Theory of Computation

Regular Expression

Outline

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

From Sipser Chapter 1.3

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

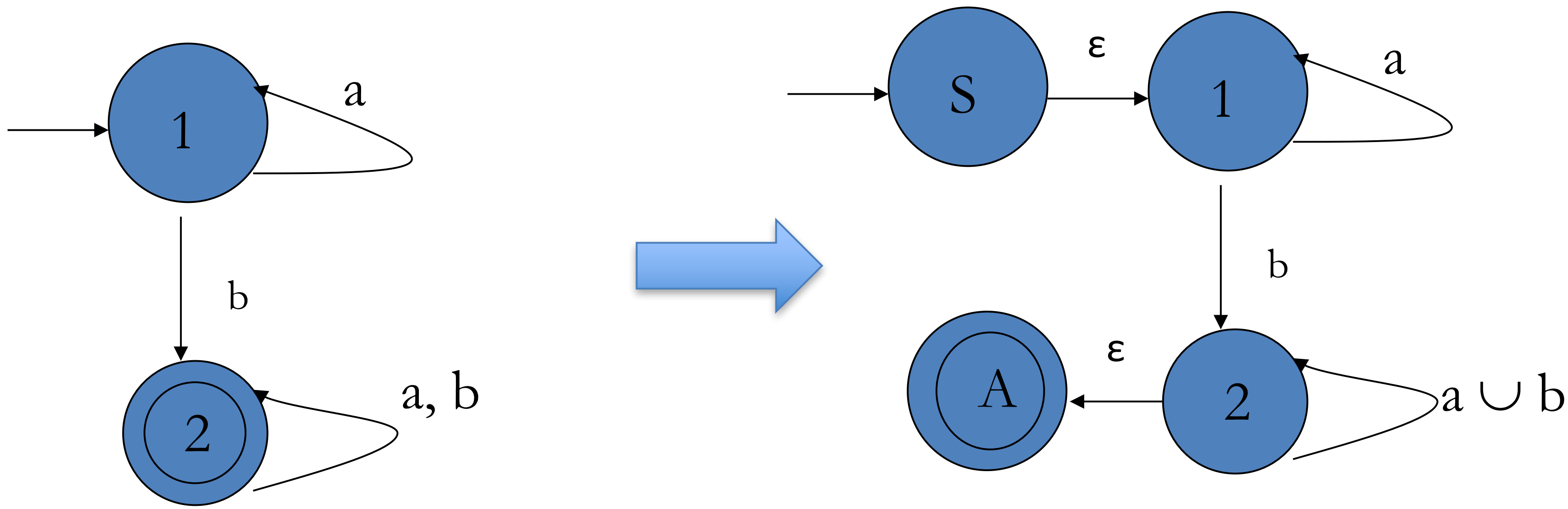
- GFAs are NFAs where transition may be labeled with regular expressions rather than just symbols from Σ
- GFAs in special form have the following properties
  - One start state with outgoing arrows going to all other states but no incoming arrows
  - One single accept states with no outgoing arrows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



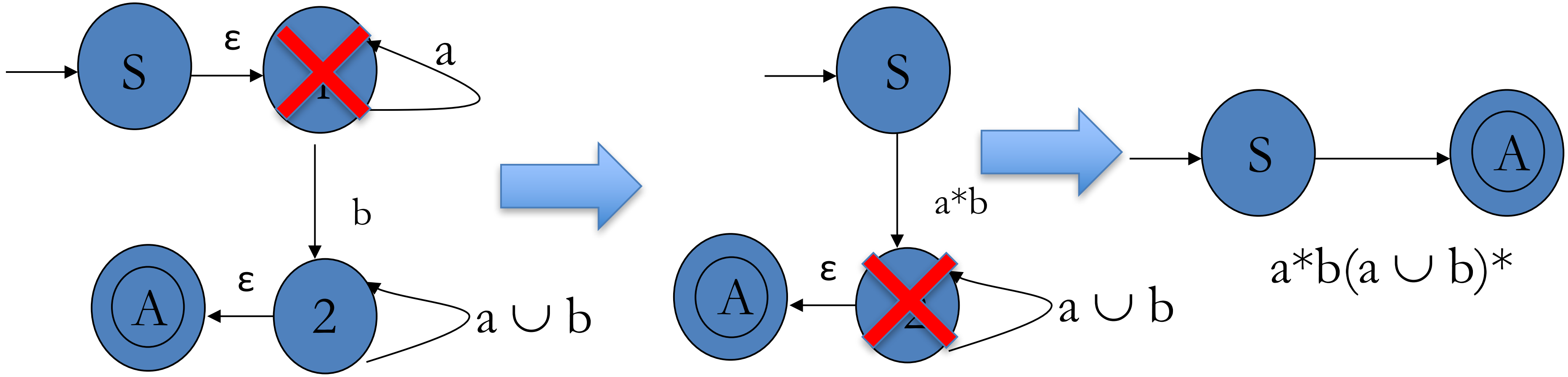
Edges marked with ∅ are redundant and may be confusing!

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