Regular Languages
Discrete Mathematic

Anders Kalhauge

Fall 2017

Set Theory

Outline

Computing

Formal languages

- Regular expressions
- RegEx
- Finite State Automata
- Non-regular Languages

Computing

Formal languages

- Regular expressions
- RegEx
- Finite State Automata
- Non-regular Languages

- Logic - the foundations of mathematics
- Electrical engineering - the design of switching circuits
- Brain research - models of neurons
- Linguistic - the formal specification of languages

A function on the natural numbers is computable by a human being following an algorithm, ignoring resource limitations, if and only if it is computable by a Turing machine

Chomsky's language specifications

- Regular languages
  - Regular expressions
  - Pattern matching
  - *Finite-state automaton*
  - Not Turing-complete
- Context-free languages
  - Backus-Naur notation
  - *Push-down automaton*
  - Turing-complete

Computing

Formal languages

- Regular expressions
- RegEx
- Finite State Automata
- Non-regular Languages

- Lexical scanner
- Syntactic Analyser
- Code generator

Alphabet Σ a finite set of characters

String over Σ Either a finite sequence of characters in Σ or the empty (null) string ϵ

Length of a string over Σ The number of characters in the string, the length of ϵ is 0

Formal language over Σ a set of strings over Σ

$\emptyset$ is a formal language
who nothing does, does nothing wrong

Alphabets and strings

- Σ is an alphabet
- Σ^n is all strings over Σ with the lenght n
- Σ^+ is all non-empty strings over Σ
- Σ^* is all strings over Σ - the Kleene closure of Σ

Combinations of Languages

- $L\,L'$: **Concatenation** of L and L'

$$L\,L' = \{x\,y \,|\, x \in L \wedge y \in L'\}$$

- $L \cup L'$: **Union** of L and L'

$$L \cup L' = \{x \,|\, x \in L \vee x \in L'\}$$

- L^*: Kleene closure of L

$$L^* = \{x \,|\, \text{ is a concatenation of strings in } L\}$$

The following are regular expressions over Σ:

Base $\emptyset$, ϵ, $x \mid x \in \Sigma$

Recursion $r, s \in$ regular expressions over $\Sigma \rightarrow$

- (rs) - r concatenated with s
- $(r \mid s)$ - r or s
- (r^*) - $r^* \in \{\epsilon, r, rr, rrr, \dots\}$[1]

Restriction Nothing else is a regular expression

Some syntactic sugar:

1. If "(", "|", ")", or "$*$" are members of the alpfabet Σ, an escape character as "\" can be used.
2. The order of precedence, "$*$", concatenation, and "|" can be used to remove parantheses
3. Outer parenthesis can be removed

[1] the Kleene closure of r

Exercise

- Define the alphabet to express an ISO-month (yyyy-mm)
- Define the regular expression for the language of ISO-months

Exercise

- Define the alphabet to express an ISO-month (yyyy-mm)
  - $\{0, 1, 2, \ldots, 9, -\}$
- Define the regular expression for the language of ISO-months

Exercise

- Define the alphabet to express an ISO-month (yyyy-mm)
  - $\{0, 1, 2, \ldots, 9, -\}$
- Define the regular expression for the language of ISO-months
  - year: $(0|1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)\ldots$ (four times)
  - month: $(0)(1|2|3|4|5|6|7|8|9)|(1)(0|1|2)$
  - date: $< date > - < month >$

defined by regular expressions

$L(r)$ is the language defined by r

Base
- $L(\emptyset) = \emptyset$
- $L(\epsilon) = \{\epsilon\}$
- $\forall a \in \Sigma, L(a) = \{a\}$

Recursion
- $L(r\,r') = L(r)L(r')$
- $L(r \,|\, r') = L(r) \cup L(r')$
- $L(r^*) = (L(r))^*$

What is $\{a, b, z\}^*$?

What is $\{a, b, z\}^*$?

$$\{\epsilon, a, b, z, aa, ab, az, ba, bb, bz, za, zb, zz, aaa, aab \ldots\}$$

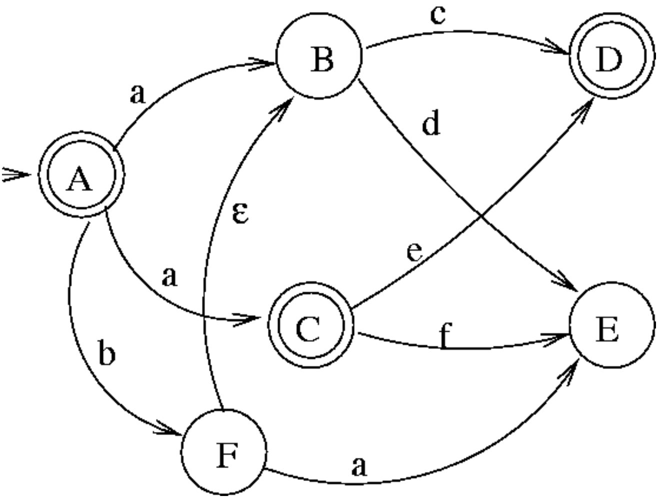
What is $\{a, b, z\}^*$?

$$\{\epsilon, a, b, z, aa, ab, az, ba, bb, bz, za, zb, zz, aaa, aab \dots\}$$

Is $\{a, b, z\}^*$ countable?

- `abc` same as abc
- `.` same as any character $x \mid x \in \Sigma$
- `a*` same as $a*$
- `a+` same as $aa*$
- `[abc]` same as $a|b|c$
- `a|b|c` same as $a|b|c$
- `[0-9]` same as $0|1|2|3|4|5|6|7|8|9$
- `[pq0-9a-d]` same as $p|q|0|1|2|3|4|5|6|7|8|9|a|b|c|d$
- `[^xyz]` same as $\Sigma - \{x, y, z\}$
- `a{4}` same as $aaaa$
- `[0-9]{4}-(0[1-9])|(1[012])` same as ?

Diagram



Finite-state Automaton

Definition

A finite-state automaton consists of:

1. a finite **input alphabet** I of input symbols
2. a finite set of **states** S
3. an **initial state** s_0, $s_0 \in S$
4. a set of **final states**[2] F
5. a **next-state function** $N : S \times I \to S$

[2] accepting states

Languages accepted by an Automaton

- A is a finite-state automaton with input alphabet I
- The string w over I, $w \in I^*$
- The symbols in w brings A from its initial state s_0 to a final state $s_f \in F$
- $L(A)$ is the language accepted by A.
- $L(A) = \{w \in I^* \mid w \text{ is accepted by } A\}$

The Eventual-State Function

- $N : S \times I \to S$ is the next-state function
  $N(s, m)$ gives the next state of A if A was in the state s, given the symbol $m \in I$
- $N^* : S \times I^* \to S$ is the eventual-state function
  $N^*(s, w)$ gives the state to which A goes if the symbols of w are input to A in sequence, starting when A is in state s

$$w \text{ is accepted by } A \iff N^*(s_0, w) \in F$$

$$L(A) = \{w \in I^* \mid N^*(s_0, w) \in F\}$$

F is the final (or accepting) states of A

Regular languages:

can be defined by a regular expression

$$\Updownarrow$$

can be accepted by a finite-state automata

Regular languages:

can be defined by a regular expression

$$\Updownarrow$$

can be accepted by a finite-state automata

but not all languages are regular. . .

A non-regular language

- L is the language over the alphabet $\Sigma = \{a, b\}$
- $L = \{w \in \Sigma^*, k \in \mathbb{N} \,|\, w = a^k b^k\}$
- $a7 = aaaaaaa$ as an example of a^k
- $L = \{ab, aabb, aaabbb, aaaabbbb, \dots\}$
- A is a finite-state automaton, it has a finite number n of states
- Choosing a $k > n$ must bring A to a state s_m already visited by $h < k$ a's
- $\therefore N^*(s_0, a^h b^k) \in F, h \neq k$ and $a^h b^k \notin L$