Copyright Warning

Curtin University of Technology

Theoretical Foundations of Computer Science

Lecture 3a

Regular Languages

Curtin University of Technology

Topics

- Regular operations
  - ➢ Union, concatenation, star
- Regular expressions
  - ➢ Definition and concepts
  - ➢ Equivalence with FA

Curtin University of Technology

Assessment Criteria

- **Express** an English or Mathematics specification as a RE.
- **Convert** a formal RE specification into an equivalent NFA specification.
- **Convert** a formal DFA specification into an equivalent RE specification.

REGULAR OPERATIONS

Regular Languages

Regular Operations

Union, Concatenation, Star

Closure

NFA and Regular Languages

- A language is called a *regular language* if some (deterministic) finite automaton recognizes it

- Any regular language can be recognized by an NFA (and hence a DFA)

- NFA only recognize regular languages
  - a corollary of previous theorem

Curtin University of Technology

Properties of Regular Languages

- Investigating the properties of automata and regular languages
  - ➢ Help develop a toolbox of techniques for designing automata for particular languages
  - ➢ Ways to prove certain languages to be beyond the capability of automata
- Operations to manipulate languages
  - ➢ Similar to arithmetic operations for numbers
  - ➢ Here the objects are languages rather than numbers
  - ➢ Used to study properties of regular languages

Regular Operations

- Let *A* and *B* be languages
  - ➢ Union: collects together strings of *A* and *B*
  - ➢ Concatenation: attaches strings from *A* in front of strings from *B* in all possible ways
  - ➢ Star: attaches any number of strings together (including the empty string, ε)

- The idea is to be able to directly specify a regular language.

Union

- Let A and B be languages

- $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
  - Collects together strings of A and B

- If A={good, bad}, and B={boy, girl}, then
  $A \cup B$ = {good, bad, boy, girl}

Concatenation

- Let A and B be languages

- $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
  - attaches strings from A in front of strings from B in all possible ways

- If $A = \{\text{good, bad}\}$, and $B = \{\text{boy, girl}\}$, then $A \circ B = \{\text{goodboy, goodgirl, badboy, badgirl}\}$

Curtin University of Technology

Star

- Let A be a language
- $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$
  - ➢ attaches any number of strings together
  - ➢ including the empty string (k=0) denoted ε
- If A={good, bad}, then
  A^* = {ε, good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, goodbadgood, goodbadbad, ...}

Curtin University of Technology

REGULAR EXPRESSIONS

Information Definition

Recursive Definition

Examples

Identities

Curtin University of Technology

Regular Expressions

- Start just with the alphabet
- Regular operations: $\cup$, $\circ$, $*$
- Use regular operations to build expressions.
- Value of a regular expression is a language.
- Example:
  - $(0 \cup 1)0^*$
  - What does it mean?

Meaning of Regular Expression

- Example: $(0 \cup 1)0^*$
  - Symbols 0 and 1 are shorthand for $\{0\}$ and $\{1\}$
  - $(0 \cup 1)$ means $(\{0\} \cup \{1\})$ or language $\{0,1\}$
  - 0^* stands for $\{0\}^*$, language of strings with any number of 0s
  - $(0 \cup 1)0^*$ is shorthand for $(0 \cup 1) \circ 0^*$
  - Consists of all strings starting with a 0 or a 1 followed by any number of 0s.

Curtin University of Technology

Role of regular expressions

- In CS applications
  - ➢ Searching for strings that satisfy certain patterns.
  - ➢ Regular expressions describe such patterns.

- Applications where regular expressions used for describing patterns
  - ➢ Utilities (*e.g.*, AWK, GREP in UNIX)
  - ➢ Programming languages (*e.g.*, PERL)
  - ➢ Text editors

Curtin University of Technology

Operator Precedence

- Star (*)
- Concatenation (°)
- Union (∪)
- Use parentheses to alter the usual order

Curtin University of Technology

Formal Definition

- R is a regular expression if R is
  - a for some a in the alphabet Σ,
  - ε,
  - ϕ,
  - $(R_1 \cup R_2)$ where R_1 and R_2 are regular expressions,
  - $(R_1 \circ R_2)$ where R_1 and R_2 are regular expressions, or
  - $(R_1{}^*)$ where R_1 is a regular expression.

Curtin University of Technology

Explanation

- Items of the definition:
  - a and ε represent languages $\{a\}$ and $\{\varepsilon\}$
  - ϕ represents the empty language
  - remaining items represent languages obtained by using operations, union, concatenation, or star
  - $\{\varepsilon\}$ represents a language containing an empty string
  - ϕ represents a language containing no strings

Examples

- Assume alphabet Σ is $\{0,1\}$.
  - $0^*10^* = \{w|w \text{ has exactly a single } 1\}$
  - $\Sigma^*1\Sigma^* = \{w|w \text{ has at least one } 1\}$
  - $\Sigma^*001\Sigma^* = \{w|w \text{ contains } 001 \text{ as a substring}\}$
  - $(\Sigma\Sigma)^* = \{w|w \text{ is a string of even length}\}$
  - $(\Sigma\Sigma\Sigma)^* = \{w|\text{length of } w \text{ is a multiple of } 3\}$

Curtin University of Technology

Examples

- Assume alphabet Σ is $\{0,1\}$.
  - $01 \cup 10 = \{01, 10\}$
  - $0\Sigma^*0 \cup 1\Sigma^*1 = \{w|w$ starts and ends with the same symbol$\}$
  - $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$
  - $(0 \cup \varepsilon)\,(1 \cup \varepsilon) = \{(\varepsilon, 0, 1, 01\}$
  - $1^*\phi = \phi$
  - $\phi^* = \{\varepsilon\}$

Curtin University of Technology

Identities

- $R \cup \phi = R$
  - Adding the empty language to another language
- $R \circ \varepsilon = R$
  - Adding the empty string to any string
- However,
  - $R \cup \varepsilon$ may not equal R (*e.g.*, if R=0, then $L(R)=\{0\}$, but $L(R \cup \varepsilon) = \{0, \varepsilon\}$)
  - $R \circ \phi$ may not equal R (*e.g.*, if R=0, then $L(R)=\{0\}$, but $L(R \circ \phi) = \phi$)

Curtin University of Technology

Regular expressions as tools in Design of compilers

- Programming language tokens such as variables and constants may be described with regular expressions
  - ➢ Example: A numerical constant may be described as a member of the language
  - ➢ $\{+,-,\varepsilon\}(DD^* \cup DD^*.D^* \cup D^*.\ DD^*)$,
  - ➢ where $D=\{0,1,\ldots,9\}$ is a digit

- Once tokens are described, the lexical analyzer can be generated automatically

Curtin University of Technology

CONVERSION BETWEEN RE AND FA

Equivalence: RL are RE

Convert a RE into NFA

Convert NFA into RE

Equivalence With Finite Automata

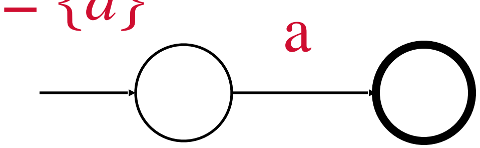
- Regular expressions and finite automata have the same descriptive power.
  - ➢ Any finite automaton can be converted to the regular expression it describes and vice versa.
  - ➢ Theorem: A language is regular iff some regular expression describes it.
  - ➢ Proof by construction.
    - if part: from any regular expression, construct an NFA
    - only if part: given any DFA, convert into a regular expression

Curtin University of Technology

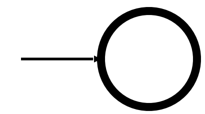
Converting R into an NFA

- Consider the six cases in the formal definition of regular expressions
  - a for some a in the alphabet Σ,
  - ε,
  - ϕ,
  - $(R_1 \cup R_2)$,
  - $(R_1 \circ R_2)$, or
  - $(R_1{}^*)$
  - where R_1 and R_2 are regular expressions.

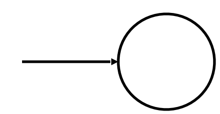
$R=a$, $L(R)=\{a\}$



$R=\varepsilon$, $L(R)=\{\varepsilon\}$

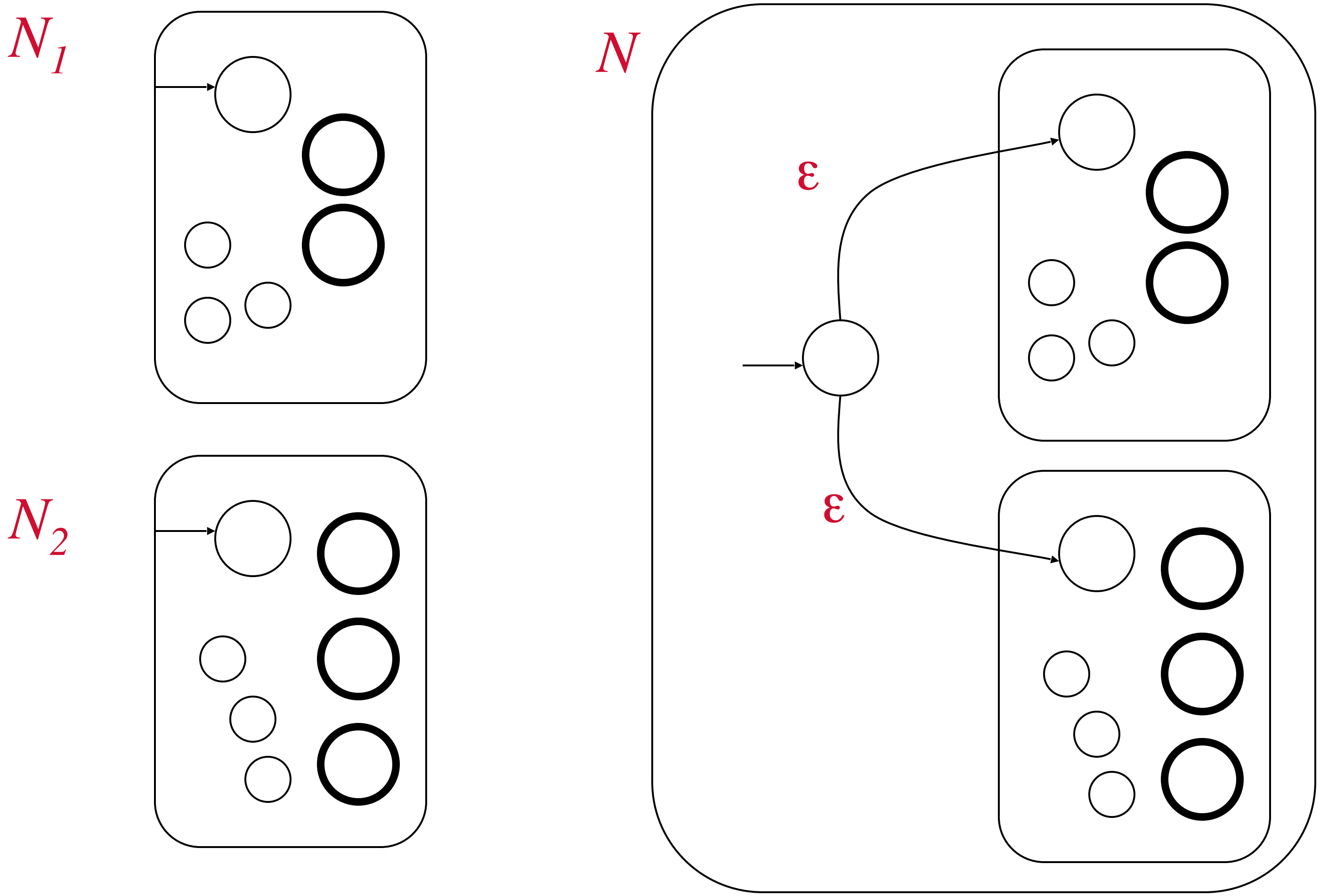


$R=\phi$, $L(R)=\phi$

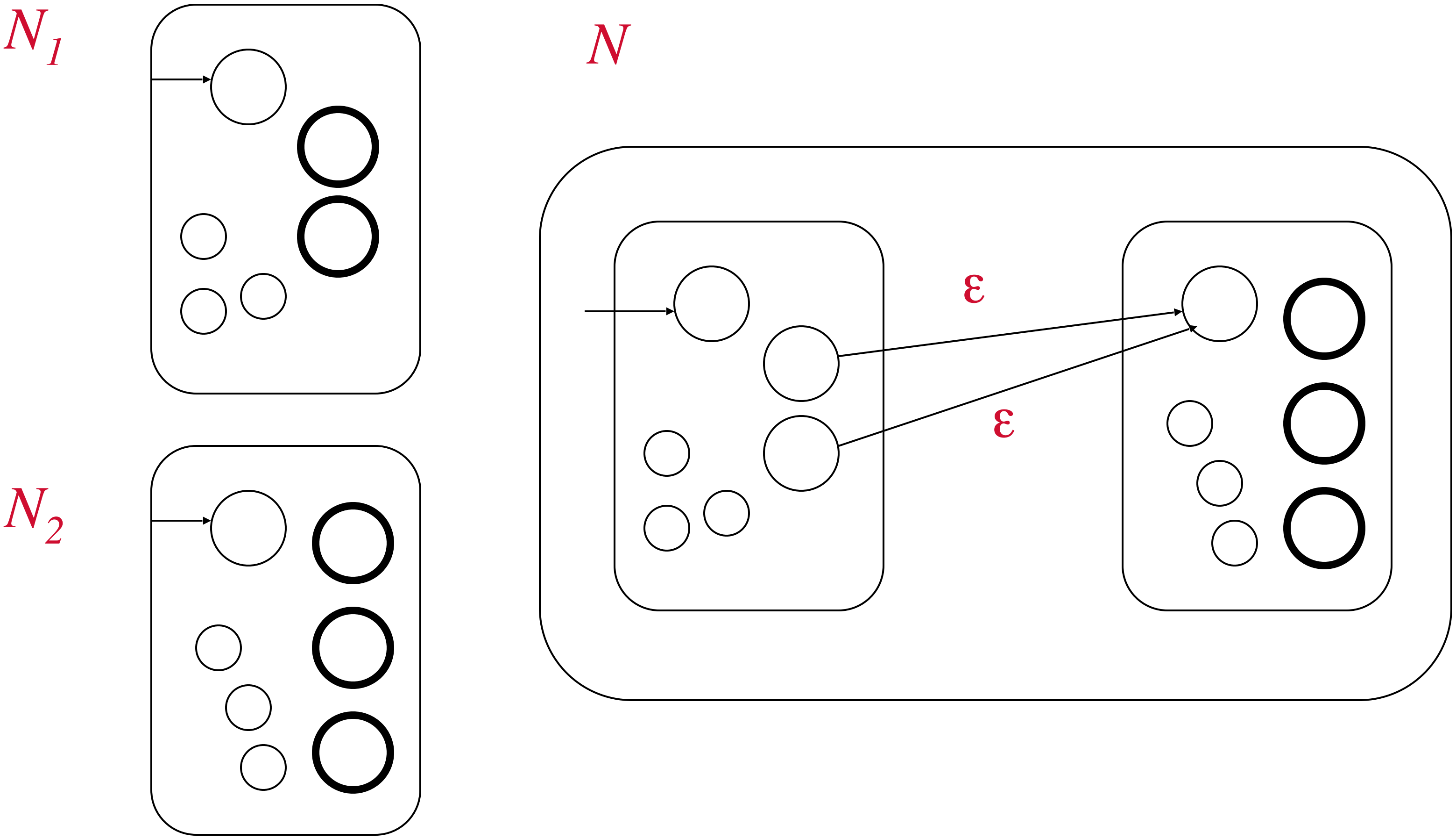


- For each one, we can build an equivalent NFA (for the first three) or find an equivalent way to join two NFAs.

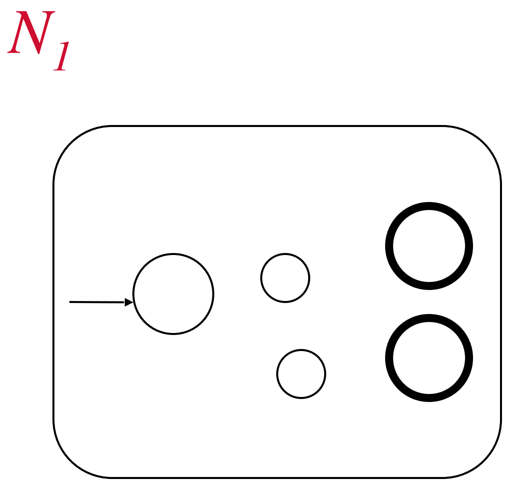
Construction of NFA for $N_1 \cup N_2$

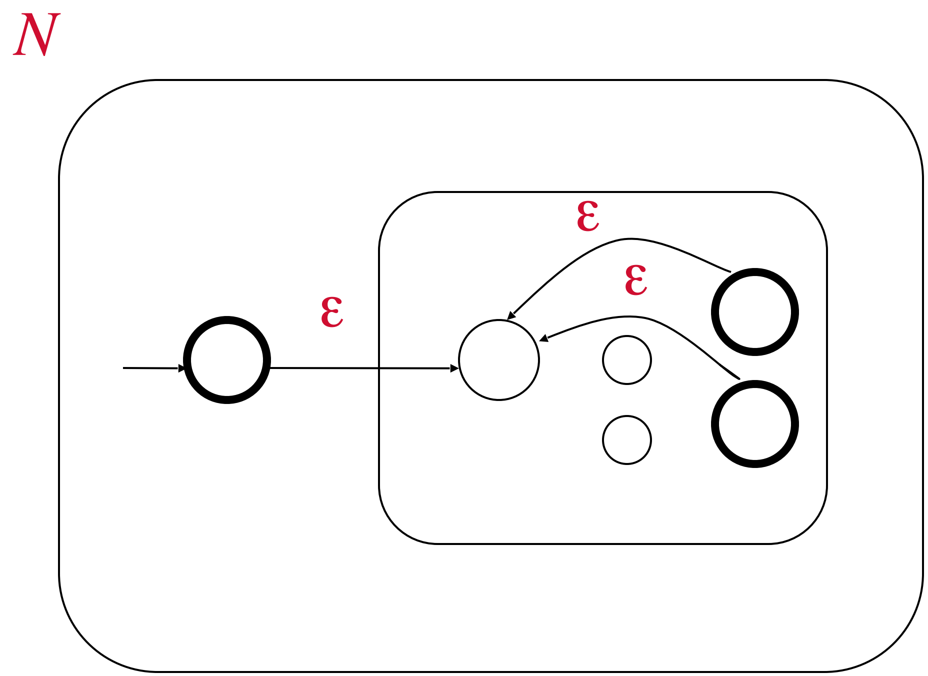


Construction of NFA for $N_1 \circ N_2$



Construction of NFA for N*

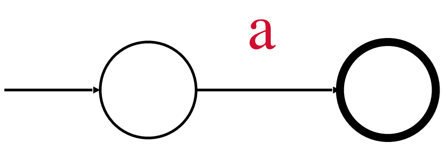




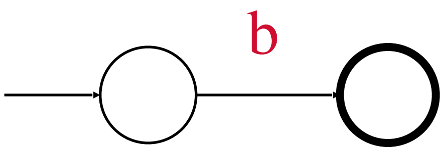
Example

- Convert (ab ∪ a)* to an NFA.

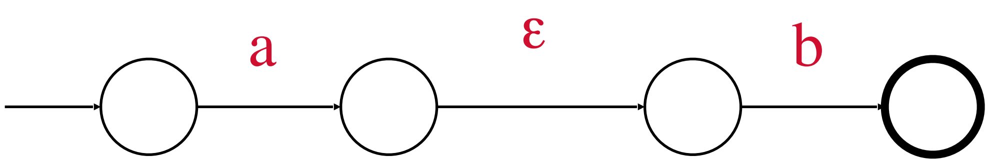
a



b



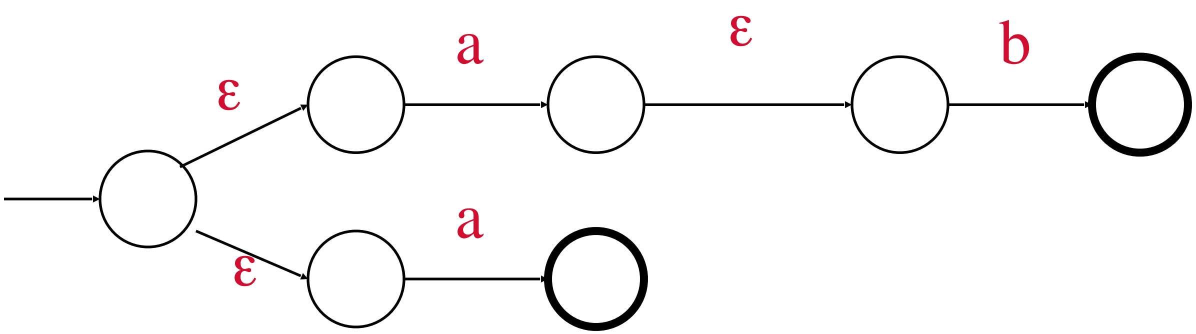
ab



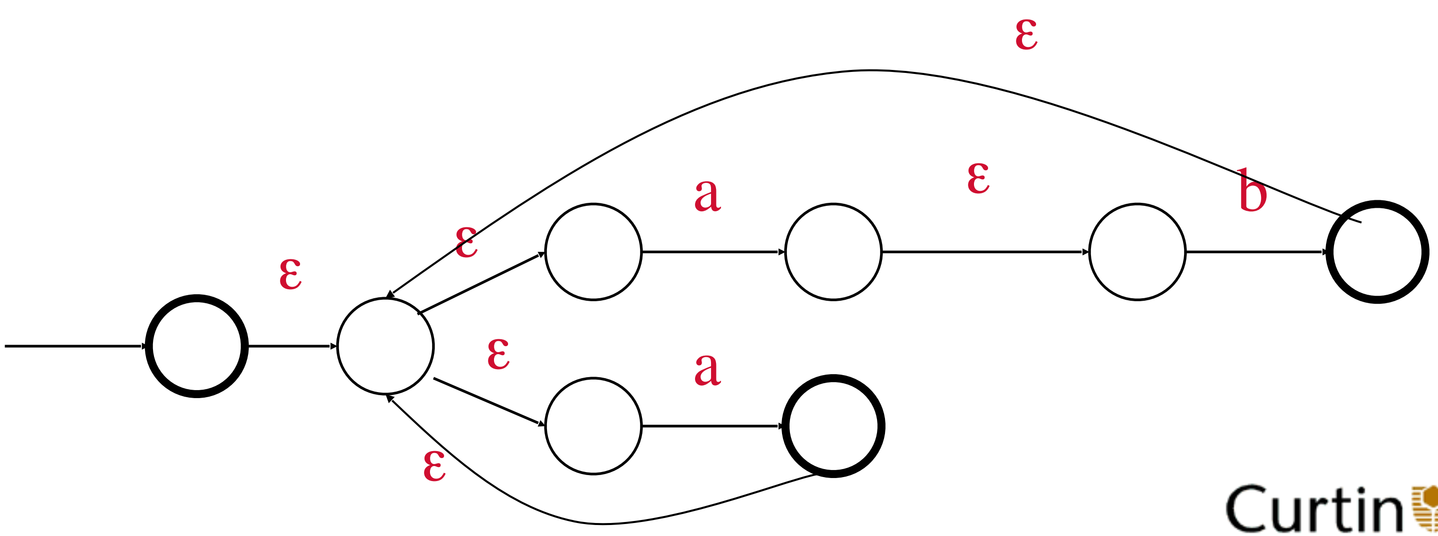
Example: RE to NFA

- Convert (ab ∪ a)* to an NFA.

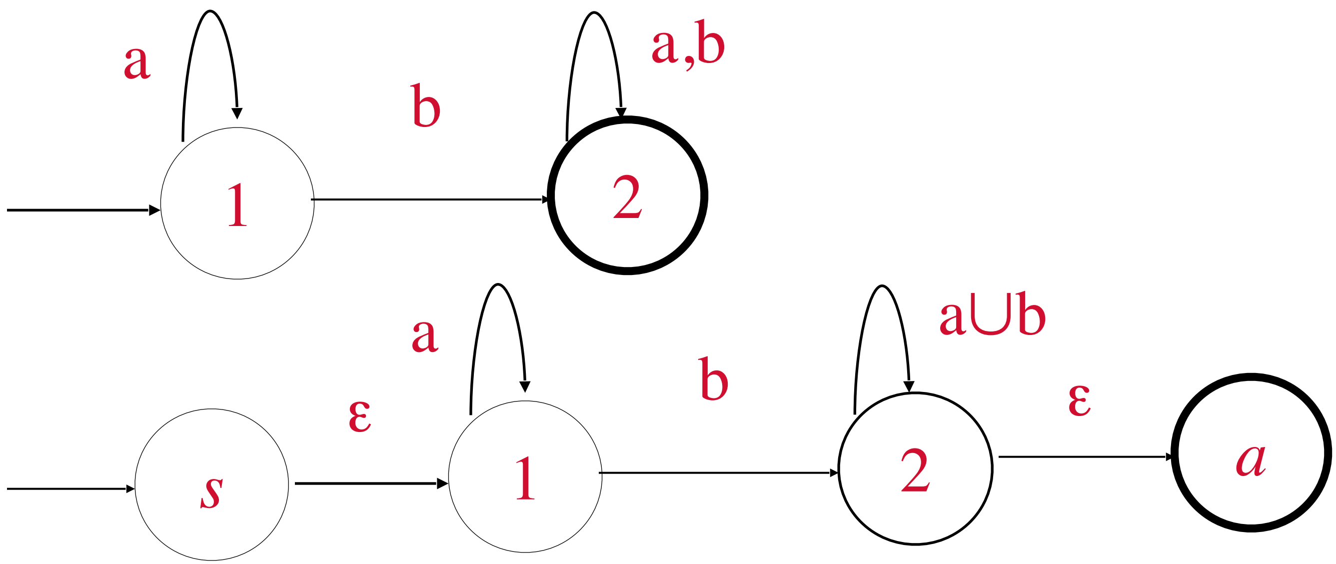
(ab ∪ a)



(ab ∪ a)*

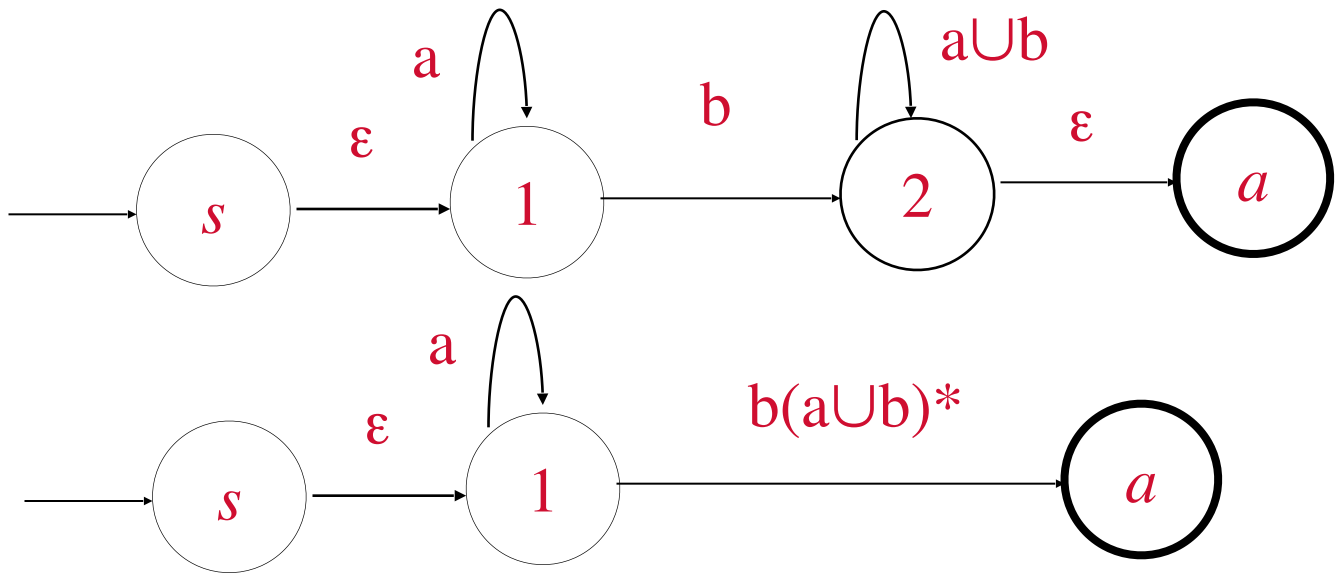


DFA to regular expression



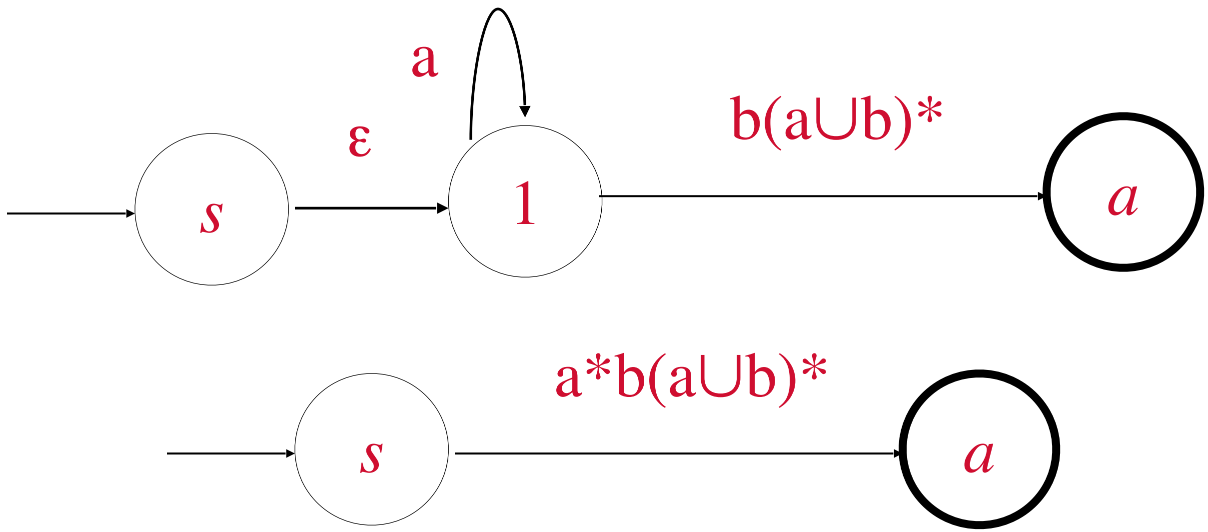
- Make a 4-state Generalized NFA:
  - ➢ Add a new start state *s* and a new accept state *a* to the DFA.
  - ➢ GNFA allows transitions on blocks of symbols instead of only one symbol per transition.

DFA to RE



- First rip out state 2
  - ➢ Repair the GNFA so that it will accept the same set of strings as before.

DFA to RE



- Next rip out state 1:
  - ➢ Repair the GNFA so that it will accept the same set of strings as before.
  - ➢ Now the arrow from *s* to *a* is labeled by the regular expression corresponding to the DFA.

Non-regular languages

- Limitations of finite automata
  - ➢ certain languages cannot be recognized by any finite automaton
- Example: Language $B = \{0^n 1^n \mid n \geq 0\}$
  - ➢ Claim:
    - a machine recognizing B need to remember how many 0s have been seen so far as it reads input
    - an unlimited number of states needed for this
    - Thus non-regular

- Need to be able to prove that something is non-regular.

Curtin University of Technology

Summary

- Regular operations
  - Union, concatenation, star
- Regular expressions
  - Definition
    - <ULO> Express specification as a RE
  - Equivalence with FA
    - <ULO> Convert RE to NFA and Convert DFA to RE