaissons deux langages indécidables,
 - ▶ L_{univ} et son complémentaire.
- Notre but est maintenant d'en obtenir d'autres, et de savoir comparer les problèmes.
- Nous introduisons pour cela la notion de **réduction**.

39

La notion de réduction

- Soient A et B deux problèmes d'alphabets respectifs M_A et M_B . Une **réduction de A vers B** est une fonction $f : M_A^* \rightarrow M_B^*$ calculable telle que

$$w \in A \text{ ssi } f(w) \in B.$$



- On note $A \leq_m B$ lorsque A se réduit à B .
 - ▶ intuitivement : $A \leq_m B$ signifie que A est plus facile que B .

40

Principales propriétés

Théorème

\leq_m est un préordre (= est reflexive, transitive) :

1. $L \leq_m L$;
2. $L_1 \leq_m L_2, L_2 \leq_m L_3$ impliquent $L_1 \leq_m L_3$.

- intuitivement : un problème est aussi facile (et difficile) que lui-même, et la relation "être plus facile que" est transitive.

Théorème

Si $A \leq_m B$, et si B est décidable alors A est décidable

- intuitivement : si un problème est plus facile qu'un problème décidable, alors il est décidable.

Théorème

Si $A \leq_m B$, et si A est indécidable, alors B est indécidable.

- intuitivement : si un problème est plus difficile qu'un problème indécidable, alors il est indécidable.

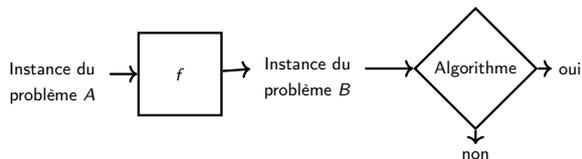
41

■ Preuve du premier théorème :

- ▶ Considérer la fonction identité pour f pour le premier point. Pour le second point, supposons $L_1 \leq_m L_2$ via la réduction f , et $L_2 \leq_m L_3$ via la réduction g . On a $x \in L_1$ ssi $g(f(x)) \in L_3$. La composée de deux fonctions calculables est calculable.

■ Preuve du second théorème :

- ▶ A est décidé par la machine de Turing qui, sur une entrée w , calcule $f(w)$, puis simule la machine de Turing qui décide B sur l'entrée $f(w)$. Puisqu'on a $w \in A$ si et seulement si $f(w) \in B$, la machine de Turing est correcte.



■ Le troisième théorème est la contraposée du second.

42

- Cette idée permet d'obtenir immédiatement la preuve de l'indécidabilité de plein d'autres problèmes.

■ Stratégie :

- ▶ pour prouver que B est indécidable, on prouve que $A \leq_m B$ pour un certain problème A déjà connu comme indécidable.

43

Exemple 1 : Le problème de l'arrêt des machines de Turing

Il n'est pas possible de déterminer algorithmiquement si une machine de Turing s'arrête.

■ Problème HALTING-PROBLEM:

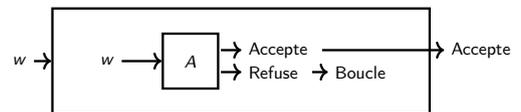
Donnée: Le codage $\langle M \rangle$ d'une machine de Turing M et une entrée w .

Réponse: Décider si M s'arrête sur l'entrée w .

Proposition Le problème HALTING-PROBLEM est indécidable.

44

Preuve : $L_{\text{univ}} \leq_m \text{HaltingProblem}$



- On construit une réduction de L_{univ} vers le problème de l'arrêt : Pour chaque couple $(\langle A \rangle, w)$, on considère la machine de Turing B définie de la façon suivante :
 - ▶ B prend en entrée un mot w ;
 - ▶ B simule A sur w ;
 - ▶ Si A accepte w , alors B accepte. Si A rejette w , alors B boucle (possiblement B simule A pour toujours, si A ne s'arrête pas).
- La fonction f qui envoie $(\langle A \rangle, w)$ sur $(\langle B \rangle, w)$ est calculable.
- De plus, on a $(\langle A \rangle, w) \in L_{\text{univ}}$ si et seulement si B s'arrête sur w , c'est-à-dire $(\langle A \rangle, w) \in \text{Halting Problem}$.

45

Exemple 2

Il n'est pas possible de déterminer algorithmiquement si une machine de Turing accepte au moins une entrée :

■ Problème L_\emptyset :

Donnée: Le codage $\langle M \rangle$ d'une machine de Turing M .

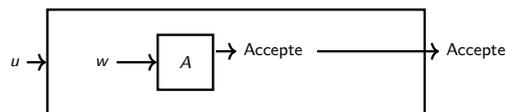
Réponse: Décider si $L(M) \neq \emptyset$.

Proposition

Le problème L_\emptyset est indécidable.

46

Démonstration



■ On construit une réduction de L_{univ} vers L_\emptyset : pour toute paire $(\langle A \rangle, w)$, on considère la machine de Turing A_w définie de la manière suivante :

- ▶ A_w prend en entrée un mot u ;
- ▶ A_w simule A sur w ;
- ▶ Si A accepte w , alors A_w accepte.

■ La fonction f qui à $(\langle A \rangle, w)$ associe $\langle A_w \rangle$ est bien calculable.

■ De plus on a $(\langle A \rangle, w) \in L_{\text{univ}}$ si et seulement si $L(A_w) \neq \emptyset$, c'est-à-dire $\langle A_w \rangle \in L_\emptyset$:

- ▶ en effet, A_w accepte soit tous les mots (et donc le langage correspondant n'est pas vide) si A accepte w , soit n'accepte aucun mot (et donc le langage correspondant est vide) sinon.

47

- Beaucoup d'exemples (dont l'exemple 1) peuvent être vus comme les conséquences d'un résultat très général

48

Théorème de Rice

Théorème (Théorème de Rice)

Toute propriété non triviale des langages semi-décidables est indécidable.

■ Autrement dit, soit une propriété P des langages semi-décidables non triviale,

- ▶ c'est-à-dire telle qu'il y a au moins une machine de Turing M telle que $L(M)$ satisfait P et une machine de Turing M telle que $L(M)$ ne satisfait pas P .

■ Alors le problème de décision L_P :

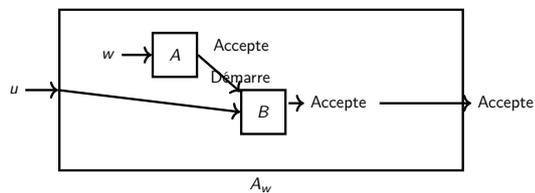
Donnée: Le codage $\langle M \rangle$ d'une machine de Turing M ;

Réponse: Décider si $L(M)$ vérifie la propriété P ;
est indécidable.

49

Démonstration graphique

- Il nous faut démontrer que le problème de décision L_P est indécidable.
 - ▶ Quitte à remplacer P par sa négation, on peut supposer que le langage vide ne vérifie pas la propriété P .
- Puisque P est non triviale, il existe un moins une machine de Turing B avec $L(B)$ qui vérifie P .



50

■ Démonstration :

- ▶ Il nous faut démontrer que le problème de décision L_P est indécidable.
 - Quitte à remplacer P par sa négation, on peut supposer que le langage vide ne vérifie pas la propriété P (prouver l'indécidabilité de L_P est équivalent à prouver l'indécidabilité de son complémentaire).
- ▶ Puisque P est non triviale, il existe un moins une machine de Turing B avec $L(B)$ qui vérifie P .
- ▶ On construit une réduction de L_{univ} vers le langage L_P . Étant donnée une paire $(\langle A \rangle, w)$, on considère la machine de Turing A_w définie de la façon suivante :
 - A_w prend en entrée un mot u ;
 - Sur le mot u , A_w simule A sur le mot w ;
 - Si A accepte w , alors A_w simule B sur le mot u : A_w accepte si seulement si B accepte u .
- ▶ Autrement dit, A_w accepte, si et seulement si A accepte w et si B accepte u . Si w est accepté par A , alors $L(A_w)$ vaut $L(B)$, et donc vérifie la propriété P . Si w n'est pas accepté par A , alors $L(A_w) = \emptyset$, et donc ne vérifie pas la propriété P .
- ▶ La fonction f qui à $(\langle A \rangle, w)$ associe $\langle A_w \rangle$ est bien calculable.

51