Théorie des Langages

CS410 - Langages et Compilation

Julien Henry
Catherine Oriat

Grenoble-INP Esisar

2012-2013

Introduction

Objectif : Étudier les notions de TL utiles en compilation :

- Automates et expressions régulières : lexicographie des langages de programmation
- Grammaires Hors Contexte : syntaxe hors contexte des langages de programmation
- Grammaires attribuées : règles contextuelles des langages de programmation

Summary

Définitions de base

Vocabulaire : ensemble fini de caractères ou symboles.
Mot (ou **chaîne**) sur un vocabulaire V : suite finie d'éléments de V.
Par convention, ε est un mot de longueur 0.
Préfixe : la chaîne x est préfixe de y si et seulement si il existe une chaîne z telle que $x.z = y$, où . est l'opérateur de concaténation.
Suffixe : la chaîne x est suffixe de y si et seulement si il existe une chaîne z telle que $z.x = y$.
Sous-chaîne : la chaîne x est sous-chaîne de y si et seulement si il existe des chaînes z et $z'/z.x.z' = y$.
Langage : On appelle langage sur un vocabulaire V un sous ensemble L de V^*.
Exemples :

- l'ensemble des mots de la langue française est un langage sur le vocabulaire constitué des lettres de l'alphabet.
- l'ensemble des phrases de la langue française est un langage sur le vocabulaire constitué des mots de la langue française.

Opérations sur les langages

Union : Soit deux langages L_1 et L_2 sur un vocabulaire V. L'union L de ces langages est :

$$L = L_1 + L_2 = L_1 \bigcup L_2 = \{w \in V^*; w \in L_1 \text{ ou } w \in L_2\}$$

Intersection : Soit deux langages L_1 et L_2 sur un vocabulaire V. L'intersection L de ces langages est :

$$L = L_1 \bigcap L_2 = \{w \in V^*; w \in L_1 \text{ et } w \in L_2\}$$

Concaténation : Soit deux langages L_1 et L_2 sur un vocabulaire V. La concaténation L de ces langages est :

$$L = L_1.L_2 = \{w \in V^*; \exists w_1 \in L_1, w_2 \in L_2, w = w_1.w_2\}$$

Opérations sur les langages

Puissance : Soit un langage L sur le vocabulaire V. L'élévation de L à la puissance n est définie par récurrence sur n :

$$\begin{aligned} L^0 &= \{\varepsilon\} \\ L^1 &= L \\ L^{n+1} &= L.L^n \end{aligned}$$

Concaténations itérées : Soit un langage L sur le vocabulaire V. Les langages L^* et L^+ sont définis par :

$$\begin{aligned} L^* &= \bigcup_{n \geq 0} L^n \\ L^+ &= \bigcup_{n \geq 1} L^n \end{aligned}$$

Summary

Définition

Définition: (Expression Régulière)

Soit un vocabulaire V. L'ensemble des expressions régulières sur V est defini par :

- *$\emptyset$ est une expression régulière ;*
- *pour tout élément a de V, a est une expression régulière ;*
- *si x et y sont des expressions régulières, alors $x + y$ est une expression régulière ;*
- *si x est une expression régulière, alors x^* est une expression régulière ;*

Les expressions peuvent être parenthésées :

- * est prioritaire sur .
- . est prioritaire sur $^+$

Langage défini par une expression régulière

Une expression régulière définit un langage :

- $\emptyset$ définit le langage $\emptyset$;
- l'expression a définit le langage $\{a\}$;
- si e_1 et e_2 sont des expressions régulières, définissant les langages L_1 et L_2, alors l'expression $e_1 + e_2$ définit $L_1 + L_2$;
- si e_1 et e_2 sont des expressions régulières, définissant les langages L_1 et L_2, alors l'expression $e_1.e_2$ définit $L_1.L_2$;
- si e est une expression régulière, définissant le langage L, alors l'expression e^* définit L^* ;

Langage Régulier

Définition: (Langage régulier)

Un langage régulier est un langage qui peut être défini par une expression régulière.

Exemples :

- Le langage $\{ab^n, n \in \mathbb{N}\}$ peut être défini par l'expression régulière ab^*.
- L'expression régulière $0 + 1(0 + 1)^*$ définit le langage des constantes binaires sans 0 superflu en tête.

Summary

Automates Finis

Définition: (Automate Fini)

Un automate fini est un quintuplet $A =< Q, V, I, F, \delta >$, où :

- *Q est un ensemble fini d'états ;*
- *V est un vocabulaire ;*
- *$I \subseteq Q$ est l'ensemble des états initiaux ;*
- *$F \subseteq Q$ est l'ensemble des états terminaux (ou finaux) ;*
- *$\delta \subseteq Q \times (V \cup \{\varepsilon\}) \times Q$ est la relation de transitions.*

Définitions

Chemin, Trace : un chemin de l'automate A est une suite de transitions $(r_0, a_1, r_1), (r_1, a_2, r_2), \cdots, (r_{n-1}, a_n, r_n)$, avec $\forall 1 \leq i \leq n, (r_{i-1}, a_i, r_i) \in \delta$. Ce chemin mène de r_0 à r_n avec la **trace** $a_1 a_2 \cdots a_n$.
Mot reconnu par un automate : un mot $x \in V^*$ est reconnu par un automate si il existe un chemin partant d'un état initial et arrivant à un état terminal de trace x.
Langage reconnu par un automate : l'ensemble des mots reconnus par un automate forme le langage reconnu par l'automate.
Équivalence entre deux automates : deux automates sont équivalents s'ils ont le même vocabulaire et reconnaissent le même langage.

Automate sans ε-transition

Une ε**-transition** est une transition de la forme (p, ε, q).

Théorème:
Si L est un langage sur V reconnu par un automate fini, alors il est reconnu par un automate fini sans ε-transition.

Automate déterministe

Définition: (Automate déterministe)
Un automate $A =< Q, V, I, F, \delta >$ est déterministe si et seulement si :

- *I contient un seul élément ;*
- *δ n'a pas d'ε-transition ;*
- *$\forall p \in Q, \forall a \in V$, il existe un unique état $q \in Q$ tel que $(p, a, q) \in \delta$. Cet unique état est noté $\delta(p, a)$.*

La relation de transition δ est donc une application :

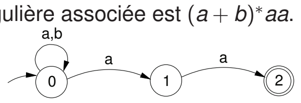
$$\begin{aligned} \delta : \quad Q \times V &\rightarrow Q \\ (p, a) &\mapsto \delta(p, a) \end{aligned}$$

Langage reconnu par un automate déterministe

Théorème:
Si L est un langage sur V reconnu par un automate fini, alors il est reconnu par un automate fini déterministe.

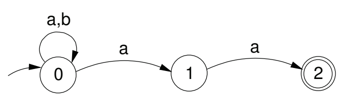
Méthode de déterminisation d'un automate

On part d'un automate non déterministe sans ε-transitions.
Exemple :
On considère le langage sur $V = \{a, b\}$ des chaînes qui se terminent par aa : l'expression régulière associée est $(a + b)^* aa$. Il est reconnu par l'automate suivant :



Méthode de déterminisation d'un automate

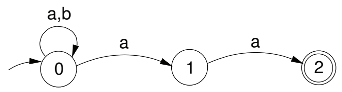
On déterminise l'automate en construisant progressivement un tableau :



Etat	a	b	Final
0	0,1	0	

Méthode de déterminisation d'un automate

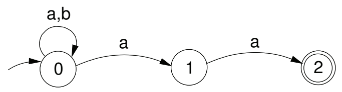
On déterminise l'automate en construisant progressivement un tableau :



Etat	*a*	*b*	Final
0	0,1	0	
0,1	0,1,2	0	

Méthode de déterminisation d'un automate

On déterminise l'automate en construisant progressivement un tableau :

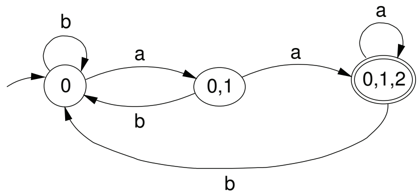


Etat	*a*	*b*	Final
0	0,1	0	
0,1	0,1,2	0	
0,1,2	0,1,2	0	F

Méthode de déterminisation d'un automate

Etat	*a*	*b*	Final
0	0,1	0	
0,1	0,1,2	0	
0,1,2	0,1,2	0	F

On peut donc construire l'automate :



Automate minimal

Définition: (Automate déterministe minimal)

Un automate déterministe est minimal si et seulement si il n'existe pas d'automate déterministe équivalent comportant un nombre strictement inférieur d'état.

Pour construire un automate déterministe minimal, on a besoin de la notion de classe d'équivalence.

Automate minimal

Soit un automate déterministe $A = (Q, V, q_0, F, \delta)$ sans état inaccessible. On définit une suite de relations d'équivalence $(\equiv_k)_{k \in \mathbb{N}}$ sur Q :

- $p \equiv_0 q$ si et seulement si $p \in F \Leftrightarrow q \in F$.
- $\forall k \in \mathbb{N}$, $p \equiv_{k+1} q$ si et seulement si $p \equiv_k q$ et $\forall a \in V, \delta(p, a) \equiv_k \delta(q, a)$.

Cette suite de relations d'équivalence converge vers la relation d'équivalence notée $\equiv$.
Chaque état p fait donc partie d'une classe d'équivalence notée $[p]$:

$$[p] = \{q \in Q; p \equiv q\}$$

Automate minimal

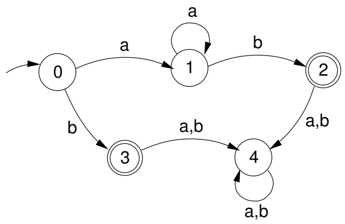
Théorème:

Soit A un automate déterministe. Un automate déterministe minimal équivalent à A, noté $\mu(A)$, est défini de la façon suivante :
$\mu(A) = (R, V, I, G, \eta)$, avec :

- $R = \{[p]; p \in Q\}$ *: ensemble des classes d'équivalence*
- $I = [i]$ *où i est l'état initial de A*
- $G = \{[q]; q \in F\}$
- $\eta([p], a) = [\delta(p, a)]$

Méthode de minimisation d'un automate

Soit l'automate déterministe suivant :



On cherche les classes d'équivalence en construisant un tableau :

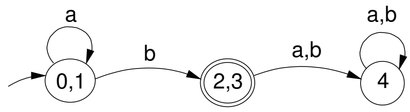
$\equiv_0$	0,1,4		2,3
$\equiv_1$	0,1	4	2,3
$\equiv_2$	0,1	4	2,3

On s'arrête lorsqu'on a convergé.

Méthode de minimisation d'un automate

$\equiv_0$	0,1,4		2,3
$\equiv_1$	0,1	4	2,3
$\equiv_2$	0,1	4	2,3

On obtient donc l'automate minimal suivant :

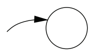


Automate et expression régulière

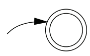
Théorème:
Les expressions régulières et les automates finis définissent la même classe de langage.

Automate associé à une expression régulière

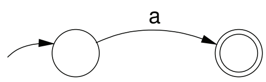
- $e = \emptyset$:



- $e = \varepsilon$:

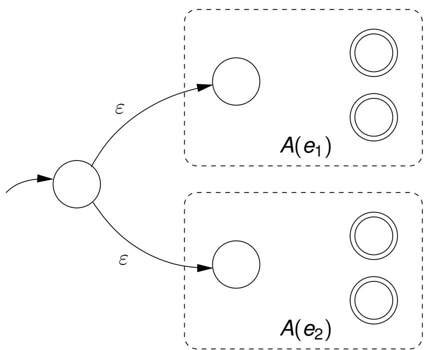


- $e = a \in V$:



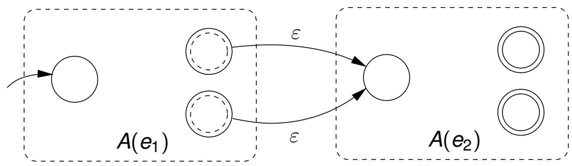
Automate associé à une expression régulière

- $e = e_1 + e_2$:



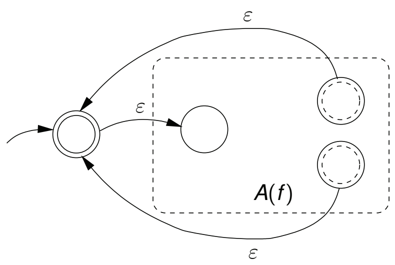
Automate associé à une expression régulière

- $e = e_1.e_2$:



Automate associé à une expression régulière

- $e = e_1^*$:



Equation sur un langage

Définition:

Soit α et β deux langages sur un vocabulaire V. Soit L un langage sur V. On dit que $x = L$ est solution de l'équation $x = \alpha x + \beta$ si et seulement si L vérifie $L = \alpha L + b$

Lemme: (Lemme d'Arden)

- *L'équation $x = \alpha x + \beta$ admet comme solution le langage $\alpha^*\beta$. Cette solution est la plus petite solution, ce qui signifie que si $x = L$ est solution, alors $\alpha^*\beta \subseteq L$.*
- *L'équation $x = x\alpha + \beta$ admet comme plus petite solution le langage $\beta\alpha^*$.*

Exemples

Soit $V = \{a, b\}$.

- $x = ax + b$ a pour solution a^*b.
- $x = ax + \varepsilon$ a pour solution a^*.
- $x = x + b$ a pour solution b.
- $x = ax$ a pour solution $\emptyset$.
- $x = xa + b$ a pour solution ba^*.

Expression régulière associée à un automate

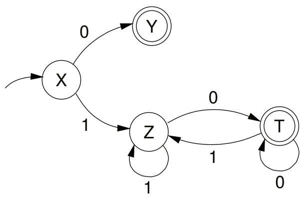
On considère un automate $A = (Q, V, I, F, \delta)$. On cherche l'expression régulière associée à A.

Technique :

- A chaque état q_i de l'automate, on associe une variable X_i, qui est le langage reconnu par l'automate lorsqu'on part de l'état q_i.
- Pour chaque variable X_i, on a l'équation :
  - $X_i = \sum_{(q_i, a_k, q_j) \in \delta} a_k X_j$ si q_i n'est pas terminal.
  - $X_i = \sum_{(q_i, a_k, q_j) \in \delta} a_k X_j + \varepsilon$ si q_i est terminal.
- On obtient un système d'équations dont on cherche la plus petite solution. L'expression régulière recherchée est alors $e = \sum_{q_i \in I} X_i$.

Expression régulière associée à un automate

Exemple : On considère l'automate suivant :



$$\begin{cases} X &= 0Y + 1Z \\ Y &= \varepsilon \\ Z &= 0T + 1Z \\ T &= 0T + 1Z + \varepsilon \end{cases}$$

Expression régulière associée à un automate

$$\begin{cases} X &= 0Y + 1Z \\ Y &= \varepsilon \\ Z &= 0T + 1Z \\ T &= 0T + 1Z + \varepsilon \end{cases} \Leftrightarrow \begin{cases} X &= 0 + 1Z \\ Y &= \varepsilon \\ Z &= 0T + 1Z \\ T &= Z + \varepsilon \end{cases}$$

$$\Leftrightarrow \begin{cases} X &= 0 + 1Z \\ Y &= \varepsilon \\ Z &= 0(Z + \varepsilon) + 1Z = (0 + 1)Z + 0 \\ T &= Z + \varepsilon \end{cases}$$

$$\Leftrightarrow \begin{cases} X &= 0 + 1(0 + 1)^*0 \\ Y &= \varepsilon \\ Z &= (0 + 1)^*0 \\ T &= Z + \varepsilon \end{cases}$$