Scanning

Compiler Construction

2024-10-22

Prof. Dr. Michael Kuhn
michael.kuhn@ovgu.de

Parallel Computing and I/O
Institute for Intelligent Cooperating Systems
Faculty of Computer Science
Otto von Guericke University Magdeburg
https://parcio.ovgu.de

Outline

Regular Expressions

- A regular expression s is a string that defines $L(s)$ for an alphabet Σ
  - $L(a) = \{a\} \quad (\forall a \in \Sigma)$
  - $L(\epsilon) = \{\epsilon\}$
  - $L(s|t) = L(s) \cup L(t)$
  - $L(st) = L(s) \cdot L(t)$ (concatenation of every combination)
  - $L(s^*) = L(s)^{0..n}$ (zero or more concatenations)

Group Exercise

- Write regular expressions for German weekdays.

Group Exercise

- Write regular expressions for German weekdays.
  - `Montag|Dienstag|Mittwoch|Donnerstag|Freitag|Samstag|Sonntag`

Group Exercise

- Write regular expressions for German weekdays.
  - `Montag|Dienstag|Mittwoch|Donnerstag|Freitag|Samstag|Sonntag`
  - `(Mon|Diens|Donners|Frei|Sams|Sonn)tag|Mittwoch`

Group Exercise

- Write regular expressions for German weekdays.
  - `Montag|Dienstag|Mittwoch|Donnerstag|Freitag|Samstag|Sonntag`
  - `(Mon|Diens|Donners|Frei|Sams|Sonn)tag|Mittwoch`
  - `((Mon|Frei|Sonn)|(Dien|Donner|Sam)s)tag|Mittwoch`

Group Exercise

- Write regular expressions for integers with thousands separators.

Group Exercise

- Write regular expressions for integers with thousands separators.
  - `[0-9]?[0-9]?[0-9](.[0-9][0-9][0-9])*`

Group Exercise

- Write regular expressions for integers with thousands separators and a decimal part.

Group Exercise

- Write regular expressions for integers with thousands separators and a decimal part.
  - `[0-9]?[0-9]?[0-9](.[0-9][0-9][0-9])*(,[0-9]+)?`

Group Exercise

- Write regular expressions for strings of X symbols with single pairs of <> and {}.

Group Exercise

- Write regular expressions for strings of X symbols with single pairs of <> and {}.
  - `X*<X*{X*}X*>X*|X*{X*<X*>X*}X*|X*<X*>X*{X*}X*|X*{X*}X*<X*>X*`

Limitations

- `X*(<|{)X*(>|})X*` – Why is this wrong?

Limitations

- `X*(<|{)X*(>|})X*` – Why is this wrong?
- `X*<X*>X*|X*{X*}X*` – Simplified form with only one pair.

Limitations

- `X*(<|{)X*(>|})X*` – Why is this wrong?
- `X*<X*>X*|X*{X*}X*` – Simplified form with only one pair.
- `(X*<X*)+(X*>X*)+|(X*{X*)+(X*}X*)+` – Why is this wrong?

Limitations...

- How about multiple pairs of <> and {}?
  - `(X*<X*{X*}X*>X*|X*{X*<X*>X*}X*|X*<X*>X*{X*}X*|X*{X*}X*<X*>X*)+` – Why is this wrong?

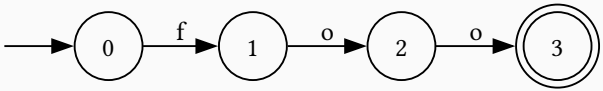
Limitations...

- How about multiple pairs of <> and {}?
  - `(X*<X*{X*}X*>X*|X*{X*<X*>X*}X*|X*<X*>X*{X*}X*|X*{X*}X*<X*>X*)+` – Why is this wrong?
  - `((X*<X*>X*)*(X*{X*}X*)*)*|X*<X*{X*}X*>X*|X*{X*<X*>X*}X*|...`

Outline

Finite Automata

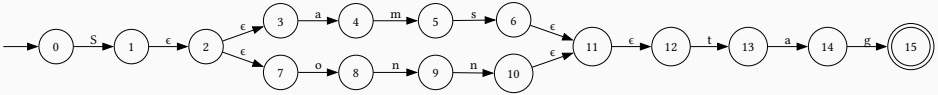
- A finite automaton can be used to represent regular expressions
  - States and edges with one starting state and potentially multiple accepting states
  - Edges are labeled with symbols from Σ
  - Automata can be deterministic or non-deterministic
  - FA for the regular expression: `foo`



Group Exercise

- Convert `S(ams|onn)tag` into an NFA.

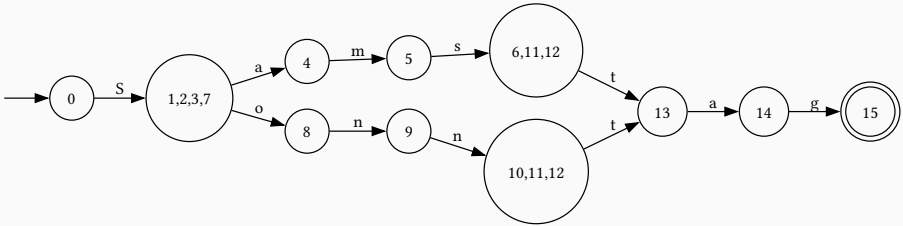
- Convert `S(ams|onn)tag` into an NFA.



Group Exercise

- Convert the previous NFA into a DFA.

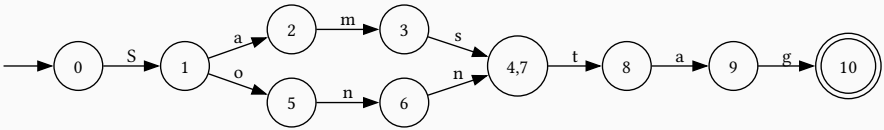
- Convert the previous NFA into a DFA.



Group Exercise

- Minimize the previous DFA.

- Minimize the previous DFA.



Outline

- Regular expressions are relatively easy but limited
  - Because regular languages are limited and cannot describe all necessary structures
- Finite automata can be used to visualize regular expressions
  - They can be deterministic or non-deterministic

References

[Thain, 2020] Thain, D. (2020). ***Introduction to Compilers and Language Design: Second Edition.*** http://compilerbook.org/.