

Informatique Théorique 3

2004/05

*Introduction à la théorie du calcul
(Automates, langages et calculabilité)*

Uwe Nestmann
I&C - IIF - LAMP2

Pour quoi «3» ?

selon le plan d'étude de la SIN:

Info Théo 1
Info Théo 2

- ➔ Logique élémentaire
(comme compensation partielle de Info Théo 2)
- ➔ Préliminaires
(Algèbres discrètes, Théorie des Nombres)
(comme compensation partielle de Info Théo 1)

Vrai ou faux?

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 (1984)

[citant un éditeur d'un magazine sur les logiciels]

Computer « Science » ? Comput**ation** Science !

Calculabilité (possibilité)

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

➔ ce cours-ci (3ème semestre)

Complexité (coûts)

l'étude de l'efficacité des solutions algorithmiques

➔ cours d'*Algorithmique* (4ème semestre)

Ensuite ...

Algorithmique

Programmation IV

Compilation

Software Engineering

Artificial Intelligence

Computer-Aided Verification

Concurrency Semantics

Distributed Computing

Type Systems

...

Support du cours

page web & forum de discussion

livres recommandés

notes (formulaire)

transparentes

quiz

exercices encadrés par 4-5 assistants

Information & Communication

<http://lamp.epfl.ch/teaching/it3/2004/html/>

<news://epflnews.epfl.ch/epfl.ic.cours.it3>

aussi par email ...

BC 349

I&C-IIF-LAMP2

Bibliographie

[HMU03] John Hopcroft, Rajeev Motwani, and Jeffrey Ullman. *Introduction to Automata Theory, Languages and Computation*. Pearson Education International, 2003. ISBN 0321210298.

[Koz97] Dexter Kozen. *Automata and Computability*. Springer Verlag New York Inc., 1997.

[Sch95] Uwe Schöning. *Theoretische Informatik — kurzgefaßt*. Spektrum Lehrbuch. Spektrum Akademischer Verlag, 1995. 2. Auflage.

[Sch01] Carol Schumacher. *Chapter Zero — Fundamental Notions of Abstract Mathematics*. Addison Wesley, 2001.

[Sip97] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.

[Wol01] Pierre Wolper. *Introduction à la calculabilité — Cours et exercices corrigés*. Dunod, Paris, 2001. 2^e édition.

Formulaire & transparents du cours

- La première semaine, ils sont imprimés et distribués par nous.
- Après, ils seront accessibles sur la page web, normalement, par semaine.
- Pour l'examen, seulement le formulaire sera admis.

Structure du cours

S1:

Préliminaires & Non-calculabilité

S2:

Grammaires

S3-6:

Langages réguliers

S7-9:

Langages hors-contextes

S10:

... & Examen de Noël (pas noté)

S11-14:

Langages récursivement énumérables

Examen en février/mars (en utilisant le formulaire)

Structure d'une semaine de cours

lundi

discussion du quiz
cours
cours
exercices <i>(avec assistants)</i>

mercredi

cours
cours, quiz, début exercices
exercice <i>(avec assistants)</i>

Semaine I

- §1.3 Fonctions versus programmes
- §0 et §1.1-1.2 :
tout ce qui est nécessaire pour §1.3
 - fonctions d'ordre supérieur
 - ensemble des parties
 - cardinalité
 - alphabets, mots, et langages
- Le formulaire + les transparents ne contiennent pas tout le matériel ...
- Exemples + démonstrations sur tableau noir

Fonctions définie & calculables ?

$$f : x \mapsto x^2 \qquad f(x) \triangleq x^2$$

$f(x) \triangleq$ "nombre de suisse qui touchent x Frs. par ans"

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

$$g(n) \triangleq \begin{cases} 1 & \text{si George W Bush sera réélu} \\ 0 & \text{sinon} \end{cases}$$

Fonctions calculable

Approche:

Intuitivement, ou pratiquement,
une fonction devrait être appelée *calculable*
si on peut écrire un programme qui la calcule.

Attention:

on fait ici référence à un mécanisme d'*exécution*...

Sujet de l'étude:

automates, machines, ...

comme *modèles d'algorithmes* (programmes)

et de *calculs* (exécution)

Non-calculabilité

Il existe plus de fonctions que de programmes.

Vrai? Faux? Evident?

Problème!

Nous cherchons à comparer la taille des ensembles infinis.

Paradoxe de Galilei : comparer le nombre d'éléments de

\mathbb{N}

$\{n^2 \mid n \in \mathbb{N}\}$

Cardinalité (I)

0.5.2 Définition (Cardinalité) Soit A et B deux ensembles.

1. A et B ont la même cardinalité, $\text{card}(A) = \text{card}(B)$, s'il existe une application bijective de A dans B . On dit aussi que A et B sont équipotents.
2. A a une cardinalité plus petite que celle de B , $\text{card}(A) \leq \text{card}(B)$, s'il existe une application injective de A dans B .
3. A a une cardinalité strictement plus petite que celle de B , $\text{card}(A) < \text{card}(B)$, si $\text{card}(A) \leq \text{card}(B)$ mais $\text{card}(A) \neq \text{card}(B)$.

\mathbb{N}

\subsetneq

\mathbb{Q}

\subsetneq

\mathbb{R}

Cardinalité (II)

0.5.3 Définition (Dénombrable) Soit A un ensemble.

1. A est *dénombrable* si A est fini ou équipotent à \mathbb{N} .
2. A est *infiniment dénombrable* si A est infini et dénombrable.
3. A est *non-dénombrable* si $\text{card}(\mathbb{N}) < \text{card}(A)$.

Fonctions vs programmes (I)

1.3.1 Définition

$$\mathbf{F}_{\mathbb{N}} \triangleq \{ f \mid f : \mathbb{N} \rightarrow \mathbb{N} \}$$

dénote l'ensemble de toutes les fonctions (totales) sur \mathbb{N} .

1.3.2 Définition Soit Σ l'alphabet d'un langage X de programmation. Alors,

$$\mathbf{P}_X \triangleq \Sigma^*$$

dénote l'ensemble des mots/programmes sur Σ (voir §1.1 et §1.2).

Alphabets, Mots (I)

1.1.1 Définition (Alphabet) On appelle *alphabet* tout ensemble fini Σ . Les éléments d'un alphabet sont traditionnellement appelés *symboles*, *lettres* ou *caractères*. \square

Pour un alphabet $\Sigma \triangleq \{s_1, s_2, \dots, s_k\}$, on considère souvent l'ordre total \preceq défini par $s_i \preceq s_j$ si et seulement si $i \leq j$.

1.1.2 Définition (Mots) Un *mot* w sur un alphabet Σ est une *séquence finie* de lettres de Σ .

Autrement dit, un mot w est un n -uplet (a_1, a_2, \dots, a_n) de lettres de Σ où $n \in \mathbb{N}$ est appelé la *longueur* de w , notée $|w|$. Si $n = 0$, w est l'unique mot de longueur nulle appelé *mot vide* que l'on note ϵ . La i^e *projection* $(w)_i$ de w est la lettre a_i pour $1 \leq i \leq n$.

L'ensemble des mots sur un alphabet Σ est noté Σ^* . L'ensemble des mots non vides, c'est à dire $\Sigma^* \setminus \{\epsilon\}$ est noté Σ^+ . \square

Alphabets, Mots (II)

1.1.3 Définition (Concaténation de mots) Soit $w \triangleq (a_1, \dots, a_n) \in \Sigma^*$ et $w' \triangleq (b_1, \dots, b_m) \in \Sigma^*$, la *concaténation* $w \cdot w'$ de w avec w' est définie par,

$$w \cdot w' \triangleq (a_1, \dots, a_n, b_1, \dots, b_m)$$

1.1.4 Théorème (Σ^*, \cdot) est un monoïde d'élément neutre ϵ .

1.1.5 Notation Pour un alphabet $\Sigma \triangleq \{s_1, \dots, s_n\}$ on identifie la lettre $s_i \in \Sigma$ avec le mot $(s_i) \in \Sigma^*$ de longueur 1. Ceci nous permet d'écrire pour $w \triangleq (a_1, a_2, \dots, a_n) \in \Sigma^*$, $w = a_1 \cdot a_2 \cdot \dots \cdot a_n = a_1 a_2 \dots a_n$ \square

**Pour mercredi:
Lire § 1.1.6-9**

Langages

1.2.1 Définition (Langage) On appelle *langage* sur un alphabet Σ tout sous-ensemble de Σ^* . Un langage n'est donc rien d'autre qu'un ensemble de mots. □

Nous utilisons les meta-variables $A, B, C \dots, L, \dots$ pour les langages.

Les deux langages \emptyset et $\{\epsilon\}$ sont des langages sur n'importe quel alphabet.

Pour construire des langages à partir d'autres langages, nous utilisons notamment les opérations suivantes.

***Pour mercredi:
Lire § 1.2.2***

Fonctions vs programmes (II)

1.3.2 Définition Soit Σ l'alphabet d'un langage X de programmation. Alors,

$$\mathbf{P}_X \triangleq \Sigma^*$$

dénote l'ensemble des mots/programmes sur Σ (voir §1.1 et §1.2).

1.3.3 Définition Soit $\text{exec}_X : \mathbf{P}_X \rightarrow (\mathbb{N} \rightarrow \mathbb{N})$ un interpréteur de X qui associe avec un programme p une fonction $\text{exec}_X(p) : \mathbb{N} \rightarrow \mathbb{N}$. Notons que tous les programmes ne sont pas associés avec des telles fonctions. Nous appelons une fonction f calculable par X si $f \in \text{img}(\text{exec}_X)$, alors si

$$\exists p \in \mathbf{P}_X . \text{exec}_X(p) = f$$

Nous dénotons par

$$\mathbf{F}_X \triangleq \{ f : \mathbb{N} \rightarrow \mathbb{N} \mid f \text{ calculable par } X \} \subseteq \mathbf{F}_{\mathbb{N}}$$

l'ensemble de fonctions calculables par (des programmes en) X .

1.3.4 Théorème Pour tout X : $\text{card}(\mathbf{F}_X) < \text{card}(\mathbf{F}_{\mathbb{N}})$.

1.3.4 Théorème Pour tout X : $\text{card}(\mathbf{F}_X) < \text{card}(\mathbf{F}_{\mathbb{N}})$.

Preuve:

$$\text{card}(\mathbf{F}_X) \stackrel{(0)}{\leq} \text{card}(\mathbf{P}_X) \stackrel{(1)}{\leq} \text{card}(\mathbb{N}) \stackrel{(2)}{<} \text{card}(\mathcal{P}(\mathbb{N})) \stackrel{(3)}{\leq} \text{card}(\mathbf{F}_{\mathbb{N}})$$

lundi:

comprendre les énoncés

mercredi:

comprendre les démonstrations

Décomposition & Induction

1.1.6 Lemme (Décomposition d'un mot) Soit w un mot sur un alphabet Σ , nous avons,

Soit $w = \epsilon$

ou alors $\exists a \in \Sigma, w' \in \Sigma^*. w = a \cdot w'$ avec $|w'| = |w| - 1$ □

1.1.7 Lemme (Principe d'induction sur les mots) Soit $P(w)$ un prédicat sur les mots d'un alphabet Σ .

Si $P(\epsilon)$

et $\forall w \in \Sigma^*. P(w) \Rightarrow \forall a \in \Sigma. P(a \cdot w)$

alors $\forall w \in \Sigma^*. P(w)$. □

Remarquons que les deux derniers lemmes peuvent se reformuler en décomposant les mots par la droite.

Ordres (I)

0.3.5 Définition (Ordres) Soit A un ensemble et $R \subseteq A \times A$ une relation.

Pré-ordre. Le couple (A, R) est un **pré-ordre** si R est réflexive et transitive.

Ordre partiel. Le pré-ordre (A, R) est un **ordre (partiel)** si R est antisymétrique.

Ordre total. L'ordre partiel (A, R) est un **ordre total** si

$$\forall a, b \in A. aRb \vee bRa$$

Ordre strict. Le couple (A, R) est un **ordre strict** si R est irreflexive et transitive.

En général, on utilise les symboles \leq, \preceq et \sqsubseteq pour les ordres, $<, \prec$ et \sqsubset pour les ordres stricts. □

Ordres (II)

1.1.8 Définition (Ordre sur les mots) Soit un alphabet Σ et un ordre total (Σ, \preceq) (et l'ordre strict induit \prec).

Ordre alphabétique. L'ordre alphabétique (Σ^*, \ll_d) est la plus petite relation sur Σ^* qui satisfait les règles suivantes.

$$A_1 \frac{}{\epsilon \ll_d w} \quad w \neq \epsilon$$

$$A_2 \frac{}{a \cdot w \ll_d a' \cdot w'} \quad a \prec a'$$

$$A_3 \frac{w \ll_d w'}{a \cdot w \ll_d a \cdot w'}$$

Ordres (III)

Ordre lexicographique. L'ordre lexicographique (Σ^*, \ll_1) est la plus petite relation sur Σ^* qui satisfait les règles suivantes.

$$\begin{array}{l} L_1 \frac{}{\epsilon \ll_1 w} \quad w \neq \epsilon \qquad L_2 \frac{}{w \cdot a \ll_1 w \cdot a'} \quad a \prec a' \\ L_3 \frac{w \ll_1 w'}{w \cdot a \ll_1 w' \cdot a'} \end{array}$$

1.1.9 Théorème

1. L'ordre alphabétique est total.
2. L'ordre lexicographique est total.

□

Une fonction non-calculable

1.1.9 Théorème Supposons qu'il existe une énumération f_1, f_2, \dots de toutes les fonctions sur les nombres naturels. Alors, la fonction définie par

$$\text{diag}(i) \triangleq f_i(i) + 1$$

ne peut pas apparaître dans cette liste de toutes les fonctions.

Preuve par diagonalisation...