

# Cours 2: Calculabilité

Olivier Bournez

bournez@lix.polytechnique.fr  
LIX, Ecole Polytechnique

# Au menu d'aujourd'hui

Quelques remarques sur les structures

Algorithme universel

Langages et problèmes (semi-)décidables

Indécidabilité

Problèmes indécidables naturels

Théorème du point fixe

## Sous menu

Quelques remarques sur les structures  
Structures finies vs non-finies  
A propos du codage

# Structures finies

## Théorème

*Un algorithme sur une structure*

$$\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$$

*avec  $M$  fini se simule par un algorithme sur la structure*

$$\mathfrak{B} = (\{0, 1\}, \mathbf{0}, \mathbf{1}, =).$$

- La notion de fonction “calculable” ou d’algorithme est donc la même sur toutes les structures finies.
  - ▶ On parlera d’algorithme *classique*.
- La notion de fonction “calculable” ou d’algorithme peut différer sur les structures non-finies.

# Structures infinies

- Sur  $\mathbb{N}$ , c'est-à-dire sur  $\mathfrak{M} = (\mathbb{N}, f_1, \dots, f_u, r_1, \dots, r_v)$  :
  - ▶ On peut coder un entier  $n$  par son écriture en binaire  $\langle n \rangle$ , et donc voir  $\mathbb{N}$  comme  $\{0, 1\}^*$
  - ▶ Si chacune des fonctions  $f_i$  et relations  $r_j$ , vue comme une fonction de  $\{0, 1\}^*$  dans  $\{0, 1\}^*$  est calculable,
    - alors la notion de fonction "calculable" sur  $\mathfrak{M}$  est la même que sur les structures finies.
    - un algorithme sur  $\mathbb{N}$  est un algorithme classique.

# Structures infinies

- Sur  $\mathbb{N}$ , c'est-à-dire sur  $\mathfrak{M} = (\mathbb{N}, f_1, \dots, f_u, r_1, \dots, r_v)$  :
  - ▶ On peut coder un entier  $n$  par son écriture en binaire  $\langle n \rangle$ , et donc voir  $\mathbb{N}$  comme  $\{0, 1\}^*$
  - ▶ Si chacune des fonctions  $f_i$  et relations  $r_j$ , vue comme une fonction de  $\{0, 1\}^*$  dans  $\{0, 1\}^*$  est calculable,
    - alors la notion de fonction "calculable" sur  $\mathfrak{M}$  est la même que sur les structures finies.
    - un algorithme sur  $\mathbb{N}$  est un algorithme classique.
- Sur  $\mathbb{R}$ , c'est-à-dire sur  $\mathfrak{M} = (\mathbb{R}, f_1, \dots, f_u, r_1, \dots, r_v)$  :
  - ▶ ce n'est pas si simple.
  - ▶ la notion de fonction calculable peut différer.

## Conventions sur les structures

- On veut au moins pouvoir faire ce que font les algorithmes sur les structures finies.

- ▶ Conventions :

- Au moins les symboles  $\mathbf{0}$ ,  $\mathbf{1}$ , *undef*,  $\mathbf{B}$  avec des interprétations  $0$ ,  $1$ , *undef*,  $B$  distinctes.
- Au moins l'égalité  $=$ .
- Au moins l'identité *Id*.
- Au moins les constantes *vrai* et *faux* d'interprétation  $1$  et  $0$ .
- Au moins un terme  $S(x, y, z)$  dont l'interprétation  $\llbracket S \rrbracket$  vérifie  $\llbracket S \rrbracket(1, y, z) = y$  et  $\llbracket S \rrbracket(0, y, z) = z$ .

- Exemples :

- Sur  $(\{0, 1\}, \mathbf{0}, \mathbf{1}, \vee, \wedge, =)$ ,  $S(x, y, z) = (x \wedge y) \vee (\neg x \wedge \neg z)$ .
- Sur  $(\mathbb{R}, +, -, *, =)$ ,  $S(x, y, z) = x * y + (1 - x) * z$ .

- ▶ Souvent certains de ces symboles seront implicites.
- ▶ Idée : on calcule au moins ce que l'on calcule par algorithme classique.

## Sous menu

Quelques remarques sur les structures

Structures finies vs non-finies

A propos du codage

# Coder des listes

- Si l'on sait coder les couples,
  - ▶ c'est-à-dire que l'on a une fonction qui à  $u$  et  $v$  associe  $\langle u, v \rangle$  qui code le couple  $(u, v)$ ,
  - ▶ et des fonctions pour retrouver  $u$  et  $v$  à partir de  $\langle u, v \rangle$

- alors, on sait coder/décoder

- ▶ les triplets :  $\langle u, v, w \rangle = \langle \langle u, v \rangle, w \rangle$
- ▶ les quadruplets :  $\langle u, v, w, x \rangle = \langle \langle \langle u, v \rangle, w \rangle, x \rangle$
- ▶ les  $n$ -uplets : ...
- ▶ les listes de longueur variable :

$$\langle x_1, x_2, \dots, x_n \rangle = \langle \langle \langle \langle x_1, x_2 \rangle \dots, x_n \rangle, \mathbf{B} \rangle$$

- ▶ les arbres
- ▶ les graphes
- ▶ ...

## Coder des couples ?

- Par concaténation.

$$\langle 010, 11 \rangle = 010\#11$$

- Par des tableaux :

- ▶  $tab(1) = 0, tab(2) = 1, tab(3) = 0, tab(4) = B, tab'(1) = 1, tab'(2) = 1, tab'(3) = B$
- ▶  $\langle tab, tab' \rangle$  est un tableau  $tab''$  avec  $tab''(1) = 0, tab''(2) = 1, tab''(3) = 0, tab''(4) = B, tab''(5) = 1, tab''(6) = 1, tab''(7) = B$

- Par une fonction particulière :

- ▶  $\mathbb{N}^2$  est dénombrable, donc il existe une fonction injective  $\langle \cdot, \cdot \rangle$  de  $\mathbb{N}^2$  dans  $\mathbb{N}$ , comme  $(p, q) \mapsto (p^2 + q^2 + 2pq + p + 3q)/2$ .

... etc ...

- On dira qu'une structure est *suffisamment expressive* si elle permet de représenter les couples, et donc les listes.

# Au menu d'aujourd'hui

Quelques remarques sur les structures

**Algorithme universel**

Langages et problèmes (semi-)décidables

Indécidabilité

Problèmes indécidables naturels

Théorème du point fixe

# Sous menu

Algorithme universel

Un algorithme est un arbre

Algorithme universel

Algorithme universel sur une structure  $\mathfrak{M}$

# Un algorithme est un arbre

Un algorithme sur la signature  $\Sigma$  peut se voir comme un arbre étiqueté par les symboles de fonctions et de relations de  $\Sigma$ , union  $\{\leftarrow, \text{if then, par}\}$ .

Car :

- Un terme est un arbre étiqueté par les symboles de  $\Sigma$
- Chaque autre nœud est :
  - ▶ soit étiqueté par le symbole  $\leftarrow$  : le nœud possède alors deux fils correspondant à deux termes.
  - ▶ soit étiqueté par le symbole **par** : le nœud possède alors 2 fils correspondant à deux instructions.
  - ▶ soit étiqueté par le symbole **if then** : le nœud possède alors deux fils, l'un correspondant à un terme  $t$ , l'autre à une instruction.

# Sous menu

## Algorithme universel

Un algorithme est un arbre

## Algorithme universel

Algorithme universel sur une structure  $\mathfrak{M}$

# Algorithme universel

Théorème (Existence d'un algorithme/d'une machine universelle)

*Il existe un algorithme qui prend en entrée  $\langle A, d \rangle$ , où*

- 1.  $A$  est un algorithme*
- 2.  $d$  est une donnée*

*qui simule l'algorithme  $A$  sur l'entrée  $d$ .*

# Sous menu

## Algorithme universel

Un algorithme est un arbre

Algorithme universel

Algorithme universel sur une structure  $\mathfrak{M}$

# Algorithme/Machine universelle

## Théorème (Existence d'un algorithme/d'une machine universelle)

*Fixons une structure  $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$  suffisamment expressive.*

*Il existe un algorithme sur  $\mathfrak{M}$  qui prend en entrée  $\langle A, d \rangle$ , où*

- 1.  $A$  est un algorithme sur  $\mathfrak{M}$*
- 2.  $d$  est une donnée*

*qui simule l'algorithme  $A$  sur l'entrée  $d$ .*

# Au menu d'aujourd'hui

Quelques remarques sur les structures

Algorithme universel

Langages et problèmes (semi-)décidables

Indécidabilité

Problèmes indécidables naturels

Théorème du point fixe

### Langages et problèmes (semi-)décidables

- Problème de décision

- Langage semi-décidable

- Langage décidable

- Semi-décidable vs décidable

- Semi-décision et énumération

- Propriétés de clôture

# Problème de décision

- Un *problème de décision*  $P$  est la donnée
  1. d'un ensemble  $E$  d'instances,
  2. et d'un sous-ensemble  $E^+$  d'instances *positives*.

Exemple :

1. *Nombre premiers* :  $E = \mathbb{N}$ ,  $E^+$  est le sous-ensemble des entiers premiers.
2. *Algorithme* :  $E = M^*$ , et  $E^+ = \{ \langle A \rangle \mid A \text{ algorithme} \}$ .
3. *Chemin* :  $E = \langle G, u, v \rangle$  et  $E^+ = \{ \langle G, u, v \rangle \mid \text{chemin de } u \text{ vers } v \text{ dans } G \}$ .

# Problème de décision

- Un *problème de décision*  $P$  est la donnée
  1. d'un ensemble  $E$  d'instances,
  2. et d'un sous-ensemble  $E^+$  d'instances *positives*.

Implicitement, on a fixé un codage.

- Parler de problèmes de décision ou de langages est équivalent :
  - ▶ On peut voir un problème comme un langage : à un problème  $P$ , on associe  $L(P) = \{w | w \in E^+\}$ .
  - ▶ On peut voir un langage  $L$  comme un problème :  $E = M^*$ ,  $E^+ = L$ .

### Langages et problèmes (semi-)décidables

Problème de décision

Langage semi-décidable

Langage décidable

Semi-décidable vs décidable

Semi-décision et énumération

Propriétés de clôture

## Langage semi-décidable

- Un langage  $L \subset M^*$  est dit *semi-décidable* (*récurivement énumérable*) s'il existe un algorithme (classique) qui sur toute entrée  $w \in M^*$  termine si et seulement si  $w \in L$ .
- Un langage  $L \subset M^*$  est dit *semi-décidable sur la structure*

$$\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$$

s'il existe un algorithme sur la structure  $\mathfrak{M}$  qui sur toute entrée  $w \in M^*$  termine si et seulement si  $w \in L$ .

On note RE la classe des langages et des problèmes semi-décidables.

On note  $L(A)$  pour les données sur lequel  $A$  termine.

### Langages et problèmes (semi-)décidables

Problème de décision

Langage semi-décidable

**Langage décidable**

Semi-décidable vs décidable

Semi-décision et énumération

Propriétés de clôture

# Langage décidable

- Un langage  $L \subset M^*$  est dit *décidable (récuratif)* s'il est semi-décidable et son complémentaire aussi.
- Un langage  $L \subset M^*$  est dit *décidable sur la structure*

$$\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$$

s'il est semi-décidable sur cette structure et son complémentaire aussi.

On note  $\mathbb{R}$  pour la classe des langages et des problèmes *décidables*.

- Corollaire :  $R \subset RE$

### Langages et problèmes (semi-)décidables

Problème de décision

Langage semi-décidable

Langage décidable

**Semi-décidable vs décidable**

Semi-décision et énumération

Propriétés de clôture

## Théorème

*Un problème ou un langage  $L$  est semi-décidable (respectivement sur une structure  $\mathfrak{M}$ ) si et seulement s'il existe un algorithme (resp. sur  $\mathfrak{M}$ ) qui*

- *termine avec la réponse 1 lorsque  $w \in L$*
- *soit termine avec la réponse 0, soit ne termine pas, lorsque  $w \notin L$ .*

## Théorème

*Un problème ou un langage  $L$  est décidable (respectivement sur une structure  $\mathfrak{M}$ ) si et seulement s'il existe un algorithme (resp. sur  $\mathfrak{M}$ ) qui*

- *termine sur toute entrée  $w$  ;*
- *termine avec la réponse 1 (vrai) lorsque  $w \in L$  ;*
- *termine avec la réponse 0 (faux) lorsque  $w \notin L$ .*

## Preuve

sens  $\Rightarrow$ .

- Supposons que  $L$  soit récursif.
- Il existe un algorithme  $M_1$  qui termine sur  $L$ , et un algorithme  $M_2$  qui termine sur son complémentaire.
- On construit un algorithme qui, sur une entrée  $w$ , simule en parallèle  $M_1$  et  $M_2$ , jusqu'à ce que l'un des deux termine. Si  $M_1$  termine, il répond 1. Si c'est  $M_2$  il répond 0.

sens  $\Leftarrow$ .

- Réciproquement, supposons qu'il existe un tel algorithme  $A$ .
- On construit un algorithme  $B$  qui termine sur  $L$  en construisant un algorithme qui simule  $A$ . Si  $A$  répond 1, l'algorithme  $B$  s'arrête alors, et si  $A$  répond 0, l'algorithme  $B$  rentre dans une boucle qui ne termine jamais.
- On construit un algorithme  $B'$  qui termine sur le complémentaire de  $L$  en inversant le rôle de 1 et 0 plus haut.

### Langages et problèmes (semi-)décidables

Problème de décision

Langage semi-décidable

Langage décidable

Semi-décidable vs décidable

**Semi-décision et énumération**

Propriétés de clôture

Supposons  $M$  fini.

### Théorème

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

■ sens  $\Rightarrow$ .

- ▶ Supposons que  $A$  termine sur les mots de  $L$ .
- ▶ L'ensemble de couples  $(t, w)$ , où  $t$  est un entier,  $w$  est un mot est effectivement dénombrable.
- ▶ Considérons un algorithme  $B$  qui pour chaque couple produit  $(t, w)$ , simule  $t$  étapes de la machine  $A$ . Si la machine  $A$  termine en exactement  $t$  étapes,  $B$  affiche alors le mot  $w$ .
- ▶ Un mot du langage  $L$ , est accepté par  $A$  en un certain temps  $t$  sera écrit par  $B$ .

■ sens  $\Leftarrow$ .

- ▶ Supposons que l'on a un algorithme classique  $B$  qui énumère tous les mots du langage  $L$ .
- ▶ On construit algorithme  $A$ , 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,  $A$  s'arrête.

### Langages et problèmes (semi-)décidables

Problème de décision

Langage semi-décidable

Langage décidable

Semi-décidable vs décidable

Semi-décision et énumération

Propriétés de clôture

## Théorème

*L'ensemble des langages semi-décidables (respectivement sur une structure  $\mathfrak{M}$ ) est clos par union et par intersection : autrement dit, si  $L_1$  et  $L_2$  sont semi-décidables, alors  $L_1 \cup L_2$  et  $L_1 \cap L_2$  le sont.*

## Théorème

*L'ensemble des langages décidables (respectivement sur une structure  $\mathfrak{M}$ ) est clos par union, intersection, et complément : autrement dit, si  $L_1$  et  $L_2$  sont décidables, alors  $L_1 \cup L_2$ ,  $L_1 \cap L_2$ , et  $L_1^c$  le sont.*

- Supposons que  $L_1$  corresponde à  $L(A_1)$  pour un algorithme  $A_1$  et  $L_2$  à  $L(A_2)$  pour un algorithme  $A_2$ .
- Alors  $L_1 \cup L_2$  (respectivement :  $L_1 \cap L_2$ ) correspond à l'algorithme  $A$  qui simule en parallèle  $A_1$  et  $A_2$  et qui termine dès que le premier des deux algorithmes (resp. les deux algorithmes)  $A_1$  et  $A_2$  termine.
  
- Le deuxième théorème découle du premier par les lois de Morgan.

# Au menu d'aujourd'hui

Quelques remarques sur les structures

Algorithme universel

Langages et problèmes (semi-)décidables

**Indécidabilité**

Problèmes indécidables naturels

Théorème du point fixe

## Indécidabilité

**Un premier problème indécidable**

Notion de réduction

Quelques autres problèmes indécidables

Théorème de Rice

Le drame de la vérification

Notion de complétude

# Le langage universel

- On appelle langage universel noté  $L_{univ}$ , le langage

$$L_{univ} = \{ \langle A, w \rangle \mid A \text{ est un algorithme qui termine sur } w \}.$$

- On appelle langage universel sur  $\mathfrak{M}$  noté  $L_{univ}$ , le langage

$$L_{univ} = \{ \langle A, w \rangle \mid A \text{ est un algorithme sur } \mathfrak{M} \text{ qui termine sur } w \}.$$

$$L_{univ} \in RE.$$

## Théorème

*Le langage universel  $L_{univ}$  (respectivement sur une structure  $\mathfrak{M}$ ) est semi-décidable (resp. sur la structure  $\mathfrak{M}$ ).*

$$L_{univ} \notin R.$$

## Théorème

*Le langage universel  $L_{univ}$  (respectivement sur une structure  $\mathfrak{M}$ ) n'est pas décidable (resp. sur la structure  $\mathfrak{M}$ ).*

$$L_{univ} \in RE - R.$$

- On prouve le résultat par un raisonnement par contradiction.
- Supposons que  $L_{univ}$  soit décidé par un algorithme  $A$ , qui s'arrête sur toute entrée.
- On construit alors un algorithme  $B$  de la façon suivante :
  - ▶  $B$  prend en entrée un mot  $\langle C \rangle$  codant un algorithme  $C$
  - ▶  $B$  appelle l'algorithme  $A$  sur le mot  $\langle\langle C \rangle, \langle C \rangle\rangle$
  - ▶ Si l'algorithme  $A$ 
    - accepte ce mot,  $B$  refuse.
    - refuse ce mot,  $B$  accepte.
- Maintenant :
  - ▶ Si  $B$  accepte  $\langle B \rangle$ , cela signifie, par définition de  $L_{univ}$  et de  $A$ , que  $A$  accepte  $\langle\langle B \rangle, \langle B \rangle\rangle$ . Mais si  $A$  accepte ce mot,  $B$  est construit pour refuser son entrée  $\langle B \rangle$ . Contradiction.
  - ▶ Si  $B$  refuse  $\langle B \rangle$ , cela signifie, par définition de  $L_{univ}$  et de  $A$ , que  $A$  refuse  $\langle\langle B \rangle, \langle B \rangle\rangle$ . Mais si  $A$  refuse ce mot,  $B$  est construit pour accepter son entrée  $\langle B \rangle$ . Contradiction.

- Autrement dit,  $R \subset RE$ , et l'inclusion est stricte.
  
- Le complémentaire du langage  $L_{univ}$  (respectivement sur une structure  $\mathfrak{M}$ ) n'est pas décidable (resp. sur la structure  $\mathfrak{M}$ ).

## Indécidabilité

Un premier problème indécidable

### Notion de réduction

Quelques autres problèmes indécidables

Théorème de Rice

Le drame de la vérification

Notion de complétude

# Fonction calculable

- Une fonction  $f : M^* \rightarrow N^*$  est *calculable* s'il existe un algorithme  $A$ , tel que pour tout mot  $w$ ,  $A$  avec l'entrée  $w$  termine, et écrit en sortie  $f(w)$ .
- La composée de deux fonctions calculables est calculable.
- Il existe des fonctions non-calculables.

# Réduction

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

$$w \in L_A \text{ ssi } f(w) \in L_B.$$

- On note  $A \leq_m B$  lorsque  $A$  se réduit à  $B$ .

Attention au sens.

# Comparer les problèmes

## Proposition (Réduction)

*Si  $A \leq_m B$ , et si  $B$  est décidable (respectivement : sur une structure  $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$ ) alors  $A$  est décidable (resp. sur  $\mathfrak{M}$ ).*

## Proposition (Réduction)

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

- $A$  est décidé par l'algorithme qui, sur une entrée  $w$ , calcule  $f(w)$ , puis simule l'algorithme qui décide  $B$  sur l'entrée  $f(w)$ .
  - Puisqu'on a  $w \in A$  si et seulement si  $f(w) \in B$ , l'algorithme est correct.
- 
- La deuxième proposition est la contraposée de la première.

# Sous menu

## Indécidabilité

Un premier problème indécidable

Notion de réduction

**Quelques autres problèmes indécidables**

Théorème de Rice

Le drame de la vérification

Notion de complétude

# Le vide est indécidable

- Il n'est pas possible de déterminer si un algorithme s'arrête sur au moins une entrée :

## Proposition

$$L_{\emptyset} = \{ \langle A \rangle \mid L(A) \neq \emptyset \}$$

*est indécidable.*

- On construit une réduction de  $L_{univ}$  vers  $L_\emptyset$  :
- pour toute paire  $\langle \langle A \rangle, w \rangle$ , on considère l'algorithme  $A_w$  défini de la manière suivante :
  - ▶  $A_w$  prend en entrée un mot  $u$ .
  - ▶ Si  $u = w$ ,  $A_w$  simule  $A$  sur  $w$ ,
  - ▶ sinon  $A$  rejette.
- La fonction  $f$  qui à  $\langle \langle A \rangle, w \rangle$  associe  $\langle A_w \rangle$  est bien calculable.
- De plus on a  $\langle \langle A \rangle, w \rangle \in L_{univ}$  si et seulement si  $L(A_w) \neq \emptyset$ , c'est-à-dire  $\langle A_w \rangle \in L_\emptyset$  :
  - ▶ en effet, le mot  $w$  est le seul qui puisse être accepté par  $A_w$ , et il l'est si et seulement si  $A$  accepte  $w$ .

# L'égalité entre langages est indécidable

- Il n'est pas possible de déterminer si deux algorithmes acceptent le même langage.

## Proposition

$$L_{\neq} = \{ \langle A, A' \rangle \mid L(A) \neq L(A') \}$$

*est indécidable.*

- On construit une réduction de  $L_\emptyset$  vers  $L_\neq$ .
- On considère un algorithme fixe  $B$  qui accepte le langage vide :
  - ▶ prendre par exemple un algorithme  $B$  qui rentre immédiatement dans une boucle sans fin.
- La fonction  $f$  qui à  $\langle A \rangle$  associe  $\langle A, B \rangle$  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  $\langle A, B \rangle \in L_\neq$ .

# Sous menu

## Indécidabilité

Un premier problème indécidable

Notion de réduction

Quelques autres problèmes indécidables

**Théorème de Rice**

Le drame de la vérification

Notion de complétude

# Théorème de Rice

## Théorème

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

*Autrement dit,*

- *soit une propriété  $P$  non-triviale, c'est-à-dire*
  1. *qui n'est pas vraie pour tout langage semi-décidable,*
  2. *et qui n'est pas non plus fausse non plus pour tout langage semi-décidable.*
- *Alors  $L_P = \{ \langle A \rangle \mid L(A) \text{ vérifie } P \}$  est indécidable.*

- Il nous faut démontrer que  $L_P$  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 un algorithme  $B$  avec  $L(B)$  qui vérifie  $P$ .
- On construit une réduction de  $L_{univ}$  vers le langage  $L_P$ .
- Étant donnée une paire  $\langle \langle A \rangle, w \rangle$ , on considère l'algorithme  $A_w$  défini 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 et seulement si  $B$  accepte  $u$ .
- Autrement dit,  $A_w$  accepte, si et seulement si  $A$  accepte  $w$  et si  $B$  accepte  $u$ .
  1. Si  $w$  est accepté par  $A$ , alors  $L(A_w)$  vaut  $L(B)$ , et donc vérifie la propriété  $P$ .
  2. 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  $\langle \langle A \rangle, w \rangle$  associe  $\langle A_w \rangle$  est bien calculable.

## Indécidabilité

Un premier problème indécidable

Notion de réduction

Quelques autres problèmes indécidables

Théorème de Rice

**Le drame de la vérification**

Notion de complétude

# 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  $\phi$
  - ▶ on souhaite déterminer si  $\mathcal{S} \models \phi$ .
- Le problème est indécidable
  - ▶ dès que  $\mathcal{S}$  permet de modéliser des systèmes aussi simples que des systèmes à  $\geq 2$  compteurs.
  - ▶ et que  $\phi$  n'est pas une propriété toujours vraie ou toujours fausse.

# Sous menu

## Indécidabilité

Un premier problème indécidable

Notion de réduction

Quelques autres problèmes indécidables

Théorème de Rice

Le drame de la vérification

**Notion de complétude**

# RE-complétude

- Un problème  $A$  est dit RE-complet, si
  1. il est récursivement énumérable
  2. tout autre problème récursivement énumérable  $B$  est tel que  $B \leq_m A$ .
  
- Autrement dit, un problème RE-complet est maximal pour  $\leq_m$  parmi les problèmes de la classe RE.

## Théorème

*Le problème  $L_{univ}$  est RE-complet.*

- $L_{univ}$  est semi-décidable.
- Maintenant, soit  $L$  un langage semi-décidable, et  $A$  l'algorithme correspondant.
- Considérons la fonction  $f$  qui à  $w$  associe le mot  $\langle\langle A \rangle, w \rangle$  :
  - ▶ on a  $w \in L$  si et seulement si  $f(w) \in L_{univ}$ .

# Au menu d'aujourd'hui

Quelques remarques sur les structures

Algorithme universel

Langages et problèmes (semi-)décidables

Indécidabilité

**Problèmes indécidables naturels**

Théorème du point fixe

# Sous menu

Problèmes indécidables naturels

Le dixième problème de Hilbert

Le problème de la correspondance de Post

Décidabilité/Indécidabilité de théories logiques

# Le dixième problème de Hilbert

## Théorème

*Étant donné un polynôme  $P \in \mathbb{N}[X_1, \dots, X_n]$  à coefficients entiers, il n'est pas possible de décider s'il possède une solution entière.*

## Sous menu

### Problèmes indécidables naturels

Le dixième problème de Hilbert

**Le problème de la correspondance de Post**

Décidabilité/Indécidabilité de théories logiques

# Le problème de la correspondance de Post

- Le *problème de la correspondance de Post* est le suivant :
  - ▶ Une instance est une suite  $(u_1, v_1), \dots, (u_n, v_n)$  de paires de mots sur l'alphabet  $\Sigma$ .
  - ▶ Une solution est une suite d'indices  $i_1, i_2, \dots, i_m$  de  $\{1, 2, \dots, n\}$  telle que

$$u_{i_1} u_{i_2} \cdots u_{i_m} = v_{i_1} v_{i_2} \cdots v_{i_m}$$

## Théorème

*Étant donnée une instance, il n'est pas possible de déterminer si elle possède une solution.*

# Sous menu

## Problèmes indécidables naturels

Le dixième problème de Hilbert

Le problème de la correspondance de Post

Décidabilité/Indécidabilité de théories logiques

# Décidabilité/Indécidabilité de théories logiques

- Une théorie logique est dite *décidable* si le problème de savoir si une formule close est vraie est décidable.

## Théorème (Presburger)

*La théorie du premier ordre des entiers muni de l'addition (mais pas de la multiplication) est décidable.*

## Théorème (Tarski)

*La théorie du premier ordre des entiers munis de l'addition et de la multiplication est indécidable.*

# Au menu d'aujourd'hui

Quelques remarques sur les structures

Algorithme universel

Langages et problèmes (semi-)décidables

Indécidabilité

Problèmes indécidables naturels

**Théorème du point fixe**

# Sous menu

Théorème du point fixe

Quines

Théorème de récursion

Théorème du point fixe de Kleene

## Proposition

*Il existe un algorithme  $A^*$  qui écrit son propre algorithme : il produit en sortie  $\langle A^* \rangle$ .*

En shell *UNIX* par exemple, le programme suivant

```
x='y='echo . | tr . "\47" ' ; echo "x=$y$x$y;$x" '  
y='echo . | tr . "\47" ' ; echo "x=$y$x$y;$x"
```

produit

```
x='y='echo . | tr . "\47" ' ; echo "x=$y$x$y;$x" ' ; y='echo .  
| tr . "\47" ' ; echo "x=$y$x$y;$x"
```

qui est une commande, qui exécutée, affiche son propre code.

- On considère des algorithmes qui terminent sur toute entrée.
- Pour deux tels algorithmes  $A$  et  $A'$ , on note  $AA'$  l'algorithme qui est obtenu en composant de façon séquentielle  $A$  et  $A'$ .
- On construit les algorithmes suivants :
  1. Étant donné un mot  $w$ , l'algorithme  $Print_w$  termine avec le résultat  $w$ .
  2. Pour une entrée  $w$  de la forme  $w = \langle X \rangle$ , où  $X$  est un algorithme, l'algorithme  $B$  produit en sortie le codage de l'algorithme  $Print_w X$ .
- On considère alors l'algorithme  $A^*$  donné par  $Print_{\langle B \rangle} B$ .
- Déroulons le résultat de cet algorithme : l'algorithme  $Print_{\langle B \rangle}$  produit en sortie  $\langle B \rangle$ . La composition par  $B$  produit alors le codage de  $Print_{\langle B \rangle} \langle B \rangle$ , qui est bien le codage de l'algorithme  $A^*$ .

# Sous menu

**Théorème du point fixe**

Quines

**Théorème de récursion**

Théorème du point fixe de Kleene

## Théorème (Théorème de récursion)

*Soit  $t : M^* \times M^* \rightarrow M^*$  une fonction calculable. Alors il existe un algorithme  $R$  qui calcule une fonction  $r : M^* \rightarrow M^*$  telle que pour tout mot  $w$*

$$r(w) = t(\langle R \rangle, w).$$

- Soit  $T$  un algorithme calculant la fonction  $t : T$  prend en entrée une paire  $\langle u, w \rangle$  et produit en sortie un mot  $t(u, w)$ .
- On considère les algorithmes suivants :
  1. Étant donné un mot  $w$ , l'algorithme  $Print_w$  prend en entrée un mot  $u$  et termine avec le résultat  $\langle w, u \rangle$ .
  2. Pour une entrée  $w'$  de la forme  $\langle \langle X \rangle, w \rangle$ , l'algorithme  $B$ 
    - 2.1 calcule  $\langle \langle Print_{\langle X \rangle} X \rangle, w \rangle$ , où  $Print_{\langle X \rangle} X$  désigne l'algorithme qui compose  $Print_{\langle X \rangle}$  avec  $X$ ,
    - 2.2 puis passe le contrôle à l'algorithme  $T$ .
- On considère alors l'algorithme  $R$  donné par  $Print_{\langle B \rangle} B$ .
- Déroulons le résultat  $r(w)$  de cet algorithme  $R$  sur une entrée  $w$  :
  - ▶ l'algorithme  $Print_{\langle B \rangle}$  produit en sortie  $\langle \langle B \rangle, w \rangle$ .
  - ▶ La composition par  $B$  produit alors le codage de  $\langle \langle Print_{\langle B \rangle} B \rangle, w \rangle$ , et passe le contrôle à  $T$ .
  - ▶ Ce dernier produit alors  $t(\langle Print_{\langle B \rangle} B \rangle, w) = t(\langle R \rangle, w) = r(w)$ .

# Sous menu

Théorème du point fixe

Quines

Théorème de récursion

Théorème du point fixe de Kleene

# Théorème du point fixe de Kleene

## Théorème

*Soit une fonction calculable qui à chaque mot  $\langle A \rangle$  codant un algorithme associe un mot  $\langle A' \rangle$  codant un algorithme. Notons  $A' = f(A)$ .*

*Alors il existe un algorithme  $A^*$  tel que  $L(A^*) = L(f(A^*))$ .*

- Considérons une fonction  $t : M^* \times M^* \rightarrow M^*$  telle que  $t(\langle A \rangle, x)$  soit le résultat de la simulation de l'algorithme  $f(A)$  sur l'entrée  $x$ .
- Par le théorème précédent, il existe un algorithme  $R$  qui calcule une fonction  $r$  telle que  $r(w) = t(\langle R \rangle, w)$ .
- Par construction  $A^* = R$  et  $f(A^*) = f(R)$  ont donc même valeur sur  $w$  pour tout  $w$ .

- On peut interpréter les résultats précédents en lien avec les virus informatiques :
  - ▶ Ces principes donnent des moyens de s'autoreproduire, en dupliquant son code.