

Informatique Théorique 3

2004/05

Semaine 4

Définition par des règles d'inférence

Exemple (juste pour motiver les formes):

X : le plus petit ensemble qui satisfait des règles des formes

$$\frac{c \text{ in } C}{c \text{ in } X} \quad \frac{x_1 \text{ in } X \quad \dots \quad x_n \text{ in } X}{f(x_1, \dots, x_n) \text{ in } X} \quad \frac{y \text{ in } X}{x \text{ in } X} P(x, y)$$

où C ensemble, f opération n -aires, P prédicat.

Alors, on est sûr qu'il n'y pas d'autres éléments dans X que ceux déduisables par ces règles.

Ceci nous offre aussi un **principe d'induction** sur X :
il suffit de définir/vérifier tous les cas de ces règles.

Noter: les *relations* et *fonctions* sont aussi des ensembles!

Expressions régulières

2.4.1 Définition (Expressions régulières) Soit Σ un alphabet. Pour chaque $a \in \Sigma$, on se donne un nouveau symbole que l'on note \mathbf{a} . On se donne également les symboles ϵ , \emptyset , l'opérateur unaire $*$, les opérateurs binaires $+$, \cdot . On définit alors **l'ensemble \mathbf{RE}_Σ** des *expressions régulières sur Σ* **comme étant le plus petit ensemble satisfaisant les règles**

$$\frac{a \in \Sigma}{\mathbf{a} \in \mathbf{RE}_\Sigma}$$

$$\frac{}{\epsilon \in \mathbf{RE}_\Sigma}$$

$$\frac{}{\emptyset \in \mathbf{RE}_\Sigma}$$

$$\frac{\mathbf{x} \in \mathbf{RE}_\Sigma}{(\mathbf{x}^*) \in \mathbf{RE}_\Sigma}$$

$$\frac{\mathbf{x} \in \mathbf{RE}_\Sigma \quad \mathbf{y} \in \mathbf{RE}_\Sigma}{(\mathbf{x} + \mathbf{y}) \in \mathbf{RE}_\Sigma}$$

$$\frac{\mathbf{x} \in \mathbf{RE}_\Sigma \quad \mathbf{y} \in \mathbf{RE}_\Sigma}{(\mathbf{x} \cdot \mathbf{y}) \in \mathbf{RE}_\Sigma}$$

Sémantique

2.4.3 Définition (Langage dénoté par une expression régulière) Soit Σ un alphabet. Le langage $L(\mathbf{x})$ dénoté par une expression régulière $\mathbf{x} \in \mathbf{RE}_\Sigma$ est défini par induction structurelle sur l'expression \mathbf{x} comme suit :

$$L(\mathbf{a}) \triangleq \{ a \}$$

$$L(\epsilon) \triangleq \{ \epsilon \}$$

$$L(\emptyset) \triangleq \emptyset$$

$$L(\mathbf{x} + \mathbf{y}) \triangleq L(\mathbf{x}) \cup L(\mathbf{y})$$

$$L(\mathbf{x} \cdot \mathbf{y}) \triangleq L(\mathbf{x}) L(\mathbf{y})$$

$$L(\mathbf{x}^*) \triangleq L(\mathbf{x})^*$$

2.4.5 Théorème Soit \mathbf{x} une expression régulière sur un alphabet Σ . Alors, il existe un AFN_ϵ M sur Σ tel que $L_\epsilon(M) = L(\mathbf{x})$.

Équivalences et ordres

2.4.6 Définition (Expressions régulières équivalentes et ordonnées)

Deux expressions régulières x et y sont équivalentes, noté $x \approx y$, si les langages qu'elles dénotent sont égaux :

$$x \approx y \Leftrightarrow L(x) = L(y)$$

Similairement, x et y sont ordonnées, noté $x \lesssim y$, si le langage dénoté par x est inclus dans le langage dénoté par y , i.e.

$$x \lesssim y \Leftrightarrow L(x) \subseteq L(y)$$

2.4.7 Lemme Soit x et y deux expressions régulières. Alors :

1. \approx est une équivalence.

2. $x \lesssim y \Leftrightarrow x + y \approx y$

3. $x \lesssim y \wedge y \lesssim x \Rightarrow x \approx y$

Lois de Kleene (I)

2.4.11 Proposition

$$\mathbf{x} + (\mathbf{y} + \mathbf{z}) \approx (\mathbf{x} + \mathbf{y}) + \mathbf{z} \quad (2.1)$$

$$\mathbf{x} + \mathbf{y} \approx \mathbf{y} + \mathbf{x} \quad (2.2)$$

$$\mathbf{x} + \mathbf{\emptyset} \approx \mathbf{x} \quad (2.3)$$

$$\mathbf{x} + \mathbf{x} \approx \mathbf{x} \quad (2.4)$$

$$\mathbf{x}(\mathbf{yz}) \approx (\mathbf{xy})\mathbf{z} \quad (2.5)$$

$$\mathbf{x}\epsilon \approx \mathbf{x} \quad (2.6)$$

$$\epsilon\mathbf{x} \approx \mathbf{x} \quad (2.7)$$

$$\mathbf{x}\mathbf{\emptyset} \approx \mathbf{\emptyset} \quad (2.8)$$

$$\mathbf{\emptyset}\mathbf{x} \approx \mathbf{\emptyset} \quad (2.9)$$

Lois de Kleene (II)

$$\mathbf{x}(y + z) \approx \mathbf{xy} + \mathbf{xz} \quad (2.10)$$

$$(\mathbf{x} + \mathbf{y})\mathbf{z} \approx \mathbf{xz} + \mathbf{yz} \quad (2.11)$$

$$\epsilon + \mathbf{xx}^* \approx \mathbf{x}^* \quad (2.12)$$

$$\epsilon + \mathbf{x}^*\mathbf{x} \approx \mathbf{x}^* \quad (2.13)$$

$$\mathbf{y} + \mathbf{xz} \lesssim \mathbf{z} \Rightarrow \mathbf{x}^*\mathbf{y} \lesssim \mathbf{z} \quad (2.14)$$

$$\mathbf{y} + \mathbf{zx} \lesssim \mathbf{z} \Rightarrow \mathbf{yx}^* \lesssim \mathbf{z} \quad (2.15)$$

Contextes

2.4.8 Définition (Contextes des expressions régulières) Soit Σ un alphabet. On définit l'ensemble \mathbf{CRE}_Σ des *contextes des expressions régulières sur Σ* comme étant le plus petit ensemble satisfaisant les règles :

$$\frac{}{[\cdot] \in \mathbf{CRE}_\Sigma} \qquad \frac{\mathbf{c} \in \mathbf{CRE}_\Sigma}{(\mathbf{c}^*) \in \mathbf{CRE}_\Sigma}$$

$$\frac{\mathbf{x} \in \mathbf{RE}_\Sigma \quad \mathbf{c} \in \mathbf{CRE}_\Sigma}{(\mathbf{x} + \mathbf{c}) \in \mathbf{CRE}_\Sigma}$$

$$\frac{\mathbf{x} \in \mathbf{RE}_\Sigma \quad \mathbf{c} \in \mathbf{CRE}_\Sigma}{(\mathbf{c} + \mathbf{x}) \in \mathbf{CRE}_\Sigma}$$

$$\frac{\mathbf{x} \in \mathbf{CRE}_\Sigma \quad \mathbf{c} \in \mathbf{CRE}_\Sigma}{(\mathbf{x} \cdot \mathbf{c}) \in \mathbf{CRE}_\Sigma}$$

$$\frac{\mathbf{x} \in \mathbf{CRE}_\Sigma \quad \mathbf{c} \in \mathbf{CRE}_\Sigma}{(\mathbf{c} \cdot \mathbf{x}) \in \mathbf{CRE}_\Sigma}$$

$[\cdot]$ est appelé le *trou* d'un contexte. Si $\mathbf{c} \in \mathbf{CRE}_\Sigma$ et $\mathbf{x} \in \mathbf{RE}_\Sigma$, alors $\mathbf{c}[\mathbf{x}]$ denote l'expression régulière que l'on obtient comme résultat de l'*application* de \mathbf{c} à \mathbf{x} , par laquelle le trou est remplacée par l'expression \mathbf{x} .

Congruences

2.4.9 Définition (Congruence) Soit R une équivalence sur \mathbf{RE}_Σ . R est une *congruence* si elle est préservée par tous les contextes $c \in \mathbf{CRE}_\Sigma$, donc, si elle satisfait la règle :

$$\frac{x R y \quad c \in \mathbf{CRE}_\Sigma}{c[x] R c[y]}$$

2.4.10 Lemme La relation \approx est une congruence.

C'est seulement par le Lemme 2.4.10, que nous avons le droit de remplacer un sous-terme x d'une expression régulière $c[x]$ par un autre terme x' égal à x étant sur que le langage du terme ne change pas.

Langages des états d'un AFD

2.4.12 Définition Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD.
Pour tout $q \in Q$, on définit le langage L_q par :

$$L_q \triangleq L((Q, \Sigma, \delta, q, F))$$

$$\text{Cas spécifique:} \\ L(M) = L_s$$

2.4.13 Lemme Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD. Alors, pour tout $q \in Q$:

$$L_q = \left(\bigcup_{a \in \Sigma} \{a\} \cdot L_{\delta(q,a)} \right) \cup \{ \epsilon \mid q \in F \} \quad (\text{LIN})$$

2.4.14 Lemme (Arden) Soit Σ un alphabet. Soit $X, A, B \subseteq \Sigma^*$ des langages sur Σ tels que $\epsilon \notin A$ et satisfaisant l'équation :

$$X = AX + B$$

Alors $X = A^*B$

Conversion AFD \geq RE

2.4.15 Définition (Conversion de AFD en RE) Soit $(Q, \Sigma, \delta, s, F)$.

1. Établir un système de $\#(Q)$ équations linéaires (du format (LIN)) qui définissent les « variables » L_q correspondants aux états $q \in Q$.
2. Par abus de notation, nous écrivons a à la place de $\{a\}$ et $+$ à la place de \cup ; ceci est justifié par Définition 2.4.3.
3. Résoudre ce système de manière récursive par élimination des variables (sauf L_s) l'une après l'autre en utilisant le Lemme 2.4.14 d'Arden et la Proposition 2.4.11.
4. On résout ce système afin de trouver une expression régulière pour L_s .

2.4.16 Théorème Soit M un AFD et \mathbf{x} le résultat du système d'équations de la Définition 2.4.15 appliqué à M . Alors, $L(\mathbf{x}) = L(M)$.

AFNG

2.4.17 Définition (AFNG) Un *automate fini non déterministe généralisé* (AFNG) est un quintuplet

$$M = (Q, \Sigma, \Delta, s, f)$$

où

- Q est un ensemble fini d'états,
- Σ est un ensemble fini de symboles, l'alphabet (d'entrée),
- $\Delta : (Q \setminus \{f\}) \times (Q \setminus \{s\}) \rightarrow \mathbf{RE}_\Sigma$ est la fonction (totale) de transition,
- $s \in Q$ est l'état initial (ou de départ),
- $f \in Q$ est l'état accepteur (ou final).

Noter:

(presque) toute paire d'états est lié par une flèche
 s et f sont différents dans ce sens

Langage d'un AFNG

2.4.18 Définition Soit $M = (Q, \Sigma, \Delta, s, f)$ un AFNG.

1. M accepte un mot $w \in \Sigma^*$
s'il existe $w_1, \dots, w_k \in \Sigma^*$ et $q_0, \dots, q_k \in Q$ tels que
 - (a) $w = w_1 \cdot \dots \cdot w_k$
 - (b) $q_0 = s$ et $q_k = f$
 - (c) $\forall i \in [1, k] : w_i \in L(\Delta(q_{i-1}, q_i))$
2. Le langage accepté par M est défini par :

$$L(M) \triangleq \{ w \in \Sigma^* \mid w \text{ accepté par } M \}$$

Conversion AFD > AFNG

2.4.19 Définition

$G : \text{AFD} \rightarrow \text{AFNG}$

$$(Q, \Sigma, \delta, s, F) \mapsto (Q_G, \Sigma, \Delta, s_G, f_G)$$

avec $Q_G \triangleq Q \cup \{s_G, f_G\}$ où $Q \cap \{s_G, f_G\} = \emptyset$ et Δ telle que

$$\frac{}{\Delta(s_G, s) = \epsilon} \qquad \frac{q \in Q \setminus \{s\} \cup \{f_G\}}{\Delta(s_G, q) = \emptyset}$$

$$\frac{q \in F}{\Delta(q, f_G) = \epsilon}$$

$$\frac{q \in Q \setminus F}{\Delta(q, f_G) = \emptyset}$$

$$\frac{q, q' \in Q \quad \{a \mid \delta(q, a) = q'\} = \{a_1 \dots, a_n\} \neq \emptyset}{\Delta(q, q') = \mathbf{a}_1 + \dots + \mathbf{a}_n}$$

$$\frac{q, q' \in Q \quad \{a \mid \delta(q, a) = q'\} = \emptyset}{\Delta(q, q') = \emptyset}$$

Élimination des états d'un AFNG

2.4.20 Théorème Soit M un AFD. Alors $L(M) = L(G(M))$

2.4.21 Définition (Réduction de AFNG)

Soit $M = (Q, \Sigma, \Delta, s, f)$ un AFNG et $\hat{q} \in Q \setminus \{s, f\}$
La réduction $R(M, \hat{q})$ de M par \hat{q} est définie par

$$R(M, \hat{q}) \triangleq (Q \setminus \{\hat{q}\}, \Sigma, \Delta', s, f)$$

avec Δ' étant définie par la règle :

$$\frac{q \in Q \setminus \{f\} \quad q' \in Q \setminus \{s\}}{\Delta'(q, q') = \Delta(q, q') + \Delta(q, \hat{q}) \cdot (\Delta(\hat{q}, \hat{q}))^* \cdot \Delta(\hat{q}, q')}$$

2.4.22 Lemme Soit M un AFNG. Alors $L(M) = L(R(M))$

Conversion AFD > RE

2.4.23 Définition (Algorithme de conversion de AFD en RE)

```
reduce( (Q, Σ, Δ, s, f) ) :=  
  si #(Q) > 2  
    choisir q ∈ Q \ {s, f}  
    retourner reduce( R((Q, Σ, Δ, s, f) q) )  
  sinon  
    retourner (Q, Σ, Δ, s, f)
```

```
afd2re(Q, Σ, δ, s, F) :=  
  calculer (Q', Σ, Δ, s', f') = reduce( G((Q, Σ, δ, s, F)) )  
  retourner Δ(s', f')
```

2.4.24 Théorème (Correction de l'algorithme de conversion) Soit M un AFD et x le résultat de l'exécution de $afd2re(M)$. Alors $L(M) = L(x)$.