# Théorie des Langages - EISTI - ING 1

Yannick Le Nir

Ecole Internationale des Sciences du Traitement de l'Information

yannick.lenir@eisti.fr

# Rappels

## Langages, grammaires et automates

- Langages algébriques (hors contexte)
  - grammaire sous forme normale de Chomsky
  - algorithme d'analyse CKY ($n^3$)
  - automates à pile
- Langages rationnels (réguliers)
  - grammaire de type 3
  - expressions régulières
  - automates états finis

# Langages et automates

### Théorème de Kleene

Un langage est régulier ssi il est reconnu par un automate d'états finis

### Objectifs

- Construction de l'automate minimal à partir d'un langage rationnel : méthode des quotients gauches
- Génération du langage à partir de l'automate : mise en équation et lemme d'Arden

# Expressions rationnelles ou régulières



## Construction

Soit $X$ un alphabet :

- $\forall x \in X$, $x$ est une expression régulière (ER)
- si $E$ est une ER, $(E)$ est une ER
- si $E_1$ et $E_2$ sont des ER, $(E_1 \ + \ E_2)$ est une ER
- si $E_1$ et $E_2$ sont des ER, $(E_1.E_2)$ est une ER (le point peut être ommis)
- si $E$ est un ER, $E^*$ est une ER

## Exemple

- $(0 \ + \ 1)^*$
- $((0(10)^*)1)$
- $((10)^*(01)^*)$

# Langages rationnels (réguliers)

## Propriété fondamentale

Un langages $L$ est rationnel (régulier) si et seulement si il existe une expression rationnelle (régulière) $E$ telle que $L = \mu(E)$.

## Remarque

On confondra souvent par abus de langage, une expression et le langage qu'elle représente

# Langages et expressions

## Equivalence

- Un langage rationnel peut être reconnu par des automates différents (cf TD 3)
- Un langage rationnel peut également être représenté par différentes expressions (dites équivalentes)

## Exemple

- Les expressions $(a^*b^*)^*$ et $(a + b)^*$ sont équivalentes et représentent toutes les deux le langage $\{a, b\}^*$
- Par contre $(aa + b)^*$ et $(aa + aab + bb)^*$ ne sont pas équivalentes car par exemple $b$ appartient au langage associé à la première mais pas à celui de la seconde.

## De l'automate au langage

- Déterminer le système d'équations à partir de l'automate
- Déterminer sa solution : le langage reconnu par l'automate
- $\rightarrow$ résolution via lemme d'Arden

# Système d'équations

## Construction d'une ER à partir d'un AFD

- Chaque état devient une inconnue (l'ele langage reconnu par cet état de l'AFD).
- Les évènements sont les constantes.
- Toutes les transitions $(q_0, e_i, q_i)$ produisent l'équation $q_0 = \sum e_i.q_i (+\varepsilon$si $q_0$ est un état final)
- La solution est la valeur de l'inconnue correspondant à l'état initial (langage reconnu par l'AFD).

# Lemme d'Arden

## Lemme

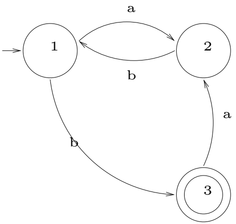
Soit $E$ une équation du type $X = A.X + D$,
avec $A$ et $D$ des langages réguliers.
Alors, une solution de $E$ est le langage $X = A^*D$ si $\varepsilon \notin A$ et le langage $X = A^+D$ sinon.

# Résolution de l'équation

## Principe

- Générer le système d'équations
- Appliquer plusieurs fois le lemme d'Arden pour résoudre le système d'équations
- La solution est le langage reconnu par l'AFD

## Exemple



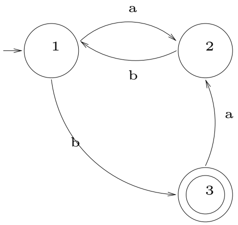

$$Y_1 = aY_2 + bY_3$$

# Résolution de l'équation

## Principe

- Générer le système d'équations
- Appliquer plusieurs fois le lemme d'Arden pour résoudre le système d'équations
- La solution est le langage reconnu par l'AFD

## Exemple



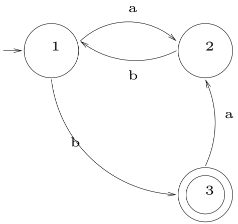

$$
\begin{aligned}
Y_1 &= aY_2 + bY_3 \\
Y_2 &= bY_1
\end{aligned}
$$

# Résolution de l'équation

## Principe

- Générer le système d'équations
- Appliquer plusieurs fois le lemme d'Arden pour résoudre le système d'équations
- La solution est le langage reconnu par l'AFD

## Exemple



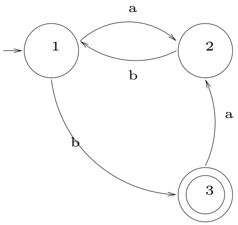

$$\begin{aligned} Y_1 &= aY_2 + bY_3 \\ Y_2 &= bY_1 \\ Y_3 &= aY_2 + \varepsilon \end{aligned}$$

# Résolution de l'équation

## Principe

- Générer le système d'équations
- Appliquer plusieurs fois le lemme d'Arden pour résoudre le système d'équations
- La solution est le langage reconnu par l'AFD

## Exemple




$$
\begin{aligned}
Y_1 &= aY_2 + bY_3 \\
Y_2 &= bY_1 \\
Y_3 &= aY_2 + \varepsilon
\end{aligned}
$$

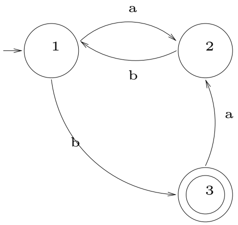
$Y_1 =$
$aY_2 + b(aY_2 + \varepsilon)$

# Résolution de l'équation

## Principe

- Générer le système d'équations
- Appliquer plusieurs fois le lemme d'Arden pour résoudre le système d'équations
- La solution est le langage reconnu par l'AFD

## Exemple




$$
\begin{aligned}
Y_1 &= aY_2 + bY_3 \\
Y_2 &= bY_1 \\
Y_3 &= aY_2 + \varepsilon
\end{aligned}
$$

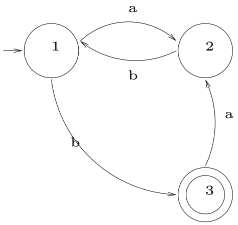
$$
Y_1 = aY_2 + b(aY_2 + \varepsilon) = (a + ba)Y_2 + b
$$

# Résolution de l'équation

## Principe

- Générer le système d'équations
- Appliquer plusieurs fois le lemme d'Arden pour résoudre le système d'équations
- La solution est le langage reconnu par l'AFD

## Exemple




$$
\begin{aligned}
Y_1 &= aY_2 + bY_3 \\
Y_2 &= bY_1 \\
Y_3 &= aY_2 + \varepsilon
\end{aligned}
$$

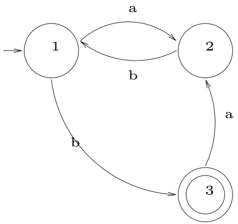
$$
\begin{aligned}
Y_1 &= \\
&aY_2 + b(aY_2 + \varepsilon) \\
&= (a + ba)Y_2 + b \\
&= (a+ba)bY_1 + b
\end{aligned}
$$

# Résolution de l'équation

## Principe

- Générer le système d'équations
- Appliquer plusieurs fois le lemme d'Arden pour résoudre le système d'équations
- La solution est le langage reconnu par l'AFD

## Exemple




$$\begin{aligned} Y_1 &= aY_2 + bY_3 \\ Y_2 &= bY_1 \\ Y_3 &= aY_2 + \varepsilon \end{aligned}$$

$$\begin{aligned} Y_1 &= aY_2 + b(aY_2 + \varepsilon) \\ &= (a+ba)Y_2 + b \\ &= (a+ba)bY_1 + b \\ &=_{arden} ((a+ba)b)^* b \end{aligned}$$

## Du langage à l'automate minimal

- Déterminer les états à partir du langage
- Déterminer les transitions
- $\longrightarrow$ construction via quotients gauches

# Quotients gauches



## Définition

Soit $L = L_1.L_2$ un langage donné sur un alphabet $\{a, b\}$.
Les quotients gauches de $L$ sont : $a^{-1}L$ et $b^{-1}L$, avec :

$$\begin{cases} u^{-1}L_1.L_2 = (u^{-1}L_1)L_2 + u^{-1}L2 & \text{si } \varepsilon \in L_1 \\ u^{-1}L_1.L_2 = (u^{-1}L_1)L_2 & \text{sinon} \\ u^{-1}(L_1 + L_2) = u^{-1}L_1 + u^{-1}L_2 & \end{cases}$$

## Théorème

L'ensemble des quotients gauches de $L$, $Q(L)$, est fini

## Proposition

Soit un AFD, $A$, d'états $E$, $Q(L) = \{L_q(A), q \in Q\}$. On peut donc construire un automate minimal qui reconnait $L$ et dont chaque état correspond à un élément de $R(L)$

# Construction de l'automate

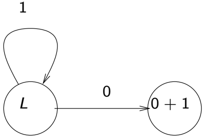


## Principe

- Chaque quotient est un état de l'automate
- Chaque transition permet de passer d'un état à l'autre
- L'état final est le quotient chaîne vide $\varepsilon$
- L'état poubelle est le quotient vide

## Exemple

Soit l'E.R. $L = 1^*0(0+1)$
$0^{-1}L = 0+1$ et $1^{-1}L = L$



# Construction de l'automate

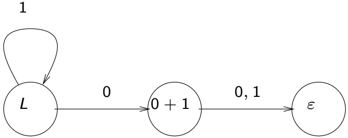
## Principe

- Chaque quotient est un état de l'automate
- Chaque transition permet de passer d'un état à l'autre
- L'état final est le quotient chaîne vide $\varepsilon$
- L'état poubelle est le quotient vide

## Exemple

Soit l'E.R. $L = 1^*0(0+1)$

$0^{-1}(0+1) = \varepsilon$ et $1^{-1}(0+1) = \varepsilon$



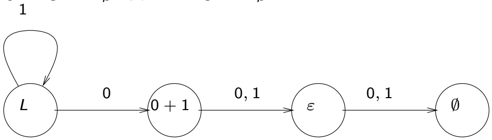
# Construction de l'automate

## Principe

- Chaque quotient est un état de l'automate
- Chaque transition permet de passer d'un état à l'autre
- L'état final est le quotient chaîne vide $\varepsilon$
- L'état poubelle est le quotient vide

## Exemple

Soit l'E.R. $L = 1^*0(0+1)$
$0^{-1}\varepsilon = \emptyset$ et $1^{-1}\varepsilon = \emptyset$



# Construction de l'automate

## Principe

- Chaque quotient est un état de l'automate
- Chaque transition permet de passer d'un état à l'autre
- L'état final est le quotient chaîne vide $\varepsilon$
- L'état poubelle est le quotient vide

## Exemple

Soit l'E.R. $L = 1^*0(0+1)$
$0^{-1}\emptyset = \emptyset$ et $1^{-1}\emptyset = \emptyset$