# Automates et Calculabilité

# – Automates finis & langages réguliers –

# Leçon 3: 3 nov 2003

Uwe Nestmann

EPFL-I&C



## Problèmes (de décision)

Question fondamentale: ***quels problèmes sont solubles par un programme exécuté sur un ordinateur?***

**Exemple:** Déterminer si un nombre naturel est pair ou impair.

- ☐ Un problème est une question générique qui s'applique à un ensemble d'éléments.
- ☐ Pour chaque élément particulier (instance) de l'ensemble il existe une réponse.
- ☐ On appelle *problème de décision* les problèmes à réponse binaire (oui/non, ou 0/1).

Pour la question fondamentale ci-dessus,
***il suffit d'étudier les problèmes de décision***,
(ce qui est plus simple . . . ).



## La formalisation des problèmes

- ☐ entrées possibles: un ensemble $E$
- ☐ réponses positives: un sous-ensemble $R \subseteq E$

$$\text{problème\_X} : E \rightarrow \{\,\text{oui}, \text{non}\,\}$$
$$a \mapsto \begin{cases} \text{oui} & \text{si } a \in R \\ \text{non} & \text{si } a \notin R \end{cases}$$

- ☐ Chaque (instance d'un) problème peut être représentée (ou: codée) par une chaîne finie (= un *mot*) de symboles/caractères ($E = A^*$) sur un alphabet fini $A$.
- ☐ Résoudre un problème consiste à décider si une instance est dans l'ensemble des réponses positives . . .
- ☐ ***Un problème est donc caractérisé par l'ensemble des mots qui sont les instances positives.***



## Attention (I)

On parlera des concepts différents de:

- ☐ caractères ($a, b, c, \ldots, 0, 1, 2, \ldots$)
- ☐ chaînes de caractères ($x, aaabbbaa, a^n, \epsilon, \ldots$)
- ☐ ensembles de caractères ($\Sigma, \emptyset, \ldots$)
- ☐ ensembles de chaînes de caractères ($\Sigma^*, A, \emptyset, \ldots$)

(Les transparents des deux premières leçons seront adaptés prochainement pour correspondre à cette notation.)

# Alphabets, mots, ... selon Kozen

Les alphabets sont dénotés par $\Sigma$.
L'ensemble $\Sigma^*$ des mots sur $\Sigma$ est défini par

$$\Sigma^* \stackrel{\text{def}}{=} \{\epsilon\} \cup \{x \mid x = a_1 \cdot \ldots \cdot a_n \ \wedge\ 0 \neq n \in \mathbb{N} \ \wedge\ a_i \in \Sigma\}$$

où $x_1 \cdot x_2$ est la concaténation (associative) de deux mots et les mots de longueur $1$ sont représentés par leur symbole.
NB: $\omega \cdot \epsilon \stackrel{\text{def}}{=} \epsilon \cdot \omega \stackrel{\text{def}}{=} \omega$.

- ☐ On utilise $x, y, z$ comme meta-variables pour les mots.
- ☐ La *longueur* d'un mot $x$ est dénotée par $|x|$.
  Elle est compositionnelle: $|xy| = |x| + |y|$.
  L'unique mot de longueur $0$ est dénoté par $\epsilon$, $|\epsilon| = 0$.
- ☐ Notons que pour tout $\Sigma$, on a $\epsilon \in \Sigma^*$.
  Par convention, $\emptyset^* \stackrel{\text{def}}{=} \{\epsilon\}$. Notons que $\emptyset \neq \{\epsilon\} \neq \epsilon \neq \emptyset$.



# Attention (II)

Pour un alphabet fixe $A$, l'opération de concaténation $\cdot$ est définie sur les chaînes de caractères, son type est $A^* \times A^* \to A^*$. Par exemple

$$abc \cdot cba \qquad (x \cdot y) \cdot z \qquad x \cdot (y \cdot z) \qquad x \cdot \epsilon \cdot abaca$$

Pourtant, on l'utilise aussi dans des cas comme

$$a \cdot b \qquad aa \cdot b \qquad (a \cdot \epsilon) \cdot c \qquad x \cdot a$$

où l'on interprète implicitement les symboles ($a, b, c$) de l'alphabet comme étant des chaînes de longueur $1$.

(Il existe aussi des représentations des chaînes de caractères où l'on distingue explicitement entre la concaténation, la pré- et la post-composition d'un symbole à une chaîne de caractères.)



# Opérations sur les mots

Soit $\Sigma$ un alphabet. Soit $x, y, z \in \Sigma^*$. Alors:

1. La concaténation est associative: $(xy)z = x(yz)$
2. Le mot vide $\epsilon$ est l'élément neutre: $\epsilon x = x\epsilon = x$

Ces propriétés signifient que $(\Sigma^*, \cdot, \epsilon)$ est un monoïde.

- ☐ La *répétition* des symboles, est définie par induction:

$$a^0 \stackrel{\text{def}}{=} \epsilon \qquad a^{n+1} \stackrel{\text{def}}{=} a^n a$$

- ☐ La *répétition* des chaînes, est définie par induction:

$$x^0 \stackrel{\text{def}}{=} \epsilon \qquad x^{n+1} \stackrel{\text{def}}{=} x^n x$$

- ☐ $\#a(x)$ dénote le nombre de $a$ contenu dans le mot $x$.
- ☐ $x$ est un *préfixe* de $z$ s'il existe un $y$ tel que $z = xy$.
  Un préfixe $x$ de $z$ est un *préfixe propre* de $z$ si $x \neq \epsilon \wedge x \neq z$.



# Exercice

Soit $\Sigma$ un alphabet fini.
Soit $\ll_{\mathrm{p}}$ la relation binaire sur $\Sigma^* \times \Sigma^*$ qui est définie par:

$$x \ll_{\mathrm{p}} y \qquad \text{si } x \text{ est un préfixe de } y$$

Quelles sont les propriétés de $(\Sigma^*, \ll_{\mathrm{p}})$ ?

- ☐ transitive ?
- ☐ reflexive ?
- ☐ stricte ?
- ☐ partielle ou totale ?

# Opérations sur les ensembles de mots

Soit $\Sigma$ un alphabet fini, et $A, B \subseteq \Sigma^*$.

- ☐ Opérations standard:
  union ($A \cup B$), intersection ($A \cap B$), complément ($\tilde{}A, \overline{A}$)
- ☐ *Concaténation*: $AB \stackrel{\text{def}}{=} \{ xy \mid x \in A \wedge y \in B \}$
- ☐ *Répétition*: $A^0 \stackrel{\text{def}}{=} \{ \epsilon \}$ $\qquad A^{n+1} \stackrel{\text{def}}{=} A^n A$

**Exemple:** Soit $A = \{ acdc \}$ et $B = \{ \epsilon, bbc \}$ et $C = \emptyset$.
Quels sont les éléments de $(AB)A$, $A(BA)$, $((AB)A)^2$, et $AC$ ?

- ☐ *Fermeture itérative* ou *Fermeture (de Kleene)*:

$$A^* \stackrel{\text{def}}{=} \bigcup_{n \geq 0} A^n \qquad\qquad A^+ \stackrel{\text{def}}{=} AA^*$$

Propriétés: $A^*A^* = A^*$; $A^{**} = A^*$; $\emptyset^* = \{ \epsilon \}$;



# Remarque

Ce qui est important c'est la taille d'un alphabet.
Le choix des caractères individuels n'est pas important.
L'alphabet $\{ 1, 2, 3 \}$ est "aussi bon" que l'alphabet $\{ a, b, c \}$.



# Automates finis

- ☐ Une abstraction d'une procédure effective ou d'un ordinateur qui n'est capable d'exécuter qu'un seul programme.
- ☐ La notion d'***état*** représente de manière abstraite:
  - une mémoire (ou des registres, dont le compteur) qui contient les instructions.
  - une mémoire servant à stocker des données.
- ☐ La notion de ***transition*** représente de manière abstraite:
  - un cycle de calcul atomique.
  - une relation$^{(*)}$ qui associe un état avec un état suivant.
- ☐ Si le nombre d'états est fini, on parle d'*automate fini*.
- ☐ $(*)$ Si, en plus, la relation est une *fonction*,
  on parle d'***automate fini déterministe*** (AFD, DFA).



# Automates finis, formellement

Un automate fini déterministe (AFD) est une structure

$$M = (\mathcal{Q}, \Sigma, \delta, s, F)$$

où

- ☐ $\mathcal{Q}$ est un ensemble fini d'*états*,
- ☐ $\Sigma$ est un ensemble fini de symboles, *l'alphabet (d'entrée)*,
- ☐ $\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q}$ est la *fonction de transition*,
- ☐ $s \in \mathcal{Q}$ est l'*état initial*,
- ☐ $F \subseteq \mathcal{Q}$ est un sous-ensemble de $\mathcal{Q}$, les *états accepteurs*.

Un automate peut être défini de manière algébrique, graphique, ou par un tableau.

# Exemple

- $\mathcal{Q} \stackrel{\text{def}}{=} \{0, 1, 2, 3\}$
- $\Sigma \stackrel{\text{def}}{=} \{a, b\}$
- $$\begin{array}{rclrcl} \delta : \mathcal{Q} \times \Sigma & \to & \mathcal{Q} & & & \\ (0, a) & \mapsto & 1 & (0, b) & \mapsto & 0 \\ (1, a) & \mapsto & 2 & (1, b) & \mapsto & 1 \\ (2, a) & \mapsto & 3 & (2, b) & \mapsto & 2 \\ (3, a) & \mapsto & 3 & (3, b) & \mapsto & 3 \end{array}$$
- $s \stackrel{\text{def}}{=} 0$
- $F \stackrel{\text{def}}{=} \{3\}$



# Fonctionnement d'un automate

Un automate est un mécanisme qui résout un problème, c'est-à-dire qui reconnaît les mots d'un *langage* (les instances positives d'un problème).

On parcours un mot de gauche à droite en partant de l'état initial; à chaque étape, on saute vers l'état déterminé par la fonction de transition selon l'état et le symbole actuel.

Si à la fin du mot (on est sûr d'y arriver, car la fonction de transition est totale et le mot est de longueur finie !) on se trouve dans un état accepteur, le mot est *accepté* par l'automate.



# Formellement, ...

Soit $(\mathcal{Q}, \Sigma, \delta, s, F)$ un AFD.
Nous définissons la fonction $\widehat{\delta} : \mathcal{Q} \times \Sigma^* \to \mathcal{Q}$ par

$$\begin{array}{rcl} \widehat{\delta}(q, \epsilon) & \stackrel{\text{def}}{=} & q \\ \widehat{\delta}(q, xa) & \stackrel{\text{def}}{=} & \delta(\widehat{\delta}(q, x), a) \end{array}$$

qui représente la version *multi-étapes* de $\delta$.

NB: Reconnaître le caractère inductif de cette définition.

Montrer que $\widehat{\delta}(q, a) = \delta(q, a)$.



# Langage accepté

**Définition:**
Un mot $x$ est dit *accepté* par un automate $M$ si

$$\widehat{\delta}(s, x) \in F$$

où $s$ est l'état initial, et $F$ l'ensemble des états accepteurs de $M$.
Dans le cas contraire, $x$ est *rejeté* par $M$.

Le *langage accepté par* $M$, dénoté par $L(M)$, est l'ensemble des mots qui sont acceptés par $M$.

$$L(M) \stackrel{\text{def}}{=} \{ x \in \Sigma^* \mid \widehat{\delta}(s, x) \in F \}$$

Un sous-ensemble $A \subseteq \Sigma^*$ est dit *régulier* s'il existe un automate fini $M$ tel que $L(M) = A$.

# Exemples

1. Quel est le langage
   de l'automate défini sur le transparent 13 ?

2. Soit $\Sigma = \{ a, b \}$.
   Quels sont les ensembles réguliers ci-dessous ?
   Et pourquoi?
   - ☐ $\Sigma^*$
   - ☐ $\{ xaaay \mid x, y \in \Sigma^* \}$
   - ☐ $\{ a^n b^n \mid n \in \mathbb{N} \}$
   - ☐ $\{x \mid x \in \Sigma^* \ \wedge \ (\exists n \in \mathbb{N} : \#a(x) = 2 * n) \ \wedge \ (\exists n \in \mathbb{N} : \#b(x) = 3 * n) \ \}$



# Produit d'automates

Soit $A, B \subseteq \Sigma^*$ deux langages réguliers.
Alors, il existe deux automates

$$\begin{aligned} M_1 &= (Q_1, \Sigma, \delta_1, s_1, F_1) \\ M_2 &= (Q_2, \Sigma, \delta_2, s_2, F_2) \end{aligned}$$

tels que $L(M_1) = A$ et $L(M_2) = B$.

Comment démontrer que le langage $A \cap B$ est aussi régulier ?
On construit un automate $M_{1\cap 2}$ tel que $L(M_{1\cap 2}) = A \cap B \ldots$

$$Q_{1\cap 2} \stackrel{\text{def}}{=} Q_1 \times Q_2 \qquad s_{1\cap 2} \stackrel{\text{def}}{=} (s_1, s_2) \qquad F_{1\cap 2} \stackrel{\text{def}}{=} F_1 \times F_2$$
$$\delta_{1\cap 2}((p,q), a) \stackrel{\text{def}}{=} (\delta_1(p,a), \delta_2(q,a))$$



# Propriétés de clôture (I)

Soit $A_1, A_2 \subseteq \Sigma^*$ deux langages réguliers,
il existe donc $M_i$ tel que $M_i$ accepte $A_i$ pour $i \in \{ 1, 2 \}$.
Alors,

- ☐ $A_1 \cap A_2$ est régulier.

  Prendre le produit $M_{1\cap 2}$ de $M_1$ et $M_2$.

  **Lemme:** Pour tout $x \in \Sigma^*$, $\widehat{\delta_{1\cap 2}}((p,q), x) = (\widehat{\delta_1}(p,x), \widehat{\delta_2}(q,x))$.

  **Théorème:** $L(M_{1\cap 2}) = L(M_1) \cap L(M_2)$.



# Propriétés de clôture (II)

- ☐ $\overline{A_1}$ est régulier.

  Construire $\overline{M_1} \ldots$ par $\overline{F_1}$ (par rapport à $Q_1$).

  **Théorème:** $L(\overline{M_1}) = \overline{L(M_1)}$.

- ☐ $A_1 \cup A_2$ est régulier.

  Construire $M_{1\cup 2} \ldots$ par $F_{1\cup 2} \stackrel{\text{def}}{=} (F_1 \times Q_2) \cup (Q_1 \times F_2)$.
  (Se souvenir que $A \cup B = \overline{\overline{A} \cap \overline{B}}$.)

  **Théorème:** $L(M_{1\cup 2}) = L(M_1) \cup L(M_2)$.