Universidad Carlos III de Madrid
www.uc3m.es

AUTOMATA THEORY AND FORMAL LANGUAGES

UNIT 5: REGULAR LANGUAGES

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

OUTLINE

PART 1:

- Finite Automata and Type-3 Grammars
  - Finite Automata associated to a Type-3 grammar (G3→FA)
  - Type-3 Grammar associated to a FA (FA→G3)

PART 2:

- Regular expressions and Regular Languages

OUTLINE

PART 1:

- **Finite Automata and Type-3 Grammars**
  - **Finite Automata associated to a Type-3 grammar (G3→FA)**
  - **Type-3 Grammar associated to a FA (FA→G3)**

PART 2:

- Regular expressions and Regular Languages

From FA to Type-3 grammar

1 From FA $\rightarrow$ G3:

Given the FA, A = (Σ, Q, qo, f, F), there is a right-linear grammar that fulfills

L(G3RL) = L(A)

That it is to say, the language generated by the grammar is the same that the recognized by the automaton

Following: How to obtain the grammar G={Σ_T, Σ_N, S, P}

from the FA= {Q, Σ, q_0, f, F}

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

From FA to Type-3 grammar

1 From FA $\rightarrow$ G3:

<u>Process</u>:

- $\Sigma_T = \Sigma$; $\Sigma_N = Q$, $S = qo$
- P= {…}

1. Transition $f(p,a) = q \rightarrow$ if q' is not a final state $\rightarrow$ p::= aq
2. $q \in F$ and $f(p,a) = q$ $\rightarrow$ p::= a and p::= aq
3. $p0 \in F \rightarrow p0 ::= \lambda$
4. If $f(p, \lambda) = q \rightarrow p ::= q$
5. $q \in F$ and $f(p, \lambda) = q$ $\rightarrow$ p ::= q and $q ::= \lambda$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

From FA to Type-3 grammar

2 From G3 $\rightarrow$ FA:

Given a right-linear G3, G = (Σ_T, Σ_N S, P), there is a FA, A, that fulfills: L(G3LD) = L(A)

Process:

- $\Sigma = \Sigma_T$
- $Q = \Sigma_N \cup \{F\}$, with $F \notin \Sigma_N$
- qo = S
- F = {F}
- f:
  - If A ::= aB $\rightarrow$ f(A,a) = B
  - If A::= a $\rightarrow$ f(A,a) = F
  - If S ::= λ $\rightarrow$ f(S, λ) = F

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

From Type-3 grammar to FA

- We have seen the procedure to obtain a FA that accepts the language described by a G3 left-linear grammar, however, this procedure does not always lead to an DFA, typically:

$$G3 \rightarrow NFA \rightarrow DFA$$

- Exercise 1: Given the left-linear grammar: G= ({0,1}, {S,U}, S,{S ::= U0, U ::= U0 | S1 | 0}) Calculate the corresponding DFA.

- Exercise 2: Given the left-linear grammar: G= ({0,1}, {S,U}, S,{S ::= U0 / λ, U ::= U0 | S1 | 0}) Calculate the corresponding DFA.

From Type-3 grammar to FA

Given the left-regular grammar G3: $G = (\Sigma_T, \Sigma_N, S, P)$

From it, we build the FA: $A= (\Sigma_T, \underbrace{\Sigma_N \cup \{p,q\}}_{Q},f, p,\{S\})$

where: $p, q \notin \Sigma_T$ and/or Σ_N

f is defined by:

1) $f(U,t) = V$ si $V::= U\ t \in P$
2) $f(p,t) = V$ si $V::= t \in P$
3) $f(U,t) = q \quad \forall t \in \Sigma_T \ / \ V::= U\ t \notin P$
4) $f(p,t) = q \quad \forall t \in \Sigma_T \ / \ V::= t \notin P$
5) $f(q,t) = q \quad \forall t \in \Sigma_T$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

From Type-3 grammar to FA

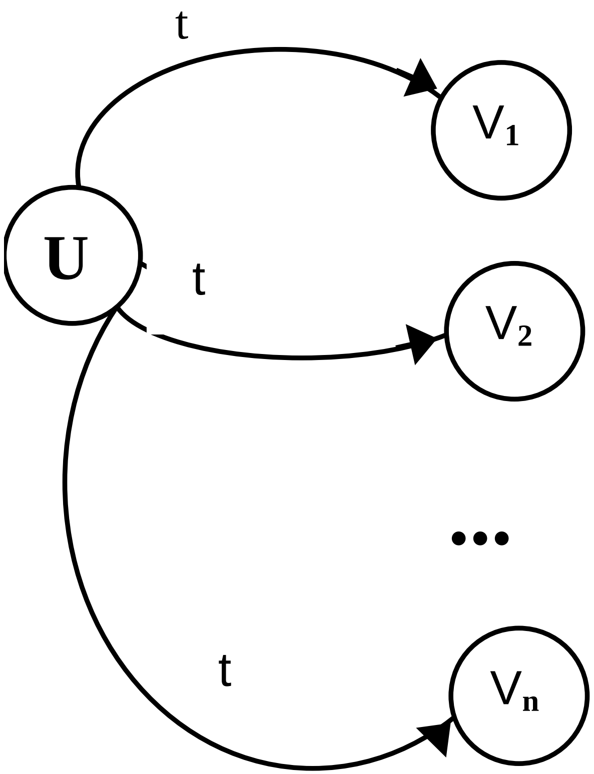
- This definition does not ensure a deterministic FA since it is possible:

  V1 ::= Ut

  V2 ::= Ut

  ...

  V3 ::= Ut



Additional Issues

And if we want to obtain a FA from a left-linear G3?

G3 left-linear → G3 right-linear → FA

And if we want to obtain a left-linear G3 from a FA?

FA → G3 right-linear → G3 left-linear

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

OUTLINE

PART 1:

- Finite Automata and Type-3 Grammars
  - Finite Automata associated to a Type-3 grammar (G3→FA)
  - Type-3 Grammar associated to a FA (FA→G3)

PART 2:

- **Regular expressions and Regular Languages**

OUTLINE

Unit 5. Part 2: Regular Expressions

OUTLINE

Unit 5. Part 2: Regular Expressions

- **Definition of a Regular Expression (RE)**
- Regular Expressions and Regular Languages
- Equivalence of Regular Expressions
- Analysis Theorem and Kleene's Synthesis Theorem
  - Solution of the Analysis Problem. Characteristic Equations
    - Solution of the Characteristic Equations
    - Algorithm to Solve the Analysis Problem
  - Synthesis Problem: Recursive Algorithm
    - Synthesis Problem: Derivatives of Regular Expressions

Definition of Regular Expression

Kleene, 1956:

“Metalanguage for expressing the set of words accepted by a FA (i.e. to express Type-3 or regular languages)"

Example: given the alphabet Σ= {0,1}

0*10* is a word of the metalanguage representing the infinite words which consist of a 1, preceded and followed by none, one or infinite zeros.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Definition of Regular Expression

- Regular expressions: rules that define exactly the set of words that are included in the language.
- Main operators:
  - **Concatenation:** xy
  - **Alternation:** x|y (x or y)
  - **Repetition:** x^* (x repeated 0 or more times)
    x^+ (x repeated 1 or more times)

Definition of Regular Expression

- Given an alphabet Σ, the rules that define regular expressions of Σ are:
  - $\forall a \in \Sigma$ is a regular expression.
  - λ is a regular expression.
  - Φ is a regular expression.
  - If **r** and **s** are regular expressions, then

    (r) r·s r|s r*

    are regular expressions.

$$r^* = \bigcup_{i=0}^{\infty} r^i$$

- Nothing else is a regular expression.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Definition of Regular Expression

- Valid RE are those obtained after applying the previous rules a finite number of times over symbols of Σ, Φ, λ

- The priority of the different operations is the following:
  - *, • ,+

OUTLINE

Unit 5. Part 2: Regular Expressions

- Definition of a Regular Expression (RE)
- **Regular Expressions and Regular Languages**
- Equivalence of Regular Expressions
- Analysis Theorem and Kleene's Synthesis Theorem
  - Solution of the Analysis Problem. Characteristic Equations
    - Solution of the Characteristic Equations
    - Algorithm to Solve the Analysis Problem
  - Synthesis Problem: Recursive Algorithm
    - Synthesis Problem: Derivatives of Regular Expressions

Regular expressions and Regular Languages

Each RE describes a regular language

- Each RE α has a set of Σ^* associated, $L(\alpha)$, that is the RL described by α. This language is defined by:

  - If $\alpha = \Phi$, $L(\alpha) = \Phi$
  - If $\alpha = \lambda$, $L(\alpha) = \{\lambda\}$
  - If $\alpha = a$, $a \in \Sigma$, $L(\alpha) = \{a\}$
  - If α and β are RE $\Rightarrow L(\alpha \mid \beta) = L(\alpha) \cup L(\beta)$
  - If α and β are RE $\Rightarrow L(\alpha \bullet \beta) = L(\alpha)\, L(\beta)$
  - If α^* is a RE $\Rightarrow L(\alpha^*) = L(\alpha)^*$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

OUTLINE

Unit 5. Part 2: Regular Expressions

- Definition of a Regular Expression (RE)
- Regular Expressions and Regular Languages
- **Equivalence of Regular Expressions**
- Analysis Theorem and Kleene's Synthesis Theorem
  - Solution of the Analysis Problem. Characteristic Equations
    - Solution of the Characteristic Equations
    - Algorithm to Solve the Analysis Problem
  - Synthesis Problem: Recursive Algorithm
    - Synthesis Problem: Derivatives of Regular Expressions

Equivalence of Regular Expressions

- Two RE are equivalent, **$\alpha = \beta$**, if they describe the same regular language, $L(\alpha) = L(\beta)$. <u>Properties</u>:
  1) $(\alpha \mid \beta) \mid \sigma = \alpha \mid (\beta \mid \sigma)$ (| is associative)
  2) $\alpha \mid \beta = \beta \mid \alpha$ (| is commutative)
  3) $(\alpha \bullet \beta) \bullet \sigma = \alpha \bullet (\beta \bullet \sigma)$ (• is associative)
  4) $\alpha \bullet (\beta \mid \sigma) = (\alpha \bullet \beta) \mid (\alpha \bullet \sigma)$ (| is distributive
     $(\beta \mid \sigma) \bullet \alpha = (\beta \bullet \alpha) \mid (\sigma \bullet \alpha)$ regarding •)
  5) $\alpha \bullet \lambda = \lambda \bullet \alpha = \alpha$ (• has a neutral element)
  6) $\alpha \mid \Phi = \Phi \mid \alpha = \alpha$ (| has a neutral element)
  7) $\lambda^* = \lambda$
  8) $\alpha \bullet \Phi = \Phi \bullet \alpha = \Phi$

Equivalence of Regular Expressions

9) $\Phi^* = \lambda$
10) $\alpha^* \bullet \alpha^* = \alpha^*$
11) $\alpha \bullet \alpha^* = \alpha^* \bullet \alpha$
12) $(\alpha^*)^* = \alpha^*$ (IMPORTANT)
13) $\alpha^* = \lambda \mid \alpha \mid \alpha^2 \mid .. \mid \alpha^n \mid \alpha^{n+1}. \alpha^*$
14) $\alpha^* = \lambda \mid \alpha \bullet \alpha^*$ (13 with n=0) (IMPORTANT)
15) $\alpha^* = (\lambda \mid \alpha)^{n-1} \mid \alpha^n \bullet \alpha^*$ (from 14)
16) Given a function f, $f: E^n_\Sigma \rightarrow E_\Sigma$ then:
$f(\alpha, \beta, ..., \sigma) \mid (\alpha \mid \beta \mid ... \mid \sigma)^* = (\alpha \mid \beta \mid ... \mid \sigma)^*$
17) Given a function, $f: E^n_\Sigma \rightarrow E_\Sigma$ then:
$(f(\alpha^*, \beta^*, ..., \sigma^*))^* = (\alpha \mid \beta \mid ... \mid \sigma)^*$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Equivalence of Regular Expressions

18) $(\alpha^* \mid \beta^*)^* = (\alpha^* \bullet \beta^*)^* = (\alpha \mid \beta)^*$ (IMPORTANT)
19) $(\alpha \bullet \beta)^* \bullet \alpha = \alpha \bullet (\beta \bullet \alpha)^*$
20) $(\alpha^* \bullet \beta)^* \bullet \alpha^* = (\alpha \mid \beta)^*$
21) $(\alpha^* \bullet \beta)^* = \lambda \mid (\alpha \mid \beta)^* \bullet \beta$ (from 14 with 20)
22) Inference Rules:
given three regular expressions R,T and S:

$$R = S^* \bullet T \Rightarrow R = S \bullet R \mid T$$

If $\lambda \notin S$, then:

$$R = S \bullet R \mid T \Rightarrow R = S^* \bullet T$$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

OUTLINE

Unit 5. Part 2: Regular Expressions

- Definition of a Regular Expression (RE)
- Regular Expressions and Regular Languages
- Equivalence of Regular Expressions
- **Analysis Theorem and Kleene's Synthesis Theorem**
  - Solution of the Analysis Problem. Characteristic Equations
    - Solution of the Characteristic Equations
    - Algorithm to Solve the Analysis Problem
  - Synthesis Problem: Recursive Algorithm
    - Synthesis Problem: Derivatives of Regular Expressions

Analysis and Kleene's Synthesis Theorems

1) Analysis Theorem:

Every language accepted by a FA is a regular language.

Solution to the problem of analysis: To find the language associated to a specific FA: "Given a FA, A, find a RE that describes L(A)".

2) Synthesis Theorem:

Every regular language is a language accepted by a FA.

Solution to the problem of synthesis: To find a recognizer for a given regular language: **"Given a RE representing a regular language, build a FA that accepts that regular language".**

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

OUTLINE

Unit 5. Part 2: Regular Expressions

- Definition of a Regular Expression (RE)
- Regular Expressions and Regular Languages
- Equivalence of Regular Expressions
- Analysis Theorem and Kleene's Synthesis Theorem
  - **Solution of the Analysis Problem. Characteristic Equations**
    - Solution of the Characteristic Equations
    - Algorithm to Solve the Analysis Problem
  - Synthesis Problem: Recursive Algorithm
    - Synthesis Problem: Derivatives of Regular Expressions

Solution of the Analysis Problem. Characteristic Equations

ANALYSIS PROBLEM (AF→RE): Given a FA, write the characteristic equations of each one of its states, solve them and obtain the requested RE.

● **CHARACTERISTIC EQUATIONS:** They describe all the strings that can be recognized from a given state:

- An equation x_i is written for each state q_i
  - First member x_i;
  - The second member has a term for each branch from q_i
    - Branches has the format $a_{ij} \bullet x_j$ where a_{ij} is the label of the branch that joins qi with q_j, x_j is the variable corresponding to q_j
    - A term a_{ij} is added for each branch that joins q_i with a final state.
    - λ is added is q_i is a final state.
    - If there is not an output branch for a state, the second member will be:

If it is a final state: $x_i = \lambda$

If it not a final state: $x_i = \Phi$

OUTLINE

Unit 5. Part 2: Regular Expressions

- Definition of a Regular Expression (RE)
- Regular Expressions and Regular Languages
- Equivalence of Regular Expressions
- Analysis Theorem and Kleene's Synthesis Theorem
  - Solution of the Analysis Problem. Characteristic Equations
    - **Solution of the Characteristic Equations**
    - Algorithm to Solve the Analysis Problem
  - Synthesis Problem: Recursive Algorithm
    - Synthesis Problem: Derivatives of Regular Expressions

Solution of the Characteristic Equations

They have the form: **X = AX + B**

where:

X: set of strings that allow transitting from q_i to $q_f \in F$

A: set of strings that allows reaching a state q from q.

B: set of strings that allows reaching a final state, without reaching again the leaving state q_i.

$\Downarrow$ (Arden solution or proof by contradiction)

The solution is: **X = A* • B**

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

OUTLINE

Unit 5. Part 2: Regular Expressions

- Definition of a Regular Expression (RE)
- Regular Expressions and Regular Languages
- Equivalence of Regular Expressions
- Analysis Theorem and Kleene's Synthesis Theorem
  - Solution of the Analysis Problem. Characteristic Equations
    - Solution of the Characteristic Equations
    - **Algorithm to Solve the Analysis Problem**
  - Synthesis Problem: Recursive Algorithm
    - Synthesis Problem: Derivatives of Regular Expressions

Solution of the Analysis Problem. Algorithm

1. Write the characteristic equations of the FA.

2. Resolve them.

3. If the initial state is q_0, X_0 gives us the set of strings that leads from q_0 to q_f and, therefore, the language accepted by the FA.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

OUTLINE

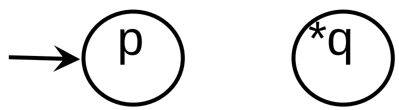
Unit 5. Part 2: Regular Expressions

- Definition of a Regular Expression (RE)
- Regular Expressions and Regular Languages
- Equivalence of Regular Expressions
- Analysis Theorem and Kleene's Synthesis Theorem
  - Solution of the Analysis Problem. Characteristic Equations
    - Solution of the Characteristic Equations
    - Algorithm to Solve the Analysis Problem
  - **Synthesis Problem: Recursive Algorithm**
    - Synthesis Problem: Derivatives of Regular Expressions

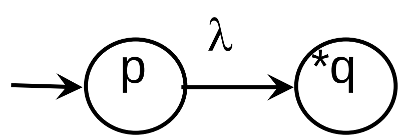
Synthesis Problem: Recursive Algorithm

SYNTHESIS PROBLEM (RE→FA): "Given an RE representing a regular language, build a FA that accepts that regular language.

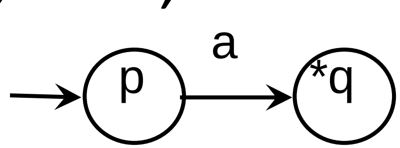
- Given a regular expression α:
  - If $\alpha = \Phi$, the automaton is:



  - If $\alpha = \lambda$, the automaton is:

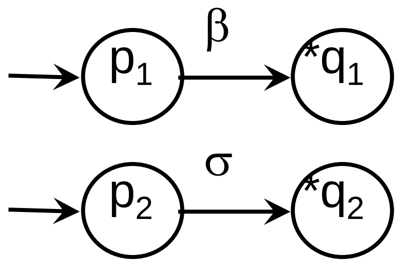


  - If $\alpha = a$, $a \in \Sigma$, the automaton is:

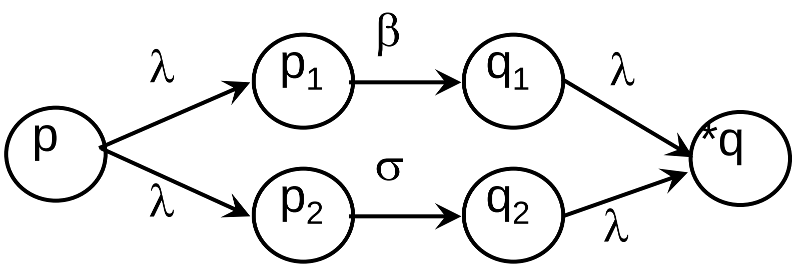


Synthesis Problem: Recursive Algorithm

- If $\alpha=\beta|\sigma$, using the automata β and σ

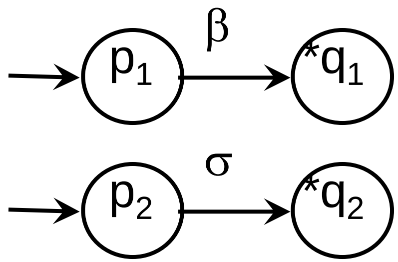


the result is:

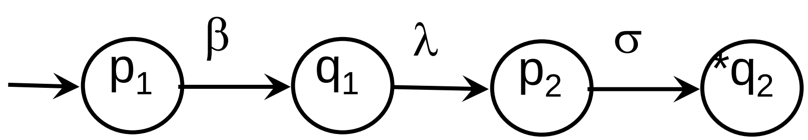


Synthesis Problem: Recursive Algorithm

- If $\alpha=\beta \bullet \sigma$, using the automata β and σ



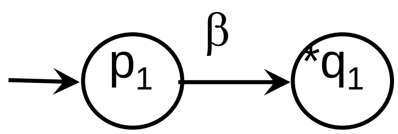
the result is:



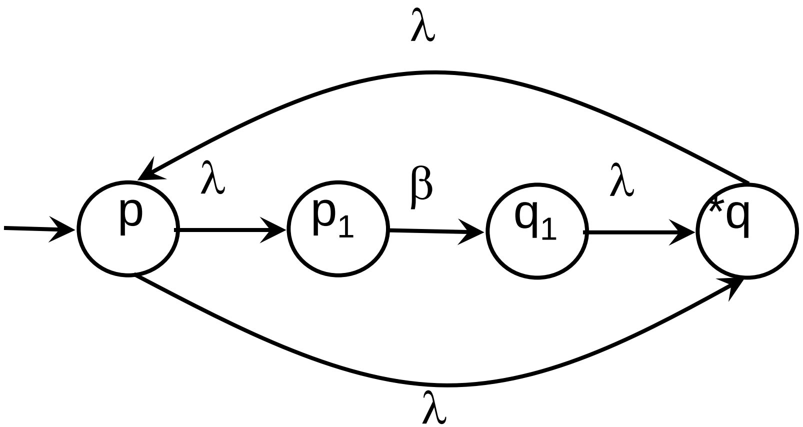
David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Synthesis Problem: Recursive Algorithm

- If $\alpha=\beta^*$, using the automata β



the result is:



OUTLINE

Unit 5. Part 2: Regular Expressions

- Definition of a Regular Expression (RE)
- Regular Expressions and Regular Languages
- Equivalence of Regular Expressions
- Analysis Theorem and Kleene's Synthesis Theorem
  - Solution of the Analysis Problem. Characteristic Equations
    - Solution of the Characteristic Equations
    - Algorithm to Solve the Analysis Problem
  - Synthesis Problem: Recursive Algorithm
    - **Synthesis Problem: Derivatives of Regular Expressions**

Solution to the synthesis problem: Derivatives of Regular Expressions

- Given a RE, construct a FA which recognizes the language that the RE describes.

  - Derive the RE and obtain a Right-Linear G3 and, from it, a FA.
    » Derivative of a RE?

- Derivative of a RE: $D_a(R) = \{ x \mid a \bullet x \in R \}$.

  » Derivative of a regular expression R with regard an input symbol a $\in \Sigma$ is the set of cues of every word represented by R whose head is a.

  » Let's see a recursive definition.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Solution to the synthesis problem: Derivatives of Regular Expressions

Given an RE → right-linear G3 grammar → FA which recognizes the language that describes the ER.

$$D_a(R) = \{ x \mid a.x \in R \}$$

Derivative of a RE: Recursive definition. $\forall$ a, b $\in \Sigma$ and R, S Reg. Exp.

- $D_a(\Phi) = \Phi$
- $D_a(\lambda) = \Phi$
- $D_a(a) = \lambda, \quad a \in \Sigma$
- $D_a(b) = \Phi, \quad \forall\, b \neq a, b \in \Sigma$
- $D_a(R+S) = D_a(R) + D_a(S)$
- $D_a(R \bullet S) = D_a(R) \bullet S + \delta(R) \bullet D_a(S) \quad \forall R$

  $\lambda \in \Sigma \;\Rightarrow\; \delta(R) = \lambda$

  $\lambda \notin \Sigma \;\Rightarrow\; \delta(R) = \Phi$
- $D_a(R^*) = D_a(R) \bullet R^*$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Solution to the synthesis problem: Derivatives of Regular Expressions

- **Definition: $D_{ab}(R)=D_b(D_a(R))$**
- From a derivative of a RE, obtain the right-linear G3 grammar.
  - The number of different derivatives of a RE is finite.
  - Once all have been obtained, you can obtain the G3 grammar:
  - **Given $D_a(R) = S$, with $S \neq \Phi$**

    $S \neq \lambda \Rightarrow R ::= aS \in P$

    $S = \lambda \Rightarrow R ::= a \in P$
  - **Given $\delta(D_a(R)) = S$**

    $\delta(D_a(R)) = \lambda \Rightarrow R::= a \in P$

    $\delta(D_a(R)) = \Phi \Rightarrow$ no rules included in P
- The axiom is R (starting RE)
- ΣT = symbols that make up the starting RE.
- ΣN = letters which distinguish each one of the different derivatives.