Example 1.41

- Convert to DFA using construction

Example

\[ \begin{array}{c}
\text{State 1} \\
\text{State 2} \\
\text{State 3}
\end{array} \]

\[ \begin{array}{c c c}
\text{Input} & \text{Transition} & \text{Next State} \\
\text{a} & \text{State 1} & \text{State 2} \\
\text{b} & \text{State 2} & \text{State 3} \\
ε & \text{State 1} & \text{State 1}
\end{array} \]

- Can remove states with no input
- Determine transitions using union and ε
- Determine accept states
- Determine states from DFA
- What are the states of DFA D?
- What are the states of DFA D? in Proof of Theorem 1.39
- Determine DFA using construction

Overview

- Regular Expressions
- Nondeterminism (continued)
The class of regular languages is closed under the concatenation operation.

\[ A \circ B = \{ xy \mid x \in A \text{ and } y \in B \} \]

Proof Idea:

The class of regular languages is closed under the star operation.

\[ A^* = \{ x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A \} \]

Proof Idea:

More Closure Properties

Theorem 1.45: Regular languages are closed under the union operation.

\[ A \cup B = \{ x \mid x \in A \text{ or } x \in B \} \]

Proof Idea: We can use NFAs.

Prove this with simpler construction than we did with DFAs.

Theorem 1.46: Regular languages are closed under the intersection operation.

\[ A \cap B = \{ x \mid x \in A \text{ and } x \in B \} \]

Proof Idea: We can use NFAs.

Prove this with simpler construction than we did with DFAs.

Note that NFA is not useful for proving intersection.

\[ A \cap B = \{ x \mid x \in A \text{ and } x \in B \} \]

Proof Idea: We can use NFAs.

Prove this with simpler construction than we did with DFAs.

Theorem 1.47: Regular languages are closed under the concatenation operation.

\[ A \circ B = \{ xy \mid x \in A \text{ and } y \in B \} \]

Proof Idea:
Regular Expressions

- In arithmetic, we can use operations to build up expressions describing languages.
- Example: \((0 \cup 1)0^*\)
- This has a value of 32
- Example: \((0 + 3) \times 4\)
- In arithmetic, we can use + and \(\times\) to build up expressions describing languages.
- Similarly, can use operations to build up expressions describing languages.
- Example: \((5 + 3) \times 4\)

Regular Expressions

- Nondeterminism (Continued)
Another Example

If \( \{0,1\} \) we can write \( \mathcal{L} \) as shorthand for \( \{0,1\} \).

Any string of 0's and 1's starts with the language \( \{0,1\} \) and applies the operation \( \ast \). So

\[ \{0,1\} \ast \{0,1\} \text{ is shorthand for } \{0,1\} \ast \{0,1\} \text{, which is the language consisting of all strings containing any number of 0's and 1's.} \]

\[ \text{Symbols 0 and 1 are shorthand for the sets } \{0\} \text{ and } \{1\}. \]

What is language of \( \{0,1\} \ast \{0,1\} \)?

Meaning of Regular Expressions (Informally)

- String from second language
- Concatenation involves all combinations of a string from first language with
- Similar to how \( \times \) in arithmetic can be implicit (e.g. \( 3y \))
- Between \( \{0\} \) and \( \{0,1\} \), there is an implicit concatenation operation.
- For example, \( \{0\} \ast \{0,1\} \) means \( \{0\} \ast \{0,1\} \), and the value is the language consisting of all strings containing any number of 0's and 1's. So
- Symbols 0 and 1 are shorthand for the sets \( \{0\} \) and \( \{1\} \).
Notes

Formal Definition

Definition 1.52: We say that $R$ is a regular expression if $R$ is

1. a for some $a$ in the alphabet

2. $\epsilon$

3. $\emptyset$

4. $(R_1 \cup R_2)$, where $R_1$ and $R_2$ are regular expressions

5. $(R_1 \circ R_2)$, where $R_1$ and $R_2$ are regular expressions

6. $(R_1)^*$, where $R_1$ is a regular expression

where $H$ is a regular expression. 

Union has higher precedence than concatenation.
Identities

- \( R \cup \emptyset \)
- \( \emptyset \circ R \)
- \( \emptyset \cap R \)
- \( R \circ \emptyset \)

Examples

- \( 0^* \)
- \( 1^* \)
- \( 0 \cup 1 \)
- \( (01)^* \)
- \( 100^* \)
- \( 1^* 0^* \)
A language is regular iff it is described by a regular expression

**Theorem 1.54:** A language is regular iff it is recognized by a NFA.
Regular Language → Regular Expression

- Restricted version of GNFA makes the proof simpler
  - other state and from each state to itself
  - except start and accept state, one arrow goes from every state to every
  - arrow going out, and no loop same as the start state
  - single accept state, with arrows coming in from every other state, but no
  - single state has transitions to every other state, but no arrows coming in

- We will use a GNFA with the following additional restrictions:
  - start state
  - arcs
  - arcs less state, by using more and more complex regular expressions on
  - will convert an DFA into a GNFA, and then into GNFA with successively
  - like in NFA, but can have regular expressions on its transitions

Generalized nondeterministic finite automaton (GNFA)

Proof idea: convert DFA into a regular expression

Example

\[(a \cup qa)^*\]
Induction

• GNFA (in restricted form) with more than 2 states
  • Proof Idea:
    - Remove a state that is not the start nor accept state
    + Repair the labels on the remaining transitions
  • Proof Idea in more detail:
    - Remove state $q_{\text{rip}}$
    + Alter label from $q_i$ to $q_j$
    + Say $R_4$ was label from $q_i$ to $q_j$
    + Say $R_1$ is label from $q_i$ to $q_{\text{rip}}$
    + Say $R_2$ is self-loop label on $q_{\text{rip}}$
    + Say $R_3$ is label from $q_{\text{rip}}$ to $q_j$
    - New label is $(R_1)(R_2)^* (R_3) \cup (R_4)$

• Easy to convert DFA into GNFA
  - Make a new start state with $\epsilon$ transition to the old start state
  - Make a new accept state with $\epsilon$ transition from old accept states
  - Any arc with multiple labels use union instead
  - Add arrows labeled $\emptyset$ where needed
  - No self loops (except loops are allowed on start and accept)
  - One will be start, another will be accept
  - Base Case: just has 2 states
  - First case: just has 2 states
  - Just one transition
  + No self loops (aren't allowed on start and accept)
  - One will be start, one will be accept

Proof
Example: Convert into a Regular Expression

1

2

\[ a, b^n \]