Mathematics for Computing Science (CS2013)

Regular Expressions

Regular expressions

- Goal: specify languages in a compact fashion
    - The specification should be usable by computer systems
- FSA as mean to specify languages not ideal
    - Formal description: difficult to write, lengthy and error-prone
    - Graphical format: easy to humans to follow, hard to process by computers
- We need a simpler **notation** to describe unambiguously a language
    - Text-like (not graphic…) and processed like text
    - With a very small set of operations/conventions
- Constructs like
    - “zero or more occurrences of a symbol” followed by
    - “at least one occurrence of a symbol” followed by
    - “one or more occurrences of a or b”
- We have one such notation: **regular expressions** (**REs**)

Formal definition of regular expressions

- A **regular expression** over $T = \{a_1, a_2, \ldots, a_m\}$ is any string of the form
  i. λ (lambda)
  ii. $\varnothing$ (empty set)
  iii. If $a_i \in T$ then $\boldsymbol{a}_i$ is a regular expression (symbol in boldface)
  iv. If R and E are regular expressions then $(R+E)$ is a regular expression
  v. If R and E are regular expressions then (RE) is a regular expression
  vi. If R and E are regular expressions then (R^*) is a regular expression
- The **syntax** of REs is as above – the "semantics" will be next
  - Definition is **inductive** (cases iv-vi) – we can go on an on
- Example REs from $T = \{0, 1\}$:
  - $(\mathbf{0})$ and $(\mathbf{1})$ are REs; $((\mathbf{0})+(\mathbf{1}))$ is an RE (case iv)
  - $((\mathbf{0})+(\mathbf{1}))$ and $(\mathbf{0})$ are REs; $(((\mathbf{0})+(\mathbf{1}))(\mathbf{0}))$ is an RE (case v)
  - $((\mathbf{0})+(\mathbf{1}))$ is an RE so $(((\mathbf{0})+(\mathbf{1}))^*)$ is an RE (case vi)

Formal definition of regular expressions (cont'd)

- What do regular expressions mean? What's their "semantics"?
    - In mathematics, the meaning of (2+1)5 is (3)5 = 3×5 = 15
- Function S (for semantics) mapping REs to a language $L \subseteq T^*$
    i. $S(\lambda) = \{\lambda\}$
    ii. $S(\varnothing) = \varnothing$
    iii. $S(\boldsymbol{a}_i) = \{\boldsymbol{a}_i\}$
    iv. $S((R+E)) = S(R) \cup S(E)$ (union of languages $S(R)$ and $S(E)$)
    v. $S(RE) = S(R)S(E)$ (concatenation of strings from $S(R)$ with strings from $S(E)$)
    vi. $S(R^*) = S(R)^*$ (strings from $S(R) \cup \{\lambda\}$ concatenated any number of times)
- N.B.: Concatenation and star operations defined in previous set of slides

Formal definition of regular expressions (cont'd)

- Example REs from $T = \{0, 1\}$:
  - $S((\mathbf{0})) = \{0\}$
  - $S((\mathbf{1})) = \{1\}$
  - $S(((\mathbf{0})+(\mathbf{1}))) = S((\mathbf{0})) \cup S((\mathbf{1})) = \{0\} \cup \{1\} = \{0, 1\}$
  - $S((((\mathbf{0})+(\mathbf{1}))(\mathbf{0}))) = S(((\mathbf{0})+(\mathbf{1})))S((\mathbf{0})) = \{0, 1\}\ \{0\} = \{00, 10\}$
  - $S((((\mathbf{0})+(\mathbf{1}))^*)) = S(((\mathbf{0})+(\mathbf{1})))^* = \{0, 1\}^* = \{\lambda, 0, 1, 00, 100, 000, \ldots\}$
- Parentheses can be omitted if we assume **precedence**
  - Star operator * has precedence over concatenation
  - Concatenation has precedence over union +
  - $(((\mathbf{0})+(\mathbf{1}))^*)$ becomes $(\mathbf{0}+\mathbf{1})^*$
  - $(((\mathbf{1})^*)((\mathbf{0})+(\mathbf{1})))$ becomes $\mathbf{1}^*\mathbf{0}+\mathbf{1}\ \mathbf{1}$

Regular languages

- $L \subseteq T^*$ is a **regular language** if there is a regular expression R such that $S(R) = L$

Computing regular expressions

- What's the language $L \subseteq \{a, b, c, d\}^*$ of the RE $\mathbf{a}^*(\mathbf{b}+\mathbf{c}+\mathbf{d})$?
  - $S(\mathbf{a}^*(\mathbf{b}+\mathbf{c}+\mathbf{d})) = S(\mathbf{a}^*)S(\mathbf{b}+\mathbf{c}+\mathbf{d}) = \{a\}^*(\{b\} \cup \{c\} \cup \{d\}) = \{a\}^*\{b, c, d\}$
  - $S(\mathbf{a}^*(\mathbf{b}+\mathbf{c}+\mathbf{d})) = \{\lambda, a, aa, aaa, aaaa, \ldots\} \{b, c, d\}$
  - $S(\mathbf{a}^*(\mathbf{b}+\mathbf{c}+\mathbf{d})) = \{\lambda b, \lambda c, \lambda d, ab, ac, ad, aab, aac, aad, aaab, aaac, aaad, \ldots\}$
  - $S(\mathbf{a}^*(\mathbf{b}+\mathbf{c}+\mathbf{d})) = \{b, c, d, ab, ac, ad, aab, aac, aad, aaab, aaac, aaad, \ldots\}$
  - N.B.: $\lambda w = w\lambda = w$
- In English:
  - All words beginning with zero or more occurrences of symbol a followed by exactly one occurrence of symbol b, c, or symbol d

Computing regular expressions (cont'd)

- What's the language $L \subseteq \{a, b\}^*$ of the RE $(\mathbf{a}+\mathbf{b})^*\mathbf{a}$?
  - $S((\mathbf{a}+\mathbf{b})^*\mathbf{a}) = S((\mathbf{a}+\mathbf{b})^*)S(\mathbf{a}) = (\{a\} \cup \{b\})^*\{a\}$
  - $S((\mathbf{a}+\mathbf{b})^*\mathbf{a}) = \{a, b\}^*\{a\}$
  - $S((\mathbf{a}+\mathbf{b})^*\mathbf{a}) = \{\lambda, a, b, ab, ba, aa, bb, \ldots\}\{a\} = \{\lambda a, aa, ba, aba, baa, aaa, bba, \ldots\}$
  - $S((\mathbf{a}+\mathbf{b})^*\mathbf{a}) = \{a, aa, ba, aba, baa, aaa, bba, \ldots\}$
- In English:
  - All words beginning with zero or more occurrences of symbols a or b ending with exactly one occurrence of symbol a
- N.B.: $\lambda \notin S((\mathbf{a}+\mathbf{b})^*\mathbf{a})$

Computing regular expressions (cont'd)

- What's the language $L \subseteq \{a, b\}^*$ of the RE $(\mathbf{a}+\mathbf{b})^*(\mathbf{a}+\mathbf{b})$?
  - $S((\mathbf{a}+\mathbf{b})^*(\mathbf{a}+\mathbf{b})) = S((\mathbf{a}+\mathbf{b})^*)S((\mathbf{a}+\mathbf{b})) = (\{a\} \cup \{b\})^*(\{a\} \cup \{b\})$
  - $S((\mathbf{a}+\mathbf{b})^*(\mathbf{a}+\mathbf{b})) = \{a, b\}^*\{a, b\} = \{\lambda, a, b, ab, ba, aa, bb, \ldots\}\{a, b\}$
  - $S((\mathbf{a}+\mathbf{b})^*(\mathbf{a}+\mathbf{b})) = \{\lambda a, \lambda b, aa, ab, ba, bb, aaa, aab, \ldots\}$
  - $S((\mathbf{a}+\mathbf{b})^*(\mathbf{a}+\mathbf{b})) = \{a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$
- In English:
  - All strings of a's and b's, with at least one character/symbol a or b

Computing regular expressions (cont'd)

- What's the language $L \subseteq \{a, b\}^*$ of the RE $(\mathbf{a}+\mathbf{bb})(\mathbf{ab})^*$?
  - $S((\mathbf{a}+\mathbf{bb})(\mathbf{ab})^*) = S(\mathbf{a}+\mathbf{bb})S(\mathbf{ab})^* = (S(\mathbf{a}) \cup S(\mathbf{bb}))(S(\mathbf{ab}))^*$
  - $S((\mathbf{a}+\mathbf{bb})(\mathbf{ab})^*) = (\{a\} \cup (S(\mathbf{b})S(\mathbf{b})))(S(\mathbf{a})S(\mathbf{b}))^*$
  - $S((\mathbf{a}+\mathbf{bb})(\mathbf{ab})^*) = (\{a\} \cup (\{b\}\{b\}))(\{a\}\{b\})^*$
  - $S((\mathbf{a}+\mathbf{bb})(\mathbf{ab})^*) = (\{a\} \cup \{bb\})(\{ab\})^*$
  - $S((\mathbf{a}+\mathbf{bb})(\mathbf{ab})^*) = \{a, bb\}(\{ab\})^*$
  - $S((\mathbf{a}+\mathbf{bb})(\mathbf{ab})^*) = \{a, bb\}\{\lambda, ab, abab, ababab, abababab, \ldots\}$
  - $S((\mathbf{a}+\mathbf{bb})(\mathbf{ab})^*) = \{a\lambda, bb\lambda, aab, bbab, aabab, bbabab, aababab, bbababab, \ldots\}$
  - $S((\mathbf{a}+\mathbf{bb})(\mathbf{ab})^*) = \{a, bb, aab, bbab, aabab, bbabab, aababab, bbababab, \ldots\}$
- In English:
  - All strings of pairs of ab's starting with a or bb

Designing regular expressions

- We can put our definitions to work by “reversing” the problem
  - Provide a regular expression to capture a language
- Provide an RE for $L \subseteq \{a, b\}^*$ with exactly one occurrence of symbol a:
  - Solution: $\mathbf{b}^*\mathbf{ab}^*$
  - Need persuasion? Compute $S(\mathbf{b}^*\mathbf{ab}^*)$!
- Provide an RE for $L \subseteq \{0, 1\}^*$ with substring 00:
  - Solution: $(\mathbf{0}+\mathbf{1})^*\mathbf{00}(\mathbf{0}+\mathbf{1})^*$
  - Need persuasion? Compute $S((\mathbf{0}+\mathbf{1})^*\mathbf{00}(\mathbf{0}+\mathbf{1})^*)$!
- Provide an RE for $L \subseteq \{0, 1\}^*$ with substring 00 or 111:
  - Solution: $(\mathbf{0}+\mathbf{1})^*(\mathbf{00}+\mathbf{111})(\mathbf{0}+\mathbf{1})^*$
  - Need persuasion? Compute $S((\mathbf{0}+\mathbf{1})^*(\mathbf{00}+\mathbf{111})(\mathbf{0}+\mathbf{1})^*)$!

Designing regular expressions

- Provide an RE for $L \subseteq \{0, 1\}^*$ for all strings with one 1 and two 0s:
  - Solution: **100+001+010**
  - We have to specify all cases
- Provide an RE for $L \subseteq \{0, 1\}^*$ for all strings with the same number of 0s and 1s:
  - Solution: (**01**)*? No, because 0011 not represented… (and it should)
  - Solution: (**0**+**1**)*? No, because 00 represented (and it shouldn't)
  - There is no RE for this language…
  - This is not a coincidence!
- Regular expressions and FSAs are closely connected
  - They have the same expressive power

Properties of regular languages

- If L_1 and L_2 are regular languages then the following are also regular languages:
  - $L_1 \cup L_2$ – the union of regular languages is a regular language
  - $L_1 \cap L_2$ – the intersection of regular languages is a regular language
  - $\overline{L_1}$ – the complement of a regular language is a regular language
  - $L_1 L_2$ – the concatenation of regular languages is a regular language
  - ${L_1}^*$ – the Kleene closure of a regular language is a regular language
- Any finite language is regular

Further notation

- We use R^+ as a shorthand for RR^*
  - $S(R^+) = S(R)^+$
- We use R^n as a shorthand for $RR\ldots RR$ (n times)
  - $S(R^n) = S(R)S(R) \ldots S(R)S(R)$
- These do not extend expressiveness of REs

Interesting/useful results

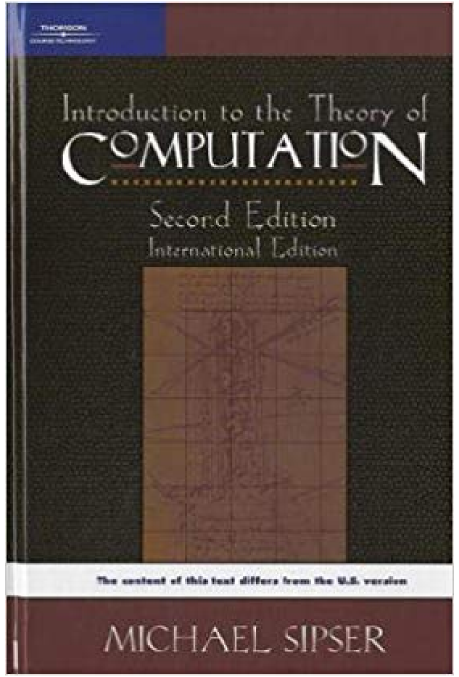
- $\varnothing \neq \{\lambda\}$ – empty set **not the same** as set with empty string
- $\varnothing^* = \{\lambda\}$ – Kleene closure of the empty set is set with empty string
- $\lambda^* = \{\lambda\}$ – Kleene closure of empty string is set with empty string
- $\varnothing R = R\varnothing = \varnothing$ – empty set concatenated with any regular expression R is the empty set

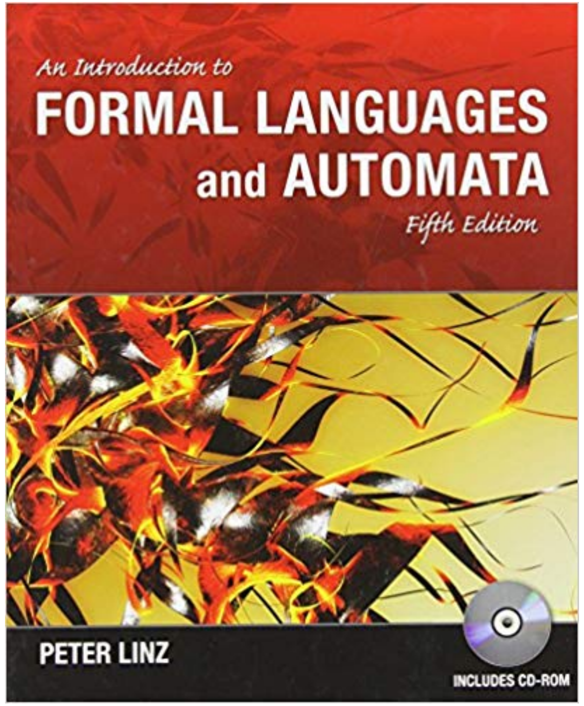
Summary

- Regular expressions: syntax and meaning
- How to compute the language of a regular expression
- How to design a regular expression to represent a language

Further reading

- Chapter 1 of "Introduction to the theory of computation", by Michael Sipser (there are copies in the library)
- Chapter 3 of "An Introduction to Formal Languages and Automata", by Peter Linz (PDF available on-line)





Summary

- Formal definition of FSA
- How definition enable us to check if a string belongs to a language
- Graphical representation of FSA
- Non-determinism vs. determinism
- Important results (theorems)