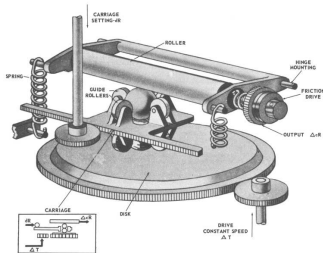


Cours 6: Calculabilité.



Olivier Bournez
bournez@lix.polytechnique.fr

INF412

Ecole Polytechnique
CSC_INF41012_EP

1

Rappel

- Demain, Mercredi 2 Octobre. PC Notée
 - ▶ 8h07-10h07 : groupes 8h00.
 - ▶ 10h08-12h08 : groupes 10h15.
- Salle habituelles
- Tous les documents sont autorisés.
- Programme : logique, calculabilité = INF412 jusqu'à aujourd'hui.

2

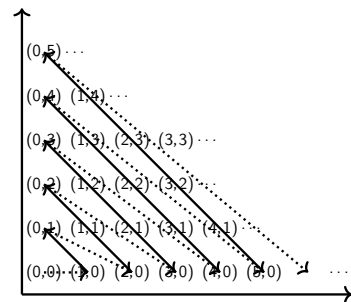
Conséquence

- On a établi : Il existe des problèmes de décision qui ne sont pas décidables.
 - ▶ Une preuve plus simple ?

3

Ensembles dénombrables

- \mathbb{N}^2 est dénombrable :

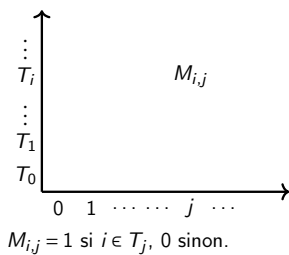


- Σ^* (les mots finis sur l'alphabet Σ) est dénombrable, lorsque Σ est dénombrable :
 - ▶ par exemple, on peut coder une suite finie $u_0 u_1 u_2 \dots$ par $2^{u_0} 3^{u_1} 5^{u_2} \dots$
- $\mathbb{N} \times \Sigma^*$ est dénombrable.

4

Ensembles non-dénombrables : argument de Cantor

- $\mathcal{P}(\mathbb{N})$ n'est pas dénombrable.



$M_{i,j} = 1$ si $i \in T_j$, 0 sinon.

- L'ensemble des langages sur un alphabet Σ^* n'est pas dénombrable, même si Σ est fini.

- ▶ Par l'absurde, si $\mathcal{P}(\mathbb{N}) = \{T_0, T_1, T_2, \dots\}$, on pourrait considérer $T^* = \{j \mid M_{j,j} = 0\}$.
- ▶ Cette partie de \mathbb{N} n'est pas dans l'énumération, car sinon elle devrait avoir un numéro j_0 :
 - si $j_0 \in T^*$, alors on devrait avoir $M_{j_0, j_0} = 1$ par définition de M , et $M_{j_0, j_0} = 0$ par définition de T^* : impossible.
 - Si $j_0 \notin T^*$, alors on devrait avoir $M_{j_0, j_0} = 0$ par définition de M , et $M_{j_0, j_0} = 1$ par définition de T^* : impossible.

5

Conséquence

Corollaire

Il existe des problèmes de décision qui ne sont pas décidables.

- Preuve : Il y a un nombre non dénombrable de problèmes de décision, et un nombre dénombrable de machines de Turing.
- C'est grave ?

6

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.

7

- 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.

8

Problèmes semi-décidables

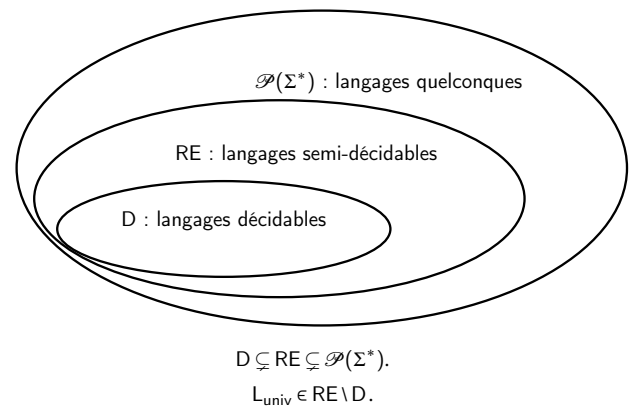
Théorème

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.

9

Le paysage de la calculabilité



10

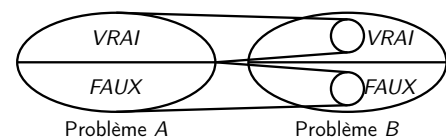
- Nous connaissons un langage indécidable :
 - ▶ L_{univ} .
- Notre but est maintenant d'en obtenir d'autres, et de savoir comparer les problèmes.
- Nous introduisons pour cela la notion de **réduction**.

11

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 .

12

Principales propriétés

Théorème

\leq_m est un préordre (= est réflexive, 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.

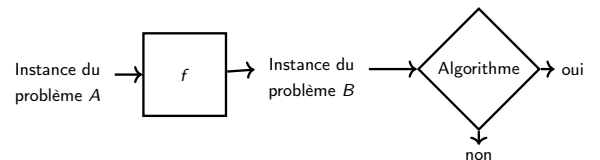
13

- 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 $g(f(x)) \in L_2$. 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.

14

- 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.

15

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:

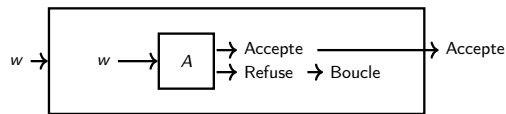
Donnée: Le codage $\langle M \rangle$ d'une machine de Turing M et une entrée w .

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

Proposition Le problème HALTING-PROBLEM est indécidable.

16

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 $(\langle A \rangle, w) \in \text{Halting Problem}$.

17

Exemple 2

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

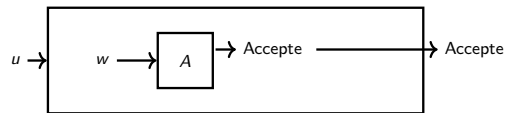
- Problème L_\emptyset :
 - Donnée :** Le codage $\langle M \rangle$ d'une machine de Turing M .
 - Réponse :** Décider si $L(M) \neq \emptyset$.

Proposition

Le problème L_\emptyset est indécidable.

18

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.

19

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

20

Théorème de Rice

Théorème (Théorème de Rice)

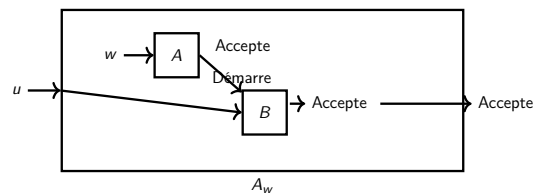
Toute propriété non triviale des langages semi-décidables est indécidable.

- 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.

21

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 .



22

- 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 (A, 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 (A, w) associe $\langle A_w \rangle$ est bien calculable.

23

Exemple 2 (déjà vu)

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.

24

Démonstration

C'est une application directe du théorème de Rice.

25

Exemple 2

Il n'est pas possible de déterminer algorithmiquement si une machine de Turing accepte le mot "informatique"

■ Problème L_2 :

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

Réponse: Décider si *informatique* $\in L(M)$.

Proposition

Le problème L_2 est indécidable.

26

Démonstration

C'est une application directe du théorème de Rice.

27

Exemple 3

■ Problème L_{\neq} :

Donnée: Le codage $\langle A \rangle$ d'une machine de Turing A et le codage $\langle A' \rangle$ d'une machine de Turing A' .

Réponse: Déterminer si $L(A) \neq L(A')$.

Proposition

Le problème L_{\neq} est indécidable.

28

Démonstration : $L_\emptyset \leq_m L_\neq$

- On construit une réduction de L_\emptyset vers L_\neq .
 - ▶ On considère une machine de Turing fixe B qui accepte le langage vide :
 - prendre par exemple une machine de Turing B qui rentre immédiatement dans une boucle sans fin.
 - ▶ La fonction f qui à $\langle A \rangle$ associe la paire $(\langle A \rangle, \langle B \rangle)$ est bien calculable.
 - ▶ De plus on a $\langle A \rangle \in L_\emptyset$ si et seulement si $L(A) \neq \emptyset$ si et seulement si $(\langle A \rangle, \langle B \rangle) \in L_\neq$.

29

Exemple 4

- Problème *INF412*:
 - Donnée:** Le codage $\langle A \rangle$ d'une machine de Turing A .
 - Réponse:** Déterminer si A accepte le mot "INF412" en moins de 412 étapes.

Proposition

Le problème *INF412* est ?

30

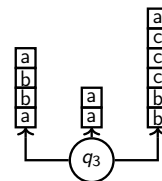
Indécidable ?

Le problème est décidable : il suffit de simuler la machine pendant 412 étapes.

31

Machines à k piles

- Une **machine à k piles**, possède un nombre fini k de piles r_1, r_2, \dots, r_k , qui correspondent à des piles d'éléments de Σ .
- Les instructions d'une machine à piles permettent seulement
 - ▶ d'empiler un symbole sur l'une des piles,
 - ▶ tester la valeur du sommet d'une pile,
 - ▶ ou dépiler le symbole au sommet d'une pile.



32

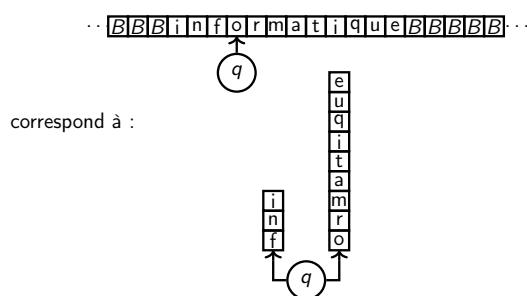
Théorème

Toute machine de Turing peut être simulée par une machine à 2 piles.

(et réciproquement).

33

- Idée de la démonstration : voir une machine de Turing comme une machine à 2-piles.



34

Machines à compteurs

- Machines à compteurs :

- ▶ une **machine à compteurs** possède un nombre fini k de compteurs r_1, r_2, \dots, r_k , qui contiennent des entiers naturels.
- ▶ Les instructions d'une machine à compteurs permettent seulement
 - de tester l'égalité d'un des compteurs à 0;
 - d'incrémenter un compteur;
 - ou de décrémenter un compteur.

(tous les compteurs sont initialement nuls, sauf celui codant l'entrée).

35

Machines à compteurs

Instructions :

- ▶ $\text{Inc}(c, j)$: incrémente compteur c puis va à l'instruction j .
- ▶ $\text{Decr}(c, j)$: décrémente compteur c puis va à l'instruction j .
- ▶ $\text{IsZero}(c, j, k)$ teste si le compteur c est nul et va à l'instruction j si c'est le cas, et à l'instruction k sinon.
- ▶ Halt arrête le calcul.

Exemple de programme avec 3 compteurs :

1. ✓ IsZero(1,5,2)
2. ✓ Decr(1,3)
3. ✓ Inc(3,4)
4. ✓ Inc(3,1)
5. ✓ Halt

- Exemple d'exécution : compteurs (3,2,0) (3,2,0) (2,2,0) (2,2,1) (2,2,2) (1,2,2) (1,2,3) (1,2,4) (0,2,4) (0,2,5) (0,2,6) (0,2,6)

36

Théorème

Toute machine à k -piles peut être simulée par une machine à $k+1$ compteurs.

37

■ Principe :

- ▶ voir une pile (donc un mot) $w = a_1 a_2 \dots a_n$ sur l'alphabet $\Sigma = \{0, 1, \dots, r-1\}$ comme l'entier $i = a_n r^{n-1} + a_{n-1} r^{n-2} + \dots + a_2 r + a_1$.
- ▶ Dépiler correspond à remplacer i par $i \text{ div } r$. Empiler le symbole a correspond à remplacer i par $i * r + a$. Lire le sommet d'une pile i correspond à calculer $i \text{ mod } r$.
- ▶ Par exemple, pour $i \text{ div } r$: en partant avec le compteur supplémentaire (celui d'indice $k+1$) à 0, on décrémente le compteur i de r et on incrémente le compteur supplémentaire de 1. On répète cette opération jusqu'à ce que le compteur i atteigne 0. On décrémente alors le compteur supplémentaire de 1 en incrémentant le compteur i de 1 jusqu'à ce que le premier soit 0.

38

Théorème

Toute machine à $k \geq 3$ compteurs se simule par une machine à 2 compteurs.

Corollaire

Toute machine de Turing se simule par une machine à 2 compteurs.

(et réciproquement).

39

■ Principe :

- ▶ Supposons $k=3$. L'idée est coder trois compteurs i, j et k par l'entier $m = 2^i 3^j 5^k$. L'un des compteurs stocke cet entier. L'autre compteur est utilisé pour faire des multiplications, divisions, calculs modulo m , pour m valant 2, 3, ou 5, comme dans la preuve précédente.
- ▶ Pour $k > 3$, on utilise le même principe, mais avec les k premiers nombres premiers.

40

Résultats obtenus

■ Résumé :

- ▶ Les modèles suivants se simulent deux à deux :
 - Les machines de Turing
 - Les machines à $k \geq 2$ piles
 - Les machines RAM
 - Les machines à $k \geq 2$ compteurs

41

Thèse de Church

■ Thèse de Church:

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

42

Fonctions récursives

■ Une fonction partielle $f : \mathbb{N}^n \rightarrow \mathbb{N}$ est *récursive* si elle est soit la constante 0, soit l'une des fonctions :

- ▶ Zero : $x \mapsto 0$ la fonction 0 ;
- ▶ Succ : $x \mapsto x + 1$ la fonction successeur ;
- ▶ Proj $_n^i$: $(x_1, \dots, x_n) \mapsto x_i$ les fonctions de projection, pour $1 \leq i \leq n$;
- ▶ Comp $_m(g, h_1, \dots, h_m)$: $(x_1, \dots, x_n) \mapsto g(h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n))$ la composition des fonctions récursives primitives g, h_1, \dots, h_m ;
- ▶ Rec(g, h) la fonction définie par récurrence comme

$$\begin{cases} f(0, x_2, \dots, x_n) = g(x_2, \dots, x_n), \\ f(x_1 + 1, x_2, \dots, x_n) = h(f(x_1, \dots, x_n), x_1, \dots, x_n), \end{cases}$$

où g et h sont récursives primitives.

- ▶ Min(g) la fonction qui à (x_2, \dots, x_n) associe le plus petit $y \in \mathbb{N}$ tel que $g(y, x_2, \dots, x_n) = 1$ s'il en existe (et qui n'est pas définie sinon).

43

Théorème

Une fonction $f : \mathbb{N}^n \rightarrow \mathbb{N}$ est *récursive* si et seulement si elle est *calculable par une machine de Turing*.

■ Thèse de Church :

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

44

Décidable = semi-décidable + co-semi-décidable

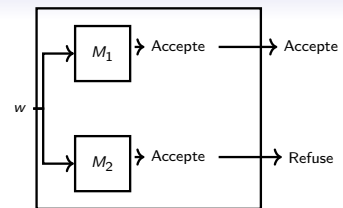
Théorème

Un langage est décidable si et seulement s'il est semi-décidable et son complémentaire aussi.

- Ce résultat justifie la terminologie de **semi-décidable**.

45

- Démonstration : Direction \Leftarrow .



- ▶ Supposons que L soit semi-décidable et son complémentaire aussi.
 - Il existe une machine de Turing M_1 qui termine en acceptant sur L ,
 - et une machine de Turing M_2 qui termine en acceptant sur son complémentaire.
- ▶ On construit une machine de Turing M qui, sur une entrée w , simule t étapes de M_1 et t étapes M_2 sur w , pour $t = 1, 2, \dots$ jusqu'à ce que l'une des deux termine :
 - si M_1 termine, la machine de Turing M accepte ;
 - si c'est M_2 , la machine M refuse.

46

- Démonstration : Direction \Rightarrow .

- ▶ Par définition, un langage décidable est semi-décidable.
- ▶ En inversant dans la machine de Turing l'état d'acceptation et de refus, son complémentaire est aussi décidable, et donc aussi semi-décidable.

47

Un problème non récursivement énumérable

- On considère alors le complémentaire du problème L_{univ} , que l'on va noter $\overline{L_{\text{univ}}}$:

Problème $\overline{L_{\text{univ}}}$:

Donnée: Le codage $\langle M \rangle$ d'une machine de Turing M et un mot w .
Réponse: Décider si la machine M n'accepte pas le mot w .

Corollaire

Le problème $\overline{L_{\text{univ}}}$ n'est pas semi-décidable.

- Preuve :

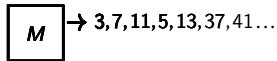
- ▶ Sinon, par le théorème précédent, le problème de décision L_{univ} serait décidable.

48

Énumération d'un langage RE

Théorème

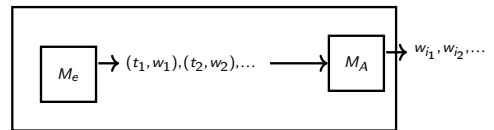
Un langage $L \subset M^*$ est récursivement énumérable (= semi-décidable) si et seulement si l'on peut produire une machine de Turing qui affiche un à un (énumère) tous les mots du langage L .



- Ce résultat justifie la terminologie de **récursivement énumérable** comme synonyme de semi-décidable.

49

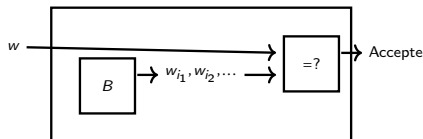
- Démonstration : Direction \Rightarrow .



- ▶ Supposons L récursivement énumérable : soit A la machine qui accepte les mots de L .
- ▶ $\mathbb{N} \times \Sigma^*$ est effectivement dénombrable : on peut construire une machine de Turing M_e qui produit le codage (t, w) de tous les couples (t, w) où t est un entier, w est un mot.
- ▶ Considérons une machine de Turing qui en plus, pour chaque couple produit (t, w) , simule t étapes de la machine A . Si la machine A termine et accepte en exactement t étapes, la machine affiche alors le mot w . Sinon elle n'affiche rien pour ce couple.

50

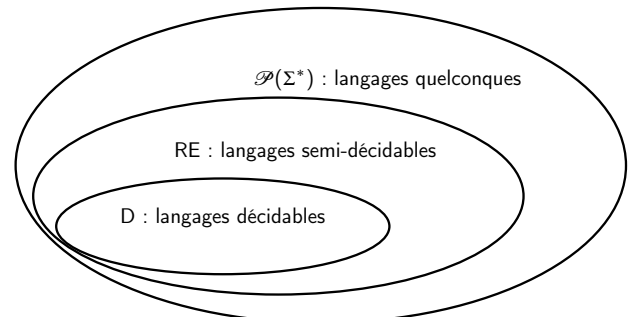
- Démonstration : Direction \Leftarrow .



- ▶ si l'on a une machine de Turing B qui énumère tous les mots du langage L , alors on peut construire une machine de Turing qui étant donné un mot w , simule B , et à chaque fois que B produit un mot compare ce mot au mot w .
 - S'ils sont égaux, alors la machine s'arrête et accepte.
 - Sinon, la machine continue à jamais.

51

Le paysage de la calculabilité



$$D \subsetneq RE \subsetneq \mathcal{P}(\Sigma^*),$$

$$L_{\text{univ}} \in RE \setminus D.$$

52

Constat dramatique

- L'objet de la vérification :
 - ▶ on se donne la description d'un système \mathcal{S}
 - ▶ on se donne la description d'une propriété ϕ
 - ▶ on souhaite déterminer si $\mathcal{S} \models \phi$, c'est-à-dire si le système vérifie sa spécification.
- Le problème est indécidable
 - ▶ dès que \mathcal{S} permet de modéliser des systèmes aussi simples que des systèmes à ≥ 2 compteurs ;
 - ▶ et que ϕ n'est pas une propriété toujours vraie ou toujours fausse.

53

Retour sur le transparent 12 du premier cours

Quelques histoires récentes :

- 370 millions de dollars :



- ≥ 475 millions de dollars :

$$\frac{4195835}{3145727} = 1.333739068902037589$$

- Garantir informatiquement qu'un système donné vérifie sa spécification n'est pas possible dans le cas général.
- On doit concevoir des méthodes qui évitent les difficultés.
- Ou qui peuvent être incomplètes. . .

54

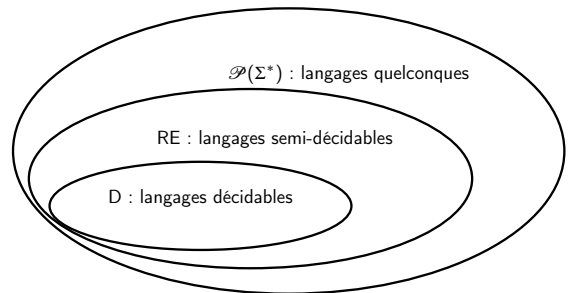
D'autres problèmes indécidables

- Indécidables :
 - ▶ Dixième problème de Hilbert :
 - Donnée :** Un polynôme $P \in \mathbb{Z}[X_1, \dots, X_n]$ à coefficients entiers.
 - Réponse :** Décider s'il possède une racine entière.
 - ▶ Simplification en calcul formel :
 - Donnée :** Une expression mathématique d'une variable x construite par composition à partir de la constante 1, l'addition, la soustraction, la multiplication, le sinus et la valeur absolue
 - Réponse :** Décider si cette expression est la fonction constante nulle.
 - ▶ En logique :
 - Donnée :** Une formule arithmétique F du premier ordre.
 - Réponse :** Décider si F est vraie sur les entiers (c-à-d $F \in Th(\mathbb{N})$).
- Remarque :
 - ▶ Théorème de Presburger : La théorie du premier ordre des entiers munis de l'addition seulement (mais pas de la multiplication) est décidable.
 - ▶ Autre remarque : Si on remplace sin par exp, dans l'énoncé, alors cela reste décidable.

55

Les cours à venir

- On souhaite parler d'une ressource/mesure élémentaire particulière : le **temps de calcul**.
- Objectif :
 - ▶ distinguer ce qui est raisonnable de ce qui n'est pas raisonnable en termes de temps de calcul.



56