Logique et calcul

D. Bonnay (Paris X & IHPST-DEC)

Cogmaster 2008-2009

Le programme

- La logique comme théorie du raisonnement correct, qui répond à la question :
  “What follows from what ?”
- La logique comme théorie du calcul, qui répond à la question :
  “Qu’est-ce qui peut calculer quoi ?”

Logique et calcul

Deux branches de la logique formelle :

- la théorie de la **calculabilité**
  - Qu'est-ce qu'une procédure de calcul / un algorithme / une méthode effective ?
  - Quels sont les problèmes qui admettent une solution selon une méthode effective ?

Logique et calcul

Deux branches de la logique formelle :

- la théorie de la **calculabilité**
  - Qu'est-ce qu'une procédure de calcul / un algorithme / une méthode effective ?
  - Quels sont les problèmes qui admettent une solution selon une méthode effective ?
- la théorie de la **complexité**.
  - Qu'est-ce qu'une procédure de calcul *efficace* ?
  - Comment peut-on mesurer la difficulté d'un problème ?

Remarque

On va s'intéresser essentiellement à des calculs sur les entiers naturels.

Mais notre notion de calcul s'applique, moyennant un codage adéquat, à la résolution de tout problème qui peut être encodé comme un problème à propos des entiers.

Remarque

On va s'intéresser essentiellement à des calculs sur les entiers naturels.

Mais notre notion de calcul s'applique, moyennant un codage adéquat, à la résolution de tout problème qui peut être encodé comme un problème à propos des entiers.

Exemple

- Problème : décider si une suite d'expressions constitue une phrase dans une grammaire donnée.

Remarque

On va s'intéresser essentiellement à des calculs sur les entiers naturels.

Mais notre notion de calcul s'applique, moyennant un codage adéquat, à la résolution de tout problème qui peut être encodé comme un problème à propos des entiers.

Exemple

- Problème : décider si une suite d'expressions constitue une phrase dans une grammaire donnée.
- Codage des suites d'expressions dans les entiers, grammaire comme fonction qui envoie les suites d'expressions dans $\{0, 1\}$ (correct / pas correct)

Remarque

On va s'intéresser essentiellement à des calculs sur les entiers naturels.

Mais notre notion de calcul s'applique, moyennant un codage adéquat, à la résolution de tout problème qui peut être encodé comme un problème à propos des entiers.

Exemple

- Problème : décider si une suite d'expressions constitue une phrase dans une grammaire donnée.
- Codage des suites d'expressions dans les entiers, grammaire comme fonction qui envoie les suites d'expressions dans $\{0, 1\}$ (correct / pas correct)
- Le problème initial se réduit à un problème à propos des entiers (calcul de la 'fonction' grammaire)

Modèles de calcul

De nombreux modèles différents :

- Machines de Turing,
  Notion de machine abstraite proposée par Alan Turing,

Modèles de calcul

De nombreux modèles différents :

- Machines de Turing,
  Notion de machine abstraite proposée par Alan Turing,
- Un langage de programmation élémentaire,
  par exemple langage impératif,

Modèles de calcul

De nombreux modèles différents :

- Machines de Turing,
  Notion de machine abstraite proposée par Alan Turing,
- Un langage de programmation élémentaire,
  par exemple langage impératif,
- Fonctions récursives,
  $\approx$ fonctions élémentaires sur les entiers + récursion

Modèles de calcul

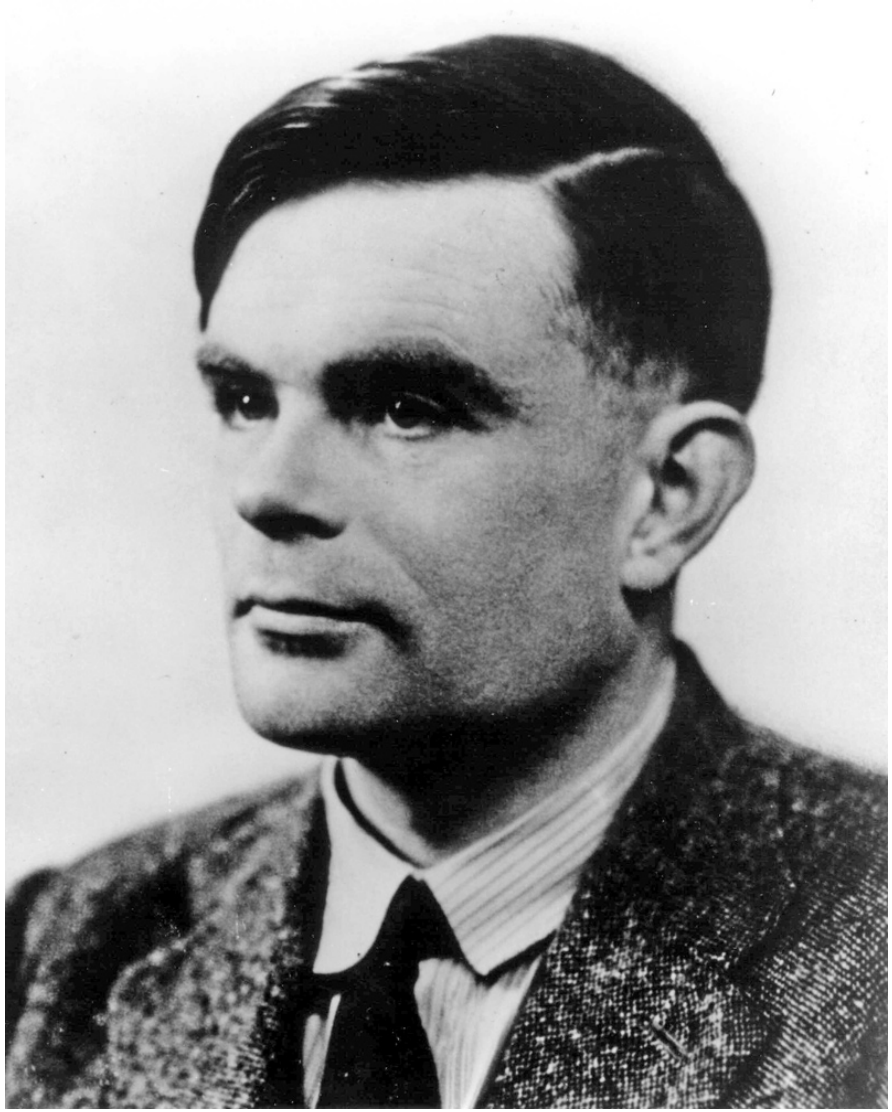
De nombreux modèles différents :

- Machines de Turing,
  Notion de machine abstraite proposée par Alan Turing,
- Un langage de programmation élémentaire,
  par exemple langage impératif,
- Fonctions récursives,
  $\approx$ fonctions élémentaires sur les entiers + récursion
- Lambda-calcul
  $\approx$ langage de programmation fonctionnelle (Lisp)

Modèles de calcul

De nombreux modèles différents :

- Machines de Turing,
  Notion de machine abstraite proposée par Alan Turing,
- Un langage de programmation élémentaire,
  par exemple langage impératif.
- Fonctions récursives,
  $\approx$ fonctions élémentaires sur les entiers + récursion
- Lambda-calcul
  $\approx$ langage de programmation fonctionnelle (Lisp)



Alan Turing (1912–1954)

Machines de Turing (1)

Machine abstraite : modèle mathématique des machines réelles

Machines de Turing (1)

Machine abstraite : modèle mathématique des machines réelles

Trois ingrédients :

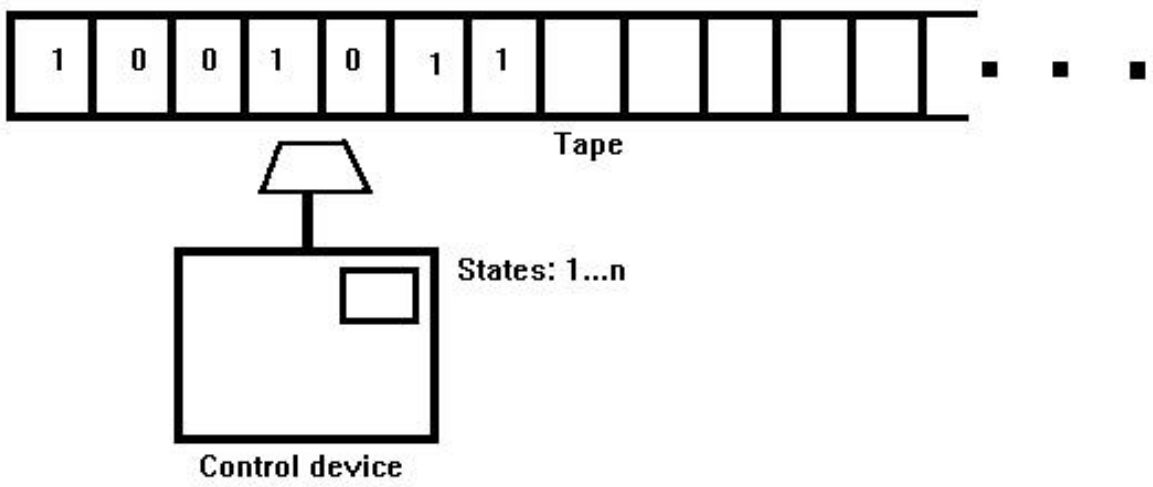
- Un ruban, avec des cases remplies de 0 ou de 1.
- Une tête de lecture / écriture
- Des instructions qui déterminent ce que va faire la machine.

Machines de Turing (1)

Machine abstraite : modèle mathématique des machines réelles

Trois ingrédients :

- Un ruban, avec des cases remplies de 0 ou de 1.
- Une tête de lecture / écriture
- Des instructions qui déterminent ce que va faire la machine.



Machine de Turing (2)

Une machine est définie par :

- un ensemble fini d'états $q_1, ..., q_n$, auxquels sont associés des instructions de la
- pour chaque état q_i, des instructions de la forme :
  si la case du ruban est marquée de 0 (resp. 1), accomplir l'action X et passer à l'état q_j.

Machine de Turing (2)

Une machine est définie par :

- un ensemble fini d'états $q_1, ..., q_n$, auxquels sont associés des instructions de la
- pour chaque état q_i, des instructions de la forme :
  si la case du ruban est marquée de 0 (resp. 1), accomplir l'action X et passer à l'état q_j.

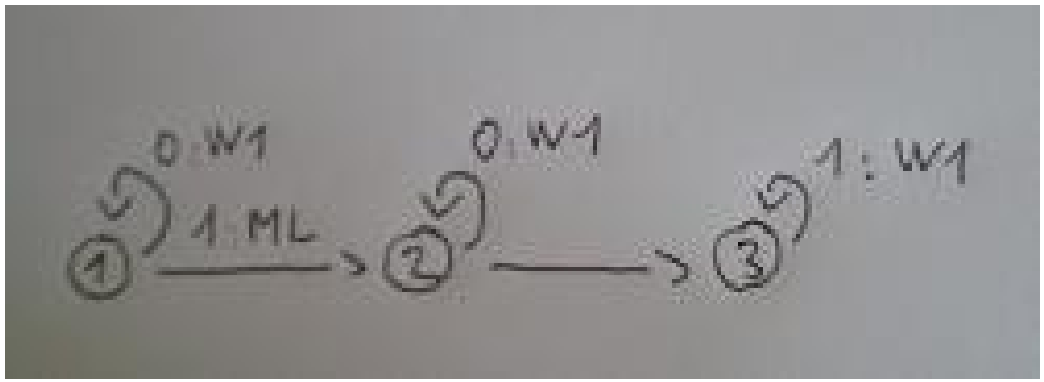
Les actions X possibles sont :

- écrire 0 [W0],
- écrire 1 [W2],
- bouger la tête d'une case vers la gauche [ML],
- bouger la tête d'une case vers la droite [MR],
- s'arrêter [pas d'instruction].

Exemple de machine (1)

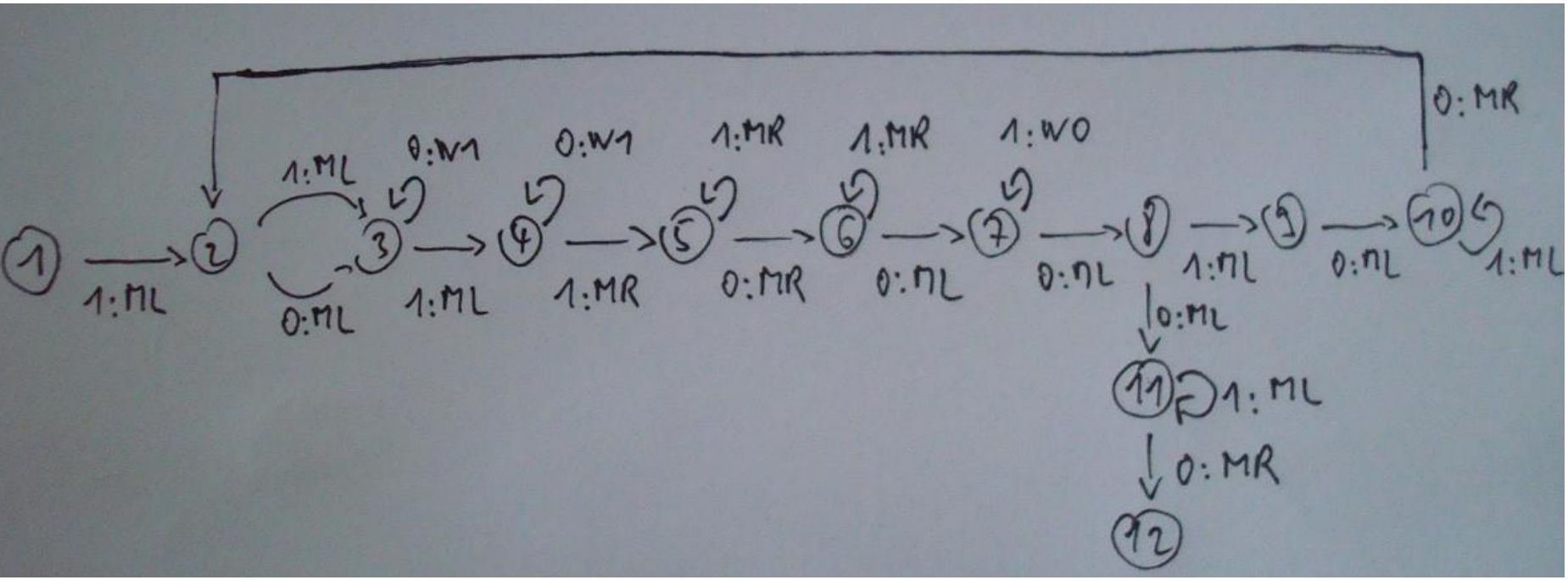
La machine à écrire trois "1" :

q_1	Si 0, exécuter W1 et passser à l'état q_1 Si 1, exécuter ML et à l'état q_2
q_2	Si 0, exécuter W1 et passser à l'état q_2 Si 1, exécuter ML et passer à l'état q_3
q_3	Si 0, exécuter W1 et passer à l'état q_3



Exemple de machine (2)

La machine à multiplier par deux :



La machine de Turing universelle

Chaque machine de Turing peut être *codée* et donnée comme entrée à une machine de Turing.

Une machine de Turing peut en *simuler* une autre, si elle se comporte comme celle-ci lorsqu'on lui donne en entrée le code de celle-ci.

Une machine de Turing serait une machine de Turing qui peut simuler toutes les machines de Turing.

Fait

Il existe une machine de Turing universelle.

Limites de la calculabilité

Fait

Il existe des fonctions qui ne sont pas calculables par machines de Turing.

Limites de la calculabilité

Fait

Il existe des fonctions qui ne sont pas calculables par machines de Turing.

Exemples de problèmes indécidables :

- Est-ce qu'une certaine machine de Turing va s'arrêter ?
- Est-ce qu'une certaine équation diophantienne a une solution ?
- Est-ce qu'un certain énoncé arithmétique est vrai ?

Programme WHILE (1)

On se donne un nombre arbitrairement grand de variables $X_0, ..., X_n$ qui prennent leur valeur dans les entiers naturels.

La valeur initiale de X_0 (et éventuellement X_1...) correspond à l'input du programme, sa valeur finale à son output.

Un programme peut réassigner des valeurs aux variables, et l'on dispose également de deux types d'instructions pour avoir des boucles.

Programme WHILE (2)

Le langage de programmation contient cinq types d'instructions :

- $X_i \leftarrow 0$, 'clear' : cette commande assigne la valeur 0 à la variable X_i.
- $X_i \leftarrow X_i + 1$, 'increment' : cette commande augmente de 1 la valeur assignée à X_i.
- $X_i \leftarrow X_j$, 'copy' : cette commande assigne à X_i la valeur de X_j et laisse la valeur de X_j inchangée.

Programme WHILE (3)

LOOP X_n ▶ P *ENDLOOP* X_n	Exécute le programme P k fois où k est la valeur **initiale** de X_n

P peut changer la valeur de X_n mais cela ne change pas le nombre d'itérations de la boucle

WHILE $X_n \neq 0$ ▶ P *ENDWHILE* $X_n \neq 0$	Exécute le programme P tant que la valeur **actuelle** de X_n est différente de 0

Exemples de programme WHILE

Un programme qui multiplie par 2 :

Exemples de programme WHILE

Un programme qui multiplie par 2 :

LOOP X_0
$X_0 \leftarrow X_0 + 1$
ENDLOOP X_0

Exemples de programme WHILE

Un programme qui multiplie par 2 :

LOOP X_0
  $X_0 \leftarrow X_0 + 1$
ENDLOOP X_0

Un programme qui soustrait 1 si $X_0 > 0$ et rend 0 sinon :

Exemples de programme WHILE

Un programme qui multiplie par 2 :

LOOP X_0
$X_0 \leftarrow X_0 + 1$
ENDLOOP X_0

Un programme qui soustrait 1 si $X_0 > 0$ et rend 0 sinon :

$X_1 \leftarrow X_1 + 1$
LOOP X_0
$X_2 \leftarrow X_2 + 1$
WHILE $X_1 \neq 0$
$X_2 \leftarrow 0$
ENDWHILE $X_1 \neq 0$
$X_1 \leftarrow 0$
ENDLOOP X_0
$X_0 \leftarrow X_2$

Equivalence

Fait

Une fonction f est calculable est par une machine de Turing
ssi
elle est calculable par un programme WHILE

On dispose également de résultats d'équivalences pour les autres modèles de calcul évoqués.

Machine de Turing et calculabilité

Mais ces modèles rendent-ils compte de notre notion intuitive de calcul ?

Thèse de Turing :

Une fonction est calculable
ssi
elle est calculable par une machine de Turing.

Perspectives

- Les limitations de la calculabilité sont-elles des limitations pour nous ou seulement des limitations pour les / certaines machines ?
- L'esprit peut-il être assimilé à une machine de Turing / un ordinateur ?

Perspectives

- Les limitations de la calculabilité sont-elles des limitations pour nous ou seulement des limitations pour les / certaines machines ?
- L'esprit peut-il être assimilé à une machine de Turing / un ordinateur ?
- Quel puissance de calcul est nécessaire à l'esprit pour quel type de processus de calcul ?

Perspectives

- Les limitations de la calculabilité sont-elles des limitations pour nous ou seulement des limitations pour les / certaines machines ?
- L'esprit peut-il être assimilé à une machine de Turing / un ordinateur ?
- Quel puissance de calcul est nécessaire à l'esprit pour quel type de processus de calcul ?
- Quelles sont les propriétés cognitivement pertinentes des procédures de calcul ?

Quelques cours à suivre...

- CO5 (D. Bonnay), **logique intro**

  Calcul propositionnel et logique du premier ordre
  lundi, 17h-19h, S1

- CA5 (G. Sandu), **logique avancée**

  Calculabilité et théorème d'incomplétude
  mardi, 14h-16h, S2

- C21 (P. Egré), **logique philosophique**

  Sémantique et épistémologie du vague
  mardi, 10h-12h, S2

- CO8 (M. Cozic), **théorie de la décision** et théorie des jeux

  S2