Theory of Computation

Regular Expression

Outline

- What are Regular Expression
- Operators in Regular Expressions
- Equivalence between Regular Expression and Finite States Automata
  - Regular Expression → NFA
  - Regular Language → Regular Expression

From Sipser Chapter 1.3

Regular Expressions

- Means of characterizing languages based on the regular operators
- Examples:
  - $(0 \cup 1)0^*$
    - A 0 or 1 followed by any number of 0's
    - Concatenation operator implied
  - What does $(0 \cup 1)^*$ mean?
    - All possible strings of 0 and 1 or ε
    - If $\Sigma = \{0,1\}$, then equivalent to Σ^*

Definition of Regular Expression

R is a regular expression if R is

1. a, for some a in alphabet Σ
2. ε
3. $\emptyset$
4. $(R1 \cup R2)$, where R1 and R2 are regular expressions
5. $(R1 \cdot R2)$, where R1 and R2 are regular expressions
6. $(R1^*)$, where R1 is a regular expression

Note: This is a recursive definition, common in computer science

- R1and R2 always smaller than R, so no issue of infinite recursion
- $\emptyset$ means language does not include any strings and ε means it includes the empty string

Operator precedence

- * has precedence over concatenation and union
- Concatenation has precedence over union
- Parentheses may change the precedence
- Example: $(0(0\cup 1)0)^*\cup 0$
  - $R1 = (0(0\cup 1)0)^*$; $R2= 0$
    - $R1\cup R2$
  - $R1 = (R3)^*$
  - $R4 = 0$; $R5 = 0\cup 1$; $R6= 0$
    - $R3=R4\cdot R5\cdot R6$
  - $R7=0$; $R8= 1$;
    - $R5 = R7\cup R8$

Is this different from $(0(0\cup 1)0)^*\cup(0)$? ✗

Is this different from $(00\cup 10)^*\cup 0$? ✓

Additional notation for *

- $R^+ = R^*R = RR^*$
  - Concatenation of at least one string from R
- R^k
  - Shor hand notation for concatenation of k strings from R
- $R^+ \cup \varepsilon = R^*$

Some Examples

- $0^*10^* =$
  - {w| w contains a single 1}
- $\Sigma^*1\Sigma^*=$
  - {w| w has at least one 1}
- $01 \cup 10 =$
  - {01, 10}
- $(0 \cup \varepsilon)(1 \cup \varepsilon) =$
  - {ε, 0, 1, 01}

Testing your understanding

- $R \cup \varnothing = R$
- $R\varepsilon = R$
- $R \cup \varepsilon = R$ if ε in R or $\{R, \varepsilon\}$ otherwise
- $R\varnothing = \varnothing$

Equivalence of Regular Expressions and FA

Theorem: A language is regular if and only if some regular expression describes it

Two directions so we need to prove:

- If a language is described by a regular expression then it is regular
- If a language is regular then there exists a regular expression that that describes it

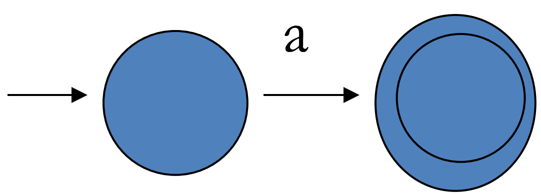
Proof: Regular Expression → Regular Language

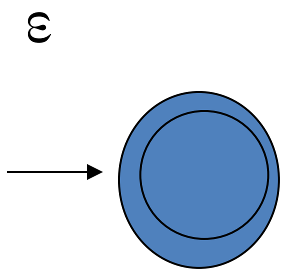
- Proof idea: Given a regular expression R describing a language L, we will….
  - Show that some FA recognizes it
  - Use NFA since may be easier and equivalent to DFA
- How do we do this?
  - We will use definition of a regular expression and show that we can build a FA covering each step.
    - Steps 1,2 and 3 of definition (handle alphabet symbols, ε, and ∅)
    - Steps 4,5 and 6 (handle union, concatenation, and star)

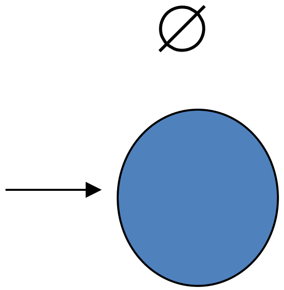
Proof: Regular Expression ➔ Regular Language

For steps 1-3 we construct the FA below. As a reminder:

1. a, for some a in alphabet Σ
2. ε
3. $\varnothing$







Proof Continued

- For steps 4-6 (union, concatenation and star) we use the result we previously obtained showing that FA are closed under union, concatenation, and star
- We have shown how to convert a Regular Expression into a FA which recognizes the same language
- By corollary, said language is regular.

Example: Regular Expression → NFA

Convert $(ab \cup a)^*$ to an NFA

(example 1.56 page 68)

- Outline of required steps:
  - Handle a
  - Handle b
  - Handle ab
  - Handle $ab \cup a$
  - Handle $(ab \cup a)^*$
- Sometimes ε-transitions may appear unnecessary of confusing
  - Be systematic! Always start including ε-transitions!

Equivalence of Regular Expressions and FA

Theorem: A language is regular if and only if some regular expression describes it

Two directions we need to prove:

- ✓ If a language is described by a regular expression then it is regular
- If a language is regular, then there exists a regulat expression that describes it

Proof: Regular Language → Regular Expression

- Proof strategy:
  - A regular language is accepted by a DFA
  - We need to show that can convert any DFA to a regular expression
- Two steps:
  - We construct a Generalized Non-deterministic Finite State Automaton (GNFA) from a DFA
  - We convert a GFA into a Regular Expression

GNFA in Special Form

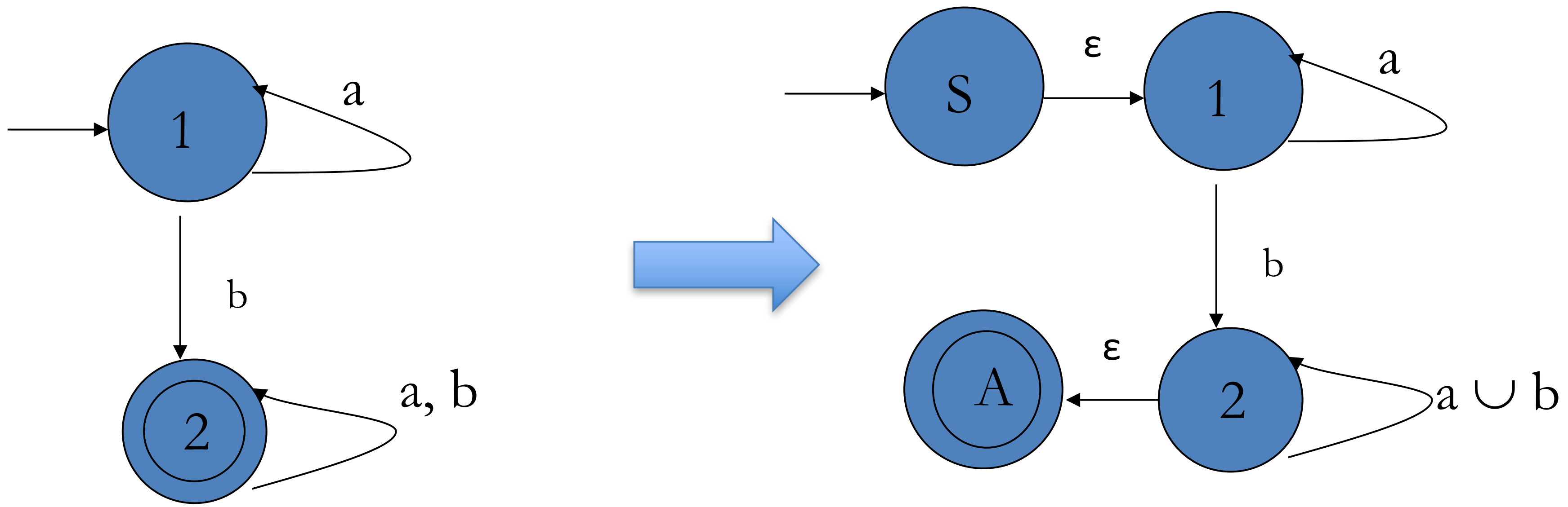
- GFAs are NFAs where transition may be labeled with regular expressions rather than just symbols from Σ
- GFAs in special form have the following properties
  - One start state with outgoing arrows going to all other states but no incoming arrows
  - One single accept states with no outgoing arrows and arrows incoming from any other state
  - All other states have arrows incoming and outgoing to every state, including themselves

DFA → GNFA

1. Add a new start state with one arrow labeled with ε to old start state
2. Add new accept state with arrows labeled with ε from all old accept states
3. If any arrow from remaining states has multiple labels, replace them with equivalent Regular Expression

   E.g., a,b -> a $\cup$ b
4. Add remaining arrows marked as $\varnothing$

DFA → GNFA example



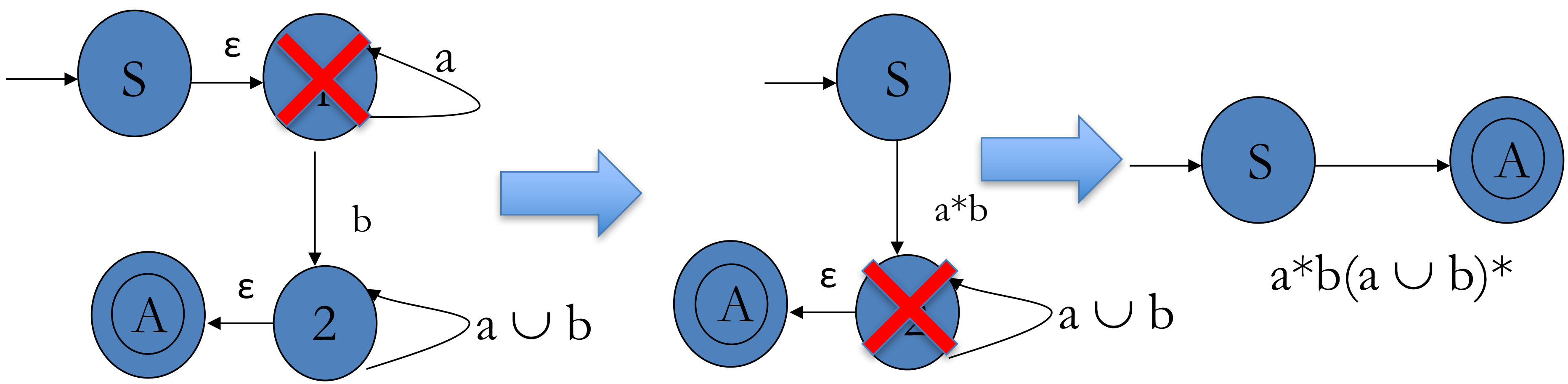
Edges marked with ∅ are redundant and may be confusing!

GNFA → Regular Expression

- We proceed in a series of steps reducing the number of states of the GFA to 2 (start and accept)
- The Regular Expression left on the only remaining arrow is equivalent to the GFA and, hence, the DFA

GFA → Regular Expression

- While GNFA has states other than "Start" and "Accept"
  - Pick a state q and remove it from the GNFA
  - Repair the transitions by combining the regular expressions by concatenation



Formalization of the proof

The textbook provides a rigorous proof by induction:

- CONVERT: procedure to transform GNFA G into Regular Expression
- Statement: L(G) = CONVERT(G)
  - Base: G has only 2 states
  - Inductive hypothesis: Statement holds if G has i>= 2 states
  - Inductive step: we show it holds if G has i+1 states
    1. Let G' denote the version of G with i states obtained after one application of CONVERT(G)
    2. We argue that if a string is accepted by G it will also be accepted by G' and vice versa
    3. Apply inductive hypothesis

Equivalence of Regular Expressions and FA

Theorem: A language is regular if and only if some regular expression describes it

Two directions we need to prove:

- ✓ If a language is described by a regular expression then it is regular
- ✓ If a language is regular then there exisits a regular expression that describes it