Automates et Calculabilité

– Introduction –

Session 1: 20 oct 2003

Uwe Nestmann

EPFL-I&C

Information & communication

- **`http//lamp.epfl.ch/teaching/aetc/2003/html/`**

- **`news://epflnews.epfl.ch/epfl.ic.cours.ac`**
  - recommandé pour questions publiques !

- I&C - IIF - LAMP2
  - pour questions non-publiques, si jamais . . .

`Uwe.Nestmann@epfl.ch`

`Johannes.Borgstroem@epfl.ch`

`Daniel.Buenzli@epfl.ch`

`Rachele.Fuzzati@epfl.ch`

`Simon.Kramer@epfl.ch`

Le sens du mot “science”

- en Physique ou Chimie?
  - mécanique, relativité, . . .
  - la vitesse de la lumière
  - la théorie de Heisenberg sur l’incertitude
- en Maths?
  - la “mère des sciences” !?
  - axiomes, algèbres, logique, calcul,
- et l’Informatique — Computer “Science”?
  - ???

Vrai ou faux ?

“Put the right kind of software into a computer,
and it will do whatever you want it to.

There may be limits on
what you can do with the machines themselves,

but *there are no limits on
what you can do with software*.”

TIME magazine (avril 1984)
citant un éditeur d’un magazine sur les logiciels.

Calculabilité & complexité

Calculabilité

l'étude des problèmes qui peuvent
être résolus par des algorithmes (procédures effectives)

Complexité

l'étude de l'efficacité des solutions algorithmiques

- ☐ algorithmes $\Rightarrow$ calculs
- ☐ modèle pour les calculs $\Rightarrow$ ***automates***, machines, ...
- ☐ description et complexité des problèmes
  $\Rightarrow$ *langages formels*

Et ça sert à quoi?

- économie de temps dans la recherche de solutions qui ne peuvent pas exister
- relation avec votre patron ou avec vos clients
- surprendre vos copains
- computer science $\neq$ just hacking programs
- interprétation philosophique ...

Objectifs

1. Quelle est la signification de la phrase
   “une fonction est calculable” ?

2. Y a-t-il des fonctions non-calculables ?

3. Peut-on déduire le pouvoir de calcul de
   certaines instructions des langages de programmation ?

Définition formelle des concepts:

- □ état, transition, non-déterminisme, . . .
- □ réduction, non-décidabilité

L'approche de Alan Turing, 1936

Machines de Turing

- un alphabet pour le ruban
- un ruban qui représente une mémoire infinie
- une tête de lecture
- un ensemble fini d'états
- une fonction de transition

L'approche de Alonzo Church, 1936

λ-calcul: Le calcul des fonctions

Syntaxe:

$$M \quad ::= \quad x \quad \Big| \quad \lambda x.M \quad \Big| \quad M\,M$$

Sémantique:

$$(\lambda x.M)N \to M[{}^{N}/_{x}]$$

si $M \to M'$ alors $MN \to M'N$

...

Quel est le bon modèle ?

- fonctions primitive-récursives (Dedekind, 1888)
- logiques combinatoires (Schönfinkel 1924, Curry)
- fonctions μ-récursives (Kleene, 1936)
- systèmes de correspondances (Post, 1936)
- “universal register machines” (Sheperdson, 1963)
- Fortran, Algol, Pascal, Ada, C, C++, SML, Java, C#, Scala, …

Thèse de Church-Turing, 1936:
Les fonctions calculables par une procédure effective sont les fonctions calculables par une machine Turing.

“The Most Amazing Fact” !?

Et la suite dans le curriculum?

- (4) *Langages Formels* (“A&C 2”):
  modèles vs langages, complexité
- (4) *Programmation IV*
  programmation fonctionnelle (λ-calcul)
- (5/7) *Compilation*
  automates et la reconnaissance des langages
- (5/7/9) *Concurrence: Langages, Programmation, et Théorie*
  programmation concurrente et répartie
- . . .

Support

- Livre principal du cours (en Anglais)
  Dexter Kozen: ***Automata and Computability***
  Springer-Verlag New York, 1997.
  utilisé par Kozen à Cornell University, et ailleurs . . .
  - bibliothèque centrale
  - bibliothèque de la faculté I&C
  - librairie EPFL, Springer, Amazon, . . .
  - les premiers 6 chapitres gratuitement . . .

- 2 livres/polycopiés en Français, à voir sur la page web
  mais ne couvrent pas exactement les sujets de ce cours

- les transparents sont accessibles sur la page web.

Plan du cours

1 — 20 oct 2003: introduction

2 — 27 oct 2003: techniques de preuves

3 — 3 nov 2003: K 2+3+4: chaînes, ensembles (réguliers), automates finis

4 — 10 nov 2003: K 5+6: non-détérminisme (NFA), “subset construction”

5 — 17 nov 2003: K 7+8: filtrage des motifs & expressions régulieres

6 — 24 nov 2003: K 9+A: expressions régulieres, DFA & algèbres de Kleene

7 — 1 dec 2003: K 11+12: limitations & lemme de gonflement

8 — 8 dec 2003: K 13+14: minimisation

9 — 15 dec 2003: K 15+16(+B): Myhill-Nerode (& collapsing NDA?)

---

10 — 5 jan 2004: K 28+29: machines de Turing & thèse de Church-Turing

11 — 12 jan 2004: K 31+32: machines universelles et diagonalisation

12 — 19 jan 2004: K 33+34: réduction & théorème de Rice

13 — 26 jan 2004: K 36+37+I: autres formalismes & programmes basés sur while

14 — 2 fév 2004: K 37+(38+39+K): λ-calcul (& peut-être l’incomplétude selon Gödel)

3 crédits

- examen écrit en juillet/septembre 2004
- exercices pas notés,
  pas de corrigé papier, mais corrigés si rendus
- Exemple d'examen le 2 février 2004
  - remplace la dernière série d'exercices
  - une semaine pour le rendre
  - pas noté, pas de corrigé papier, mais corrigé si rendu
  - par conséquent pas de bouns
- une session supplémentaire en juillet
  pour les questions.

Les exercices . . .

- schéma 2+1
- exercices toutes les semaines, **pas notés**
- une heure d'exercices avec les assistants à disposition
- possibilité de les discuter via le forum
- possibilité de rendre des solutions personelles jusqu'au jeudi de la même semaine, 9:15,
- "feedback" le lundi suivant avant/pendant l'heure d'exercices
- **A lire avant la prochaine fois**
  - *"Psychologists' tips on how and how not to learn"* by Wilfrid Hodges.
  - *"How to Read Mathematics"* by Shai Simonson and Fernando Gouveau

Psychologists' tips . . . (en Anglais)

2. "Trying to learn" facts is completely useless.
4. Do lots of examples . . .
5. Watching somebody else do examples, and copying other people's solutions, are completely useless . . .
6. Identifying your difficulty is half way to solving it.
   - Ask for advice like this:
     - I can get this far, but then I'm stuck because . . .
     - When I apply the rules like this, I get the wrong answer.
   - Never ask for advice like this:
     - I still can't do this kind of problem.
7. Don't try to learn a proof word by word. Learn its *shape*.

Répétition des notions d'algébre !

- relations
  - binaire, ternaire, ...
  - composition
- fonctions
  - ... vues comme relations
  - partielle/totale
  - injective
  - surjective
  - converse/inverse
- propriétés des relations
  - reflexivité, transitivité, symétrie, anti-symétrie
  - préordre, ordre partiel, noyeau, equivalence, ...

Répétition des techniques de preuve

- preuves sur des ensembles
- preuves par l’absurde, contre-exemples
- preuves déductives
- définitions & preuves inductives
  - induction naturelle
  - induction structurelle
- diagonalisation
- ...

Fonctions $\mathbb{N} \to \mathbb{N}$ calculable?

- définitions mathématiques $s(x) \stackrel{\text{def}}{=} \sqrt{x^2}$
- problèmes de comptage/recensement

$$f(x) \stackrel{\text{def}}{=} \text{“nombre de suisse qui touchent } x \text{ Frs. par ans”}$$

- basées sur des problèmes de décision

$$p(n) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{si } n \text{ est un nombre pair} \\ 0 & \text{si } n \text{ est un nombre impair} \end{cases}$$

$$g(n) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{si “la conjecture de Goldbach est vrai”} \\ 0 & \text{sinon} \end{cases}$$

Fonctions et programmes

Intuition: un programme représente une fonction calculable.

Soit $\mathcal{F}_{\mathbb{N}} \stackrel{\text{def}}{=} \{\, f \mid f : \mathbb{N} \to \mathbb{N} \,\}$ l'ensemble de toutes les fonctions (totales) sur les nombres naturels.

Supposons que chaque programme d'un langage X est une chaîne de symboles sur l'alphabet (fini) de ce langage. Appelons $\mathcal{P}_X$ l'ensemble des programmes ainsi définis.

Quelle est alors la relation exacte entre ces ensembles?

$$|\mathcal{F}_{\mathbb{N}}| \quad \begin{matrix} > \\ = \\ < \end{matrix} \quad |\mathcal{P}_X|$$

... et la preuve?

Cardinalité

Définition:
Deux ensembles A et B ont la même cardinalité, $|A| = |B|$, s'il y a une bijection $A \leftrightarrow B$.

A a une cardinalité plus grande que celle de B, $|A| > |B|$, s'il y a une surjection $A \rightarrow B$, mais pas de bijection $A \leftrightarrow B$.

Un ensemble A est *dénombrable* s'il a la même cardinalité que $\mathbb{N}$, si $|A| = |\mathbb{N}|$.

Un ensemble A est *non-dénombrable* s'il a une cardinalité plus grande que $\mathbb{N}$, si $|A| > |\mathbb{N}|$.

Question:
Est-ce que $A \subset B$ implique $|A| < |B|$?

$$|\mathbb{Q}| \stackrel{?}{=} |\mathbb{N}| \stackrel{?}{=} |\mathbb{R}|$$

Théorème à démontrer

$$|\mathcal{F}_{\mathbb{N}}| \qquad > \qquad |\mathcal{P}_X|$$

$$|\mathcal{F}_{\mathbb{N}}| \overset{(3)}{\geq} |2^{\mathbb{N}}| \qquad \overset{(2)}{>} \qquad |\mathbb{N}| \overset{(1)}{=} |\mathcal{P}_X|$$

On écrit $|A| \geq |B|$ si $|A| > |B|$ ou $|A| = |B|$.
Alors, pour démontrer $|A| \geq |B|$,
il suffit de trouver une surjection $A \to B$;
on ne dit rien sur l'existence eventuelle d'une bijection.

(0) Ordre lexicographique

Définition:
Soit A un alphabet.
Soit $\prec$ un ordre total sur A.
Soit $\omega_1 = a_1 \ldots, a_n$ et $\omega_2 = b_1 \ldots, b_m$ deux mots sur A.
On dit que ω_1 est lexicographiquement plus petit que ω_2, si

- ☐ $n < m$, ou
- ☐ $n = m$ et il y a un $i \leq n$ tel que:
  - $a_j = b_j$ pour tout $j < i$, et
  - $a_i \prec b_i$.

Exemple: $A =$ ASCII, $\prec =$ ASCII-énumeration
abc, abd, $aaaa$, sont en ordre lexicographique.

Théorème: L'ordre lexicographique est total.

(1) Bijection $b : \mathcal{P}_X \to \mathbb{N}$

Pour n'importe quel programme P,
soit *lexico.plus.petit*(P) le nombre de programmes qui sont plus petit selon l'ordre lexicographique sur $\mathcal{P}_X$

$$b : P \mapsto \textit{lexico.plus.petit}(P)$$

Exercice:
Essayer de démontrer que b est en fait une bijection.

(2) Cantor

Théorème:

Soit $A \neq \emptyset$ un ensemble.

Soit $2^A \stackrel{\text{def}}{=} \{ B \mid B \subseteq A \}$
l'ensemble de tous les sous-ensembles de A.

Alors $|A| < |2^A|$.

Exercice:
Essayer de donner une preuve
pour le cas ou A est un ensemble fini.

(3) Surjection $c : \mathcal{F}_{\mathbb{N}} \to 2^{\mathbb{N}}$

La fonction

$$\begin{array}{rccl} c : & (\mathbb{N} \to \mathbb{N}) & \to & 2^{\mathbb{N}} \\ & f & \mapsto & f(\mathbb{N}) \end{array} \qquad f(\mathbb{N}) := \{\, f(n) \mid n \in \mathbb{N} \,\}$$

associe avec toute fonction f sur les nombres naturels l'image de f, alors $c(f) := f(\mathbb{N})$.

Exemple: Soit $f : x \mapsto 3 * x$.
Alors $c(f) = 3 * \mathbb{N}$ (l'ensemble des nombres multiples de 3).

Exercice:

- ☐ La fonction c elle-même n'est pas une surjection. Pourquoi?
- ☐ Essayez d'adapter la définition de la fonction c tel que la nouvelle fonction devienne une surjection.
- ☐ optionnel: $|\mathcal{F}_{\mathbb{N}}| > |2^{\mathbb{N}}|$ ou $|\mathcal{F}_{\mathbb{N}}| = |2^{\mathbb{N}}|$?