

CSCI 256: Theory of Computing

CHAPTER 4 *Properties of Regular Languages*

Dr. Benjoe A. Juliano

Tel. 530 898-4619 (office)
530 898-6442 (dept)
Fax. 530 898-5995

Juliano@ecst.csuchico.edu
<http://www.ecst.csuchico.edu/~juliano>

B. Juliano © 2002, based on notes by J. Ullman



The Pumping Lemma for RLs

Theorem 4.1

(The *pumping lemma for regular languages*.)

Let L be a regular language. Then there exists a constant n (which depends on L) such that for every string $w \in L$ such that $|w| \geq n$, $w = xyz$, such that:

1. $y \neq \epsilon$
2. $|xy| \leq n$
3. for all $k \geq 0$, $xy^kz \in L$

B. Juliano © 2002, based on notes by J. Ullman



The Pumping Lemma for RLs

Theorem 4.1 (*Pumping lemma, continued*)

PROOF:

- " Suppose L regular. Then $L = L(A)$ for some DFA $A = (Q, \Sigma, \delta, q_0, F)$.
- " Suppose $|Q| = n$.
- " Consider $w \in L$ where $|w| \geq n$
 - " Say $w = a_1 a_2 \dots a_m$, $m \geq n$, $a_i \in \Sigma$ for $1 \leq i \leq m$.
- " For $0 \leq i \leq n$, $p_i \ll \Delta(q_0, a_1 a_2 \dots a_i)$
 - " Note: $p_0 = q_0$

B. Juliano © 2002, based on notes by J. Ullman



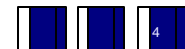
The Pumping Lemma for RLs

Theorem 4.1 (*Pumping lemma, continued*)

PROOF:

- " Since $|Q| = n$, by the *pigeonhole principle*, one cannot find $n+1$ different p_i 's for $0 \leq i \leq n$ to be distinct.
- " Find $i \neq j$, where $0 \leq i < j \leq n$, such that $p_i = p_j$.

B. Juliano © 2002, based on notes by J. Ullman



The Pumping Lemma for RLs

- **Theorem 4.1** (*Pumping lemma, continued*)

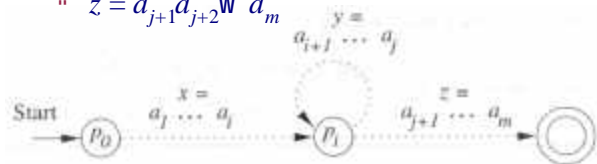
- **PROOF:**

Break $w = xyz$ as

$$x = a_1 a_2 \dots a_i$$

$$y = a_{i+1} a_{i+2} \dots a_j$$

$$z = a_{j+1} a_{j+2} \dots a_m$$



The Pumping Lemma for RLs

- **Theorem 4.1** (*Pumping lemma, continued*)

- **PROOF:**

For $k \geq 0$, let A receive the string $w = xy^kz$

Case 1: $k=0$.

Then, $w = xz = a_1 a_2 \dots a_i a_{j+1} a_{j+2} \dots a_m$. A

goes from $p_0 = q_0$ to p_i on prefix x .

Since $p_i = p_j$, then A continues from p_i to

$p_m \in F$ on suffix z . Thus, A accepts

$w = xz$.

The Pumping Lemma for RLs

- **Theorem 4.1** (*Pumping lemma, continued*)

- **PROOF:**

For $k \geq 0$, let A receive the string $w = xy^kz$

Case 2: $k > 0$.

Then, A goes from $p_0 = q_0$ to p_i on x , circles

from p_i to p_i k times on y^k , and then to

$p_m \in F$ on z .

Thus, for any $k \geq 0$, xy^kz is also accepted by A ;

that is, $xy^kz \in L$.

The Pumping Lemma for RLs

- The **Pumping Lemma (PL)** is used to show that a language L is *not* regular ...

1. Start by assuming L is regular. Then $L = L(A)$ for some DFA $A = (Q, \Sigma, \delta, q_0, F)$.

2. Let $n = |Q|$ serve as the **PL** constant.

- n does not really have to be anything specific

3. Choose some $w \in L$

- typically, w depends on n .

The Pumping Lemma for RLs

- Use of **PL**, *continued ...*
- 4. Applying the **PL**, we know w can be broken into xyz , satisfying the **PL** properties.
 - again, we may not know how to break w , so we use x,y,z as parameters.
- 5. Derive a contradiction by picking i such that $xy^iz \notin L$.
 - i might depend on parameter n , x , y , and/or z

B. Juliano © 2002, based on notes by J. Ullman

9

The Pumping Lemma for RLs

- **Example:**
 - $L = \{w \in \{0\}^* \mid \text{where } |w| \text{ is a square}\}$
 - Claim: L not regular.
 - Suppose L regular. Then $L=L(A)$ for some DFA $A=(Q,\Sigma,\delta,q_0,F)$; so, let $n=|Q|$.
 - Consider $w=0^{n^2} \in L$. Then $w=xyz$, where $|xy| \leq n$ and $y \neq \epsilon$.
 - By **PL**, $xyyz \in L$; but, $n^2 < |xyyz| \leq n^2 + n$.

B. Juliano © 2002, based on notes by J. Ullman

10

The Pumping Lemma for RLs

- **Example, continued ...:**
 - $L = \{w \in \{0\}^* \mid \text{where } |w| \text{ is a square}\}$
 - Claim: L not regular.
 - The next perfect square after n^2 is $(n+1)^2 = n^2 + 2n + 1$.
 - Thus $|xyyz|$ is not square, so $xyyz \notin L$.
 - Since we have derived a contradiction, the only unproved assumption — that L is regular — must be at fault. Therefore, L is not regular.

B. Juliano © 2002, based on notes by J. Ullman

11

From Exercise 4.1.1 of ITLC, Hopcroft, Motwani, & Ullman, 2001.

The Pumping Lemma for RLs

- **Examples:**
 - $L_1 = \{w \in \{0,1\}^* \mid w = 0^n 1^n, n \geq 1\}$
 - $L_2 = \text{set of strings of balanced parentheses}$
 - $L_3 = \{w \in \{0,1\}^* \mid w = 0^n 10^n, n \geq 1\}$
 - $L_4 = \{w \in \{0,1\}^* \mid w = 0^n 1^m 2^n, n, m \geq 0\}$
 - $L_5 = \{w \in \{0,1\}^* \mid w = 0^n 1^m, n \leq m\}$
 - $L_6 = \{w \in \{0,1\}^* \mid w = 0^n 1^{2n}, n \geq 1\}$

B. Juliano © 2002, based on notes by J. Ullman

12

The Pumping Lemma for RLs



The Pumping Lemma can be good for you ...

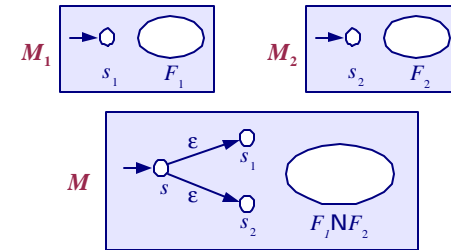
B. Juliano © 2002, based on notes by J. Ullman

Closure Properties of RLs

Boolean Operations

Theorem 4.4 (Closure under Union)

- If L and M are RLs, then so is $L \cup M$.



$$L(M) = L(M_1) \cup L(M_2)$$

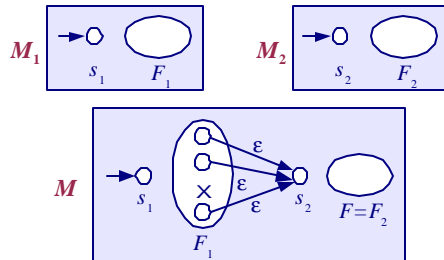
B. Juliano © 2002, based on notes by J. Ullman

Closure Properties of RLs

Boolean Operations

Theorem (Closure under Concatenation)

- If L and M are RLs, then so is LM .



$$L(M) = L(M_1) \cdot L(M_2)$$

B. Juliano © 2002, based on notes by J. Ullman

Closure Properties of RLs

Boolean Operations

Theorem (Closure under Kleene star)

- If L is RL, then so is L^* .

Theorem 4.5 (Closure under Complementation)

- If L is RL over Σ , then so is $\overline{L} = \Sigma^* - L$.

Theorem 4.8 (Closure under Intersection)

- If L and M are RLs, then so is $L \cap M$.

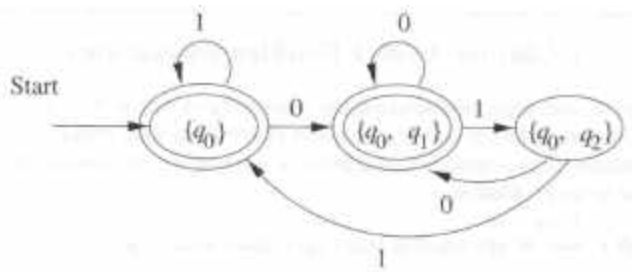
Theorem 4.10 (Closure under Difference)

- If L and M are RLs, then so is $L - M$.

B. Juliano © 2002, based on notes by J. Ullman

Closure Properties of RLs

From Figure 4.2 of (ATLC, Hopcroft, Motwani, & Ullman, 2001).

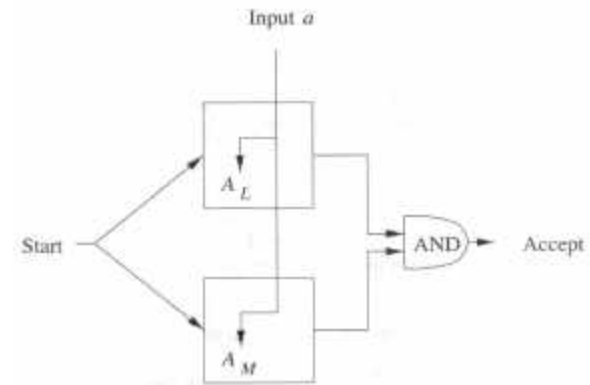
Automaton for $\{0, 1\}^* - L(A)$, where $L(A) = (0+1)^* 01$.

B. Juliano © 2002, based on notes by J. Ullman

17

Closure Properties of RLs

From Figure 4.2 of (ATLC, Hopcroft, Motwani, & Ullman, 2001).



$$A = (Q_L \times Q_M, \Sigma, \delta, (q_L, q_M), F_L \times F_M)$$
, where

$$\delta((p, q), a) = (\delta(p, a), \delta(q, a))$$

B. Juliano © 2002, based on notes by J. Ullman

18

Closure Properties of RLs

- **Reversal**
 - **Definition**
 - The **reversal** of a string $w = a_1 a_2 \dots a_n$, denoted w^R , is the string w written backwards as $a_n a_{n-1} \dots a_1$.
 - **Theorem 4.11** (*Closure under Reversal*)
 - If L is RL, then so is L^R .

B. Juliano © 2002, based on notes by J. Ullman

19

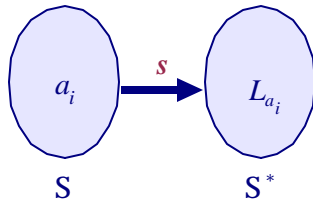
Closure Properties of RLs

- **Substitution**
 - Take a regular language L over some alphabet Σ .
 - For each $a \in \Sigma$, let L_a be a regular language.
 - Let s be the **substitution** defined by $s(a) = L_a$ for each a .
 - A Extend s to strings by $s(a_1 a_2 \dots a_n) = s(a_1) s(a_2) \dots s(a_n)$; i.e., concatenate the languages $L_{a_1} L_{a_2} \dots L_{a_n}$.
 - A Extend s to languages by $s(M) = \bigcup_{w \in M} s(w)$.
 - Then $s(L)$ is regular.

B. Juliano © 2002, based on notes by J. Ullman

20

Closure Properties of RLs



$$s: S \rightarrow S^*$$

- Extending s to strings by $s: S^* \rightarrow S^*$.
- Extend s to languages by $s: S^* \rightarrow S^*$.

B. Juliano © 2002, based on notes by J. Ullman

21

Closure Properties of RLs

Homomorphisms

- A string **homomorphism** is a function on strings that works by *substituting* a particular string for each symbol.

Theorem 4.14

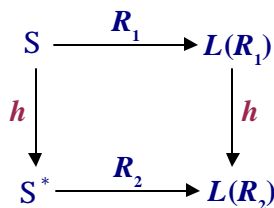
- If L is a regular language over alphabet Σ , and h is a homomorphism on Σ , the $h(L)$ is also regular.

B. Juliano © 2002, based on notes by J. Ullman

22

Closure Properties of RLs

Homomorphisms



- **Note:** R_i 's are regular expressions.

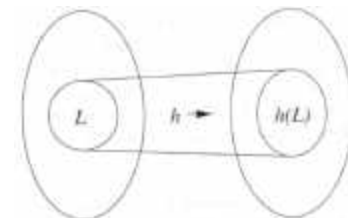
B. Juliano © 2002, based on notes by J. Ullman

23

Closure Properties of RLs

Inverse Homomorphisms

- Suppose h is a homomorphism from some alphabet S to strings in another (possibly the same) alphabet T . Let L be a language over alphabet T . Then $h^{-1}(L)$ is the set of strings $w \in S^*$ such that $h(w) \in L$.



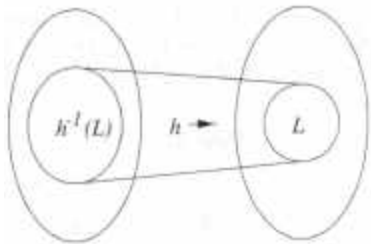
B. Juliano © 2002, based on notes by J. Ullman

24

Closure Properties of RLs

Theorem 4.16

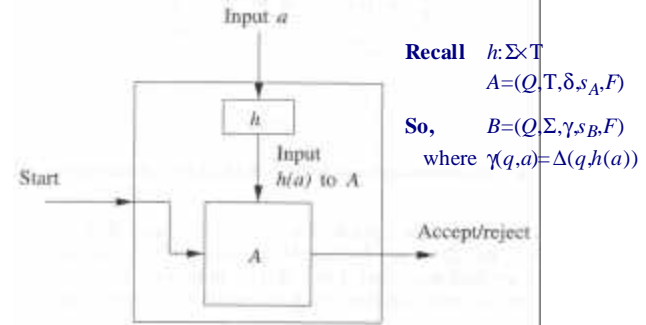
- If h is a homomorphism from alphabet S to alphabet T , and L is a regular language over T , then $h^{-1}(L)$ is also a regular language.



Closure Properties of RLs

Constructive Proof of Theorem 4.16

- Given a DFA A for RL L .



Recall $h: \Sigma \times T$
 $A = (Q, T, \delta, s_A, F)$

So, $B = (Q, \Sigma, \gamma, s_B, F)$
 where $\gamma(q, a) = \Delta(q, h(a))$

Decision Properties of RLs

Converting among Representations

- NFA-to-DFA, $O(n^3 2^n)$
 - Compute ϵ -closure in $O(n^3)$; for each of the 2^n states in DFA, compute transitions by consulting the ϵ -closure and NFA's transition table in $O(n^3)$.
- DFA-to-NFA, $O(n)$
 - Modify transition table to be on sets (and ϵ)

Decision Properties of RLs

Converting among Representations, *continued ...*

- FA-to-RE, $O(n^3 4^n)$
 - Generate n^2 expressions n times where size of the RE constructed can quadruple in each round.
- RE-to-FA, $O(n)$
 - RE of length n ; parse RE into expression tree and then use ϵ -NFA construction algorithm.

Decision Properties of RLs

- **Testing Emptiness of RLs**
 - " Choose DFA representation.
 - " Use a *graph reachability algorithm* to test if at least one accepting state is reachable from the start state.
 - " Note that reachability calculations take no more than $O(n^2)$ if the automaton has n states.

B. Juliano © 2002, based on notes by J. Ullman

29

Decision Properties of RLs

- **Testing Membership in a RL**
 - " Choose DFA representation.
 - " Simulate the DFA on input w .
- **Testing an RL's Finiteness**
 - " Every finite language is regular (why?).
 - " A regular language is not necessarily finite.
 - " DFA A with cycles $\Rightarrow L(A)$ is infinite.
 - " RE E , presence of $*$ almost always means infinite, except for annihilators and ϵ^* .

B. Juliano © 2002, based on notes by J. Ullman

30

Equivalence & Minimization

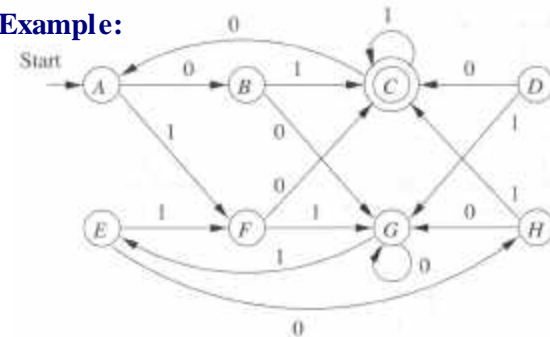
- **Testing Equivalence of States**
 - Real goal is testing equivalence of representations of two regular languages
 - Interesting fact: DFAs have unique (up to state names) minimum-state equivalents
 - States $p, q \in Q$ of DFA A are *equivalent* if
 - For all w , $\Delta(p, w)$ is an accepting state if and only if $\Delta(q, w)$ is an accepting state.
 - Note that $\Delta(p, w)$ and $\Delta(q, w)$ do not have to be the same state.
 - If two states are not equivalent, then they are *distinguishable*.

B. Juliano © 2002, based on notes by J. Ullman

31

Equivalence & Minimization

Example:



- $\{C, G\}$ distinguishable
- $\{A, G\}$ distinguishable by 01 or 10, but not by ϵ , 0 or 1.
- $\{A, E\}$ equivalent (Why?)

B. Juliano © 2002, based on notes by J. Ullman

32

Equivalence & Minimization

- Testing Equivalence of States, *continued ...*
 - Table-filling algorithm** (via recursive discovery)
 - BASIS: $p \in F, q \notin F \Rightarrow \{p, q\}$ distinguishable
 - INDUCTION: Let $p, q \in Q$ such that for some $a \in \Sigma$, $r = \delta(p, a)$ and $s = \delta(q, a)$ are distinguishable. Then, $\{p, q\}$ distinguishable.
 - $\exists w \in \Sigma^*$ that distinguishes r from s ; i.e., either $\Delta(r, w) \in F$ or $\Delta(s, w) \in F$, but not both. Then, string aw must distinguish p from q since $\Delta(p, aw) = \Delta(r, w)$ and $\Delta(q, aw) = \Delta(s, w)$.

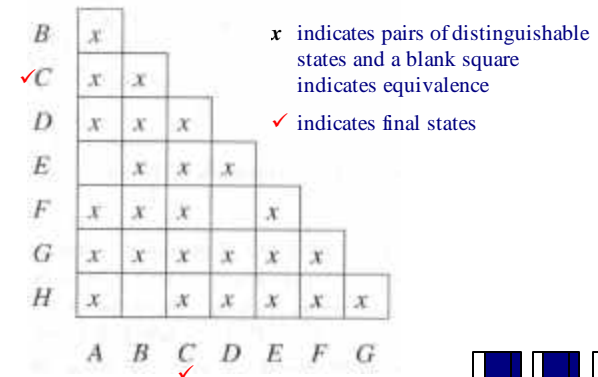
B. Juliano © 2002, based on notes by J. Ullman

33

From Figure 4.9 of *ITLC*, Hopcroft, Motwani, & Ullman, 2001.

Equivalence & Minimization

- Example:** (See Figure 4.8 or Slide 32)



B. Juliano © 2002, based on notes by J. Ullman

34

Equivalence & Minimization

- Theorem 4.20**
 - If two states are not distinguished by the *table-filling algorithm*, then the states are equivalent.
- Testing Equivalence of RLs**
 - Given regular languages L_1 and L_2
 - Convert each representation to a DFA
 - Consider DFA A where $L(A) = L_1 \cup L_2$
 - Use the table-filling algorithm to test if $\{s_1, s_2\}$ are equivalent; if so, $L_1 = L_2$

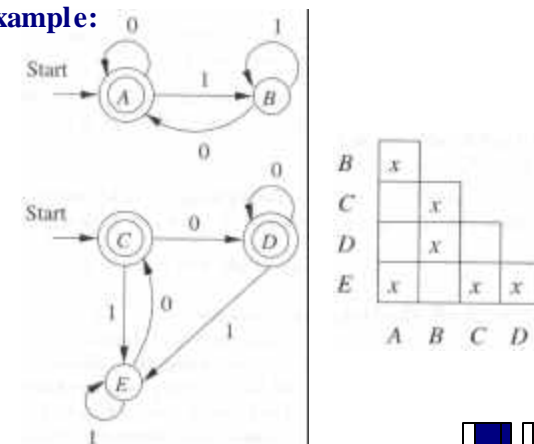
B. Juliano © 2002, based on notes by J. Ullman

35

From Figures 4.10 and 4.11 of *ITLC*, Hopcroft, Motwani, & Ullman, 2001.

Equivalence & Minimization

- Example:**



B. Juliano © 2002, based on notes by J. Ullman

36

Equivalence & Minimization

- Minimization of DFAs
 - For each DFA there is an equivalent DFA that has as few states as any DFA accepting the same language. Further, this minimum-state DFA is unique for the language.
 - State equivalence partitions the set of states.

B. Juliano © 2002, based on notes by J. Ullman

37

From Figure 4.9 of *ITLC*, Hopcroft, Motwani, & Ullman, 2001.

Equivalence & Minimization

- Example:

B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x
	A	B	C	D	E	F	G

Partitions: {A,E}
 {B,H}
 {C}
 {D,F}
 {G}

B. Juliano © 2002, based on notes by J. Ullman

38

Equivalence & Minimization

- Example:

B	x		
C		x	
D		x	
E	x		x x
	A	B	C

Partitions: {A,C,D}
 {B,E}

B. Juliano © 2002, based on notes by J. Ullman

39

Equivalence & Minimization

- Theorem 4.23
 - The equivalence of states is transitive.
- Theorem 4.24
 - If we create for each state q of a DFA a *block* consisting of q and all the states equivalent to q , then the different blocks of states form a partition of the set of states.

B. Juliano © 2002, based on notes by J. Ullman

40

Equivalence & Minimization

- **DFA minimization algorithm** $A=(Q_A, \Sigma, \delta_A, s_A, F_A)$
 1. Use the *table-filling algorithm* to find all pairs of equivalent states.
 2. Partition the set of states Q_A into blocks of mutually exclusive states by the method described above.
 3. Construct the minimum-state equivalent DFA B by using the blocks as its states.
 - " s_B is the block containing s_A .
 - " F_B is the set of blocks containing $f \in F_A$.

Equivalence & Minimization

- **Theorem 4.26**
 - If A is a DFA and M the DFA constructed from A by the *DFA minimization algorithm*, then M has as few states as any DFA equivalent to A .

Copyright and Intellectual Property Notice

This document and its contents are the *Intellectual Property* (IP) of Dr. Benjoe A. Juliano of the Department of Computer Science at California State University, Chico (CSUC). Dr. Juliano claims exclusive moral rights of ownership under current *Copyright Laws* (Title 17 of the United States Code and 1998 Digital Millennium Copyright Act) and *IP Policies/Guidelines* (CSUC EM83-08, EM97-07, and Article 39 of the CFA/CSU Contract) including, but not limited to:

- the exclusive right to copy, reproduce, and/or distribute this document;
- the right to be identified as the creator of this work (the right of attribution);
- the right to take action against false attribution; and
- the right to object to derogatory treatment of this work (the right of integrity).