Regular Languages

Peter Bauer

V_01.02.00

Outline

Motivation

Definitions

- Regular Grammar
- Regular Language
- Regular Set
- Regular Expression

Finite Automata

- Deterministic Finite Automaton (DFA)
- Transformations

Motivation

- Simplest type of grammar
- Applications in pattern matching
- Implementations
  - lex / flex
  - In editors for searching (vi, emacs, ...)
  - In many scripting languages (Perl, Python, JavaScript, ...)
  - In frameworks like .NET (`System.Text.RegularExpressions`) or Java (`java.util.regex`)

Regular Grammar

Definition

A grammar G is called a *regular grammar* if it has only rules of the form

- $A \rightarrow \alpha$,
- $A \rightarrow \alpha B$,

where $A, B \in V_N$ and $\alpha \in V_T$.

Regular Grammar

Definition

A grammar G is called a *regular grammar* if it has only rules of the form

- $A \rightarrow \alpha$,
- $A \rightarrow \alpha B$,

where $A, B \in V_N$ and $\alpha \in V_T$.

Remark

G may also have the rule $S \rightarrow \varepsilon$ if S does not appear on the right side of the grammar.

Reverse Regular Grammar

Definition

A grammar is called a *reverse regular grammar* if it has only rules of the form $A \rightarrow \alpha$ or $A \rightarrow B\alpha$, where $A, B \in V_N$ and $\alpha \in V_T$.

Reverse Regular Grammar

Definition

A grammar is called a *reverse regular grammar* if it has only rules of the form $A \rightarrow \alpha$ or $A \rightarrow B\alpha$, where $A, B \in V_N$ and $\alpha \in V_T$.

Remark

G may also have the rule $S \rightarrow \varepsilon$ if S does not appear on the right side of the grammar.

An Example

Example

The following regular grammar describes a simple form of an identifier. c is a terminal class of characters and d is a terminal class of digits.

$$\begin{array}{lcl} S & \rightarrow & c \mid cR \\ R & \rightarrow & c \mid d \mid cR \mid dR \end{array}$$

Regular Language

Definition

The language $L(G(S))$ of a regular grammar G is a *regular language* over an alphabet Σ and is defined as follows:

Regular Language

Definition

The language $L(G(S))$ of a regular grammar G is a *regular language* over an alphabet Σ and is defined as follows:

- The empty set Φ is a regular language.

Regular Language

Definition

The language $L(G(S))$ of a regular grammar G is a *regular language* over an alphabet Σ and is defined as follows:

- The empty set Φ is a regular language.
- The set with the empty string $\{\varepsilon\}$ is a regular language.

Regular Language

Definition

The language $L(G(S))$ of a regular grammar G is a *regular language* over an alphabet Σ and is defined as follows:

- The empty set Φ is a regular language.
- The set with the empty string $\{\varepsilon\}$ is a regular language.
- For each $\omega \in \Sigma$ the language $\{\omega\}$ is a regular language.

Regular Language

Definition

The language $L(G(S))$ of a regular grammar G is a *regular language* over an alphabet Σ and is defined as follows:

- The empty set Φ is a regular language.
- The set with the empty string $\{\varepsilon\}$ is a regular language.
- For each $\omega \in \Sigma$ the language $\{\omega\}$ is a regular language.
- If A and B are regular languages, then $A \cup B$, AB (concatenation), and A^* (Kleene star) are regular languages.

Regular Language

Definition

The language $L(G(S))$ of a regular grammar G is a *regular language* over an alphabet Σ and is defined as follows:

- The empty set Φ is a regular language.
- The set with the empty string $\{\varepsilon\}$ is a regular language.
- For each $\omega \in \Sigma$ the language $\{\omega\}$ is a regular language.
- If A and B are regular languages, then $A \cup B$, AB (concatenation), and A^* (Kleene star) are regular languages.
- No other language over Σ is a regular language.

Regular Set

Definition

Analogously to regular languages we can define *regular sets* as follows:

- Φ, $\{\varepsilon\}$, and $\{\omega\}$ for each $\omega \in \Sigma$ are regular sets.
- If A and B are regular sets, then $A \cup B = \{\alpha \mid \alpha \in A \vee \alpha \in B\}$, $AB = \{\alpha\beta \mid \alpha \in A \wedge \beta \in B\}$, and $A^* = \bigcup_{n \geq 0} P_n$ are regular sets.
- Nothing else is a regular set.

Regular Set

Definition

Analogously to regular languages we can define *regular sets* as follows:

- Φ, $\{\varepsilon\}$, and $\{\omega\}$ for each $\omega \in \Sigma$ are regular sets.
- If A and B are regular sets, then $A \cup B = \{\alpha \mid \alpha \in A \vee \alpha \in B\}$, $AB = \{\alpha\beta \mid \alpha \in A \wedge \beta \in B\}$, and $A^* = \bigcup_{n \geq 0} P_n$ are regular sets.
- Nothing else is a regular set.

Example

Let $V = \{\mathtt{c}, \mathtt{d}\}$ be a set where `c` stands for a character and `d` stands for a digit. $\{c\}(\{c\} \cup \{d\})^*$ is a regular set that describes a simple form of identifier.

Regular Expression

Regular expression describe regular sets by

- Omitting the curled brackets
- Writing $+$ instead of $\cup$

Regular Expression

Regular expression describe regular sets by

- Omitting the curled brackets
- Writing $+$ instead of $\cup$

Remark

- To avoid parentheses the following priority is assumed:
  1. Kleene star has the highest priority
  2. Concatenation has the second highest priority
  3. $+$ has the lowest priority
- Sometimes $|$ is used instead of $+$

Regular Expression

Regular expression describe regular sets by

- Omitting the curled brackets
- Writing $+$ instead of $\cup$

Remark

- To avoid parentheses the following priority is assumed:
  1. Kleene star has the highest priority
  2. Concatenation has the second highest priority
  3. $+$ has the lowest priority
- Sometimes $|$ is used instead of $+$

Example

The identifier example from above would boil down to $\mathtt{c}(\mathtt{c}+\mathtt{d})^*$.

Deterministic Finite Automaton – DFA

Definition

A *Deterministic Finite Automaton* (DFA) is a quintuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- Σ is a set of inputs
- $\delta : Q \times \Sigma \to Q$ the state transition function
- q_0 the start state
- $F \subseteq Q$ the final states

Deterministic Finite Automaton – DFA

Definition

A *Deterministic Finite Automaton* (DFA) is a quintuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- Σ is a set of inputs
- $\delta : Q \times \Sigma \to Q$ the state transition function
- q_0 the start state
- $F \subseteq Q$ the final states

Remark

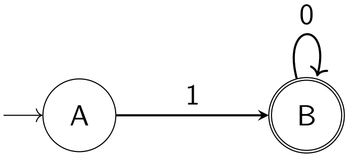
DFAs are often depicted by means of state diagrams

State Diagrams

- $Q = \{A, B\}$
- $\Sigma = \{0, 1\}$
- $\delta(A, 1) = B, \delta(B, 0) = B$
- $q_0 = A$
- $F = \{B\}$

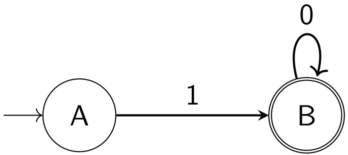
State Diagrams

- $Q = \{A, B\}$
- $\Sigma = \{0, 1\}$
- $\delta(A, 1) = B, \delta(B, 0) = B$
- $q_0 = A$
- $F = \{B\}$



State Diagrams

- $Q = \{A, B\}$
- $\Sigma = \{0, 1\}$
- $\delta(A, 1) = B, \delta(B, 0) = B$
- $q_0 = A$
- $F = \{B\}$

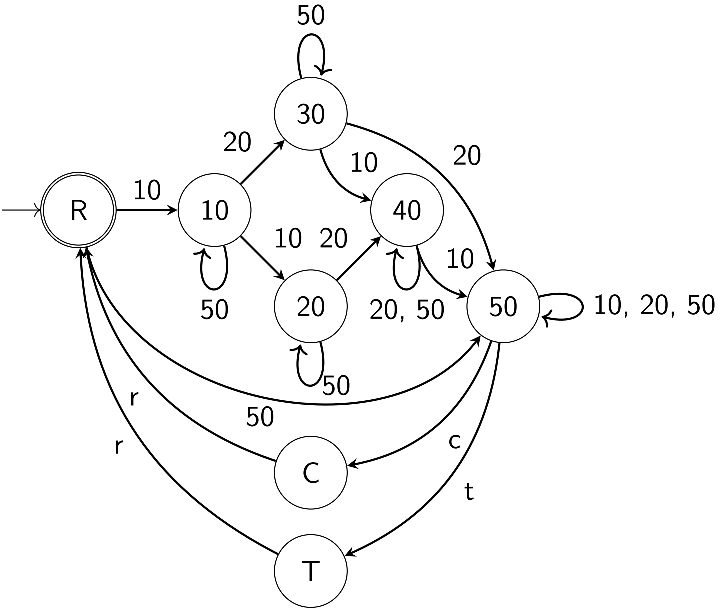


- States are circles
- Transitions are arrows
- Inputs are labels on the arrows
- Initial states are circles with an arrow from "nowhere"
- Final states are double circles

Coffee Machine

- Coins of 10, 20 and 50 cents are accepted
- Machine does NOT give change, i.e., if a coin is inserted which would cause an "overpay" the coin is thrown into the return tray and the machine doesn't change state.
- After a total amount of 50 cents inserted "c" for coffee and "t" for tea can be chosen.
- When the machine prepares tea or coffee it is in a state T or C, respectively.
- When the chosen drink is ready the user must press a button "o" to open the chamber in order to grab the cup and the machine goes back into its initial state.

DFA of a Coffee Machine

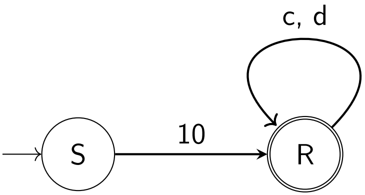


DFA Accepting a Simple Identifier

$$
\begin{array}{lcl}
S & \rightarrow & c \mid cR \\
R & \rightarrow & c \mid d \mid cR \mid dR
\end{array}
$$

DFA Accepting a Simple Identifier

$$
\begin{array}{rcl}
S & \rightarrow & c \mid cR \\
R & \rightarrow & c \mid d \mid cR \mid dR
\end{array}
$$



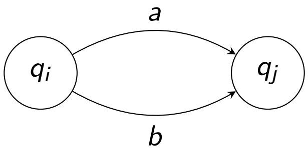
DFA Syntax Check Procedure

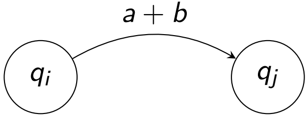
- checkString is a function of a class DFA
- input is the string to be checked
- delta is the state transition function δ

```
func checkString(_ input: String) -> Bool {
  var currentState: State = initialState
  var stringIsValid = true
  for c in input {
    if let nextState = delta(currentState, c), stringIsValid {
      // transition exists and we are still on track
      currentState = nextState
    } else {
      stringIsValid = false
    }
  }
  return stringIsValid && finalStates.contains(currentState)
}
```

Transformation of a DFA to a Regular Expression — Step 1

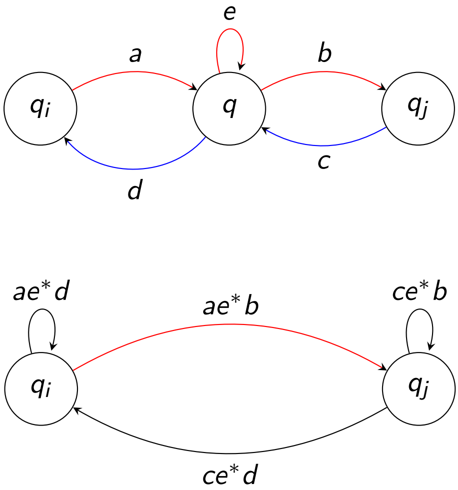
Replace different transitions from one state to another into one





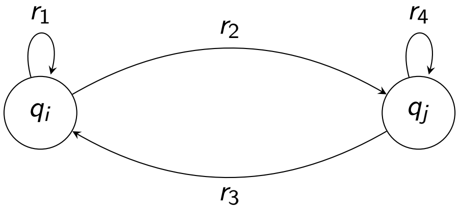
Transformation of a DFA to a Regular Expression — Step 2

Remove intermediate states



Transformation of a DFA to a Regular Expression — Step 3

Remove remaining states



- We need $r_1^* r_2$ in any way to go from q_i to q_j.
- Then we have two possibilities to reach q_j again:
  - Either we repeat r_4
  - Or we repeat $r_3 r_1^* r_2$
- This results in $r_1^* r_2 (r_4 + r_3 r_1^* r_2)^*$

Transformation of a Regular Expression to a DFA

Given a regular expression $\mathtt{a} \mid \mathtt{b}(\mathtt{cc}^* \mid (\mathtt{a} \mid \mathtt{c})^*)\mathtt{b}$

1. Subscript each symbol of the regular expression with a unique index: $\mathtt{a}_1 \mid \mathtt{b}_2(\mathtt{c}_3\mathtt{c}_4{}^* \mid (\mathtt{a}_5 \mid \mathtt{c}_6)^*)\mathtt{b}_7$.
   The indices are the states of the DFA.

Transformation of a Regular Expression to a DFA

Given a regular expression $\mathtt{a} \mid \mathtt{b}(\mathtt{cc}^* \mid (\mathtt{a} \mid \mathtt{c})^*)\mathtt{b}$

1. Subscript each symbol of the regular expression with a unique index: $\mathtt{a}_1 \mid \mathtt{b}_2(\mathtt{c}_3\mathtt{c}_4{}^* \mid (\mathtt{a}_5 \mid \mathtt{c}_6)^*)\mathtt{b}_7$.
   The indices are the states of the DFA.
2. Create an initial state by adding a state 0 at the very beginning of the expression: ${}_0(\mathtt{a}_1 \mid \mathtt{b}_2(\mathtt{c}_3\mathtt{c}_4{}^* \mid (\mathtt{a}_5 \mid \mathtt{c}_6)^*)\mathtt{b}_7)$.
3. Determine the final states by looking for states which don't require to have a symbol followed. In our example these are the states

Transformation of a Regular Expression to a DFA

Given a regular expression $\mathtt{a} \mid \mathtt{b}(\mathtt{cc}* \mid (\mathtt{a} \mid \mathtt{c})*)\mathtt{b}$

1. Subscript each symbol of the regular expression with a unique index: $\mathtt{a}_1 \mid \mathtt{b}_2(\mathtt{c}_3\mathtt{c}_4* \mid (\mathtt{a}_5 \mid \mathtt{c}_6)*)\mathtt{b}_7$.
   The indices are the states of the DFA.
2. Create an initial state by adding a state 0 at the very beginning of the expression: $_0(\mathtt{a}_1 \mid \mathtt{b}_2(\mathtt{c}_3\mathtt{c}_4* \mid (\mathtt{a}_5 \mid \mathtt{c}_6)*)\mathtt{b}_7)$.
3. Determine the final states by looking for states which don't require to have a symbol followed. In our example these are the states 1 and 7.

Transformation of a Regular Expression to a DFA

Given a regular expression $\mathtt{a} \mid \mathtt{b}(\mathtt{cc}* \mid (\mathtt{a} \mid \mathtt{c})*)\mathtt{b}$

1. Subscript each symbol of the regular expression with a unique index: $\mathtt{a}_1 \mid \mathtt{b}_2(\mathtt{c}_3\mathtt{c}_4* \mid (\mathtt{a}_5 \mid \mathtt{c}_6)*)\mathtt{b}_7$.
   The indices are the states of the DFA.
2. Create an initial state by adding a state 0 at the very beginning of the expression: $_0(\mathtt{a}_1 \mid \mathtt{b}_2(\mathtt{c}_3\mathtt{c}_4* \mid (\mathtt{a}_5 \mid \mathtt{c}_6)*)\mathtt{b}_7)$.
3. Determine the final states by looking for states which don't require to have a symbol followed. In our example these are the states 1 and 7.
4. Construct a transition matrix by documenting for each state, which symbols may follow and into which subsequent states the automaton will pass. See next slide

Example Continued

${}_0(\mathtt{a}_1 \mid \mathtt{b}_2(\mathtt{c}_3\mathtt{c}_4\mathtt{*} \mid (\mathtt{a}_5 \mid \mathtt{c}_6)\mathtt{*})\mathtt{b}_7)$.

1. The first transition from 0 is to 1 by reading an `a`. The second transition from 0 is to 2 by reading a `b`. 1 is a final state.

Example Continued

$_0(\mathtt{a}_1 \mid \mathtt{b}_2(\mathtt{c}_3\mathtt{c}_4{}^* \mid (\mathtt{a}_5 \mid \mathtt{c}_6)^*)\mathtt{b}_7)$.

1. The first transition from 0 is to 1 by reading an a. The second transition from 0 is to 2 by reading a b. 1 is a final state.
2. The transitions from 2 are to the states 3 (c), 5 (a), 6 (c), and 7 (b).

Example Continued

$_0(\mathtt{a}_1 \mid \mathtt{b}_2(\mathtt{c}_3\mathtt{c}_4{}^* \mid (\mathtt{a}_5 \mid \mathtt{c}_6)^*)\mathtt{b}_7)$.

1. The first transition from 0 is to 1 by reading an `a`. The second transition from 0 is to 2 by reading a `b`. 1 is a final state.
2. The transitions from 2 are to the states 3 (c), 5 (a), 6 (c), and 7 (b).
3. The transitions from 3 are to the states 4 (c) and 7 (b).

Example Continued

$_0(\mathtt{a}_1 \mid \mathtt{b}_2(\mathtt{c}_3\mathtt{c}_4{}^* \mid (\mathtt{a}_5 \mid \mathtt{c}_6)^*)\mathtt{b}_7)$.

1. The first transition from 0 is to 1 by reading an a. The second transition from 0 is to 2 by reading a b. 1 is a final state.
2. The transitions from 2 are to the states 3 (c), 5 (a), 6 (c), and 7 (b).
3. The transitions from 3 are to the states 4 (c) and 7 (b).
4. The transitions from 4 are to 4 (c) and to 7(b).

Example Continued

$_0(\mathtt{a}_1 \mid \mathtt{b}_2(\mathtt{c}_3\mathtt{c}_4{}^* \mid (\mathtt{a}_5 \mid \mathtt{c}_6)^*)\mathtt{b}_7)$.

1. The first transition from 0 is to 1 by reading an a. The second transition from 0 is to 2 by reading a b. 1 is a final state.
2. The transitions from 2 are to the states 3 (c), 5 (a), 6 (c), and 7 (b).
3. The transitions from 3 are to the states 4 (c) and 7 (b).
4. The transitions from 4 are to 4 (c) and to 7(b).
5. The transitions from 5 are to 5 (a), to 6 (c), and to 7 (b).

Example Continued

$_0(\mathtt{a}_1 \mid \mathtt{b}_2(\mathtt{c}_3\mathtt{c}_4{}^* \mid (\mathtt{a}_5 \mid \mathtt{c}_6)^*)\mathtt{b}_7)$.

1. The first transition from 0 is to 1 by reading an a. The second transition from 0 is to 2 by reading a b. 1 is a final state.
2. The transitions from 2 are to the states 3 (c), 5 (a), 6 (c), and 7 (b).
3. The transitions from 3 are to the states 4 (c) and 7 (b).
4. The transitions from 4 are to 4 (c) and to 7(b).
5. The transitions from 5 are to 5 (a), to 6 (c), and to 7 (b).
6. The transitions from 6 are to 6 (c) to 5 (a), and to 7 (b).

Example Continued

${}_0(\mathtt{a}_1 \mid \mathtt{b}_2(\mathtt{c}_3\mathtt{c}_4{}^* \mid (\mathtt{a}_5 \mid \mathtt{c}_6)^*)\mathtt{b}_7)$.

1. The first transition from 0 is to 1 by reading an a. The second transition from 0 is to 2 by reading a b. 1 is a final state.
2. The transitions from 2 are to the states 3 (c), 5 (a), 6 (c), and 7 (b).
3. The transitions from 3 are to the states 4 (c) and 7 (b).
4. The transitions from 4 are to 4 (c) and to 7(b).
5. The transitions from 5 are to 5 (a), to 6 (c), and to 7 (b).
6. The transitions from 6 are to 6 (c) to 5 (a), and to 7 (b).
7. 7 is a final state.