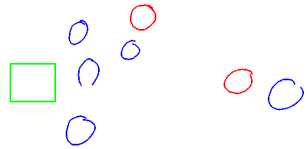
Script **generated by TTT**

Title: Petter: Compiler Construction (30.04.2020) - 06: Powerset Construction by Need

Date: Wed Apr 22 16:03:08 CEST 2020

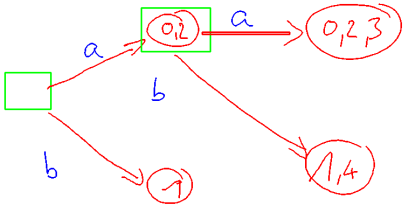
Duration: 18:45 min

Pages: 6



Chapter 4:

Turning NFAs deterministic

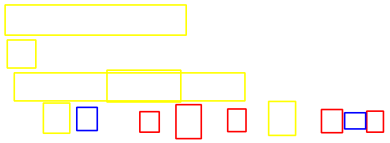


Powerset Construction

Theorem:

For every non-deterministic automaton $A = (Q, \Sigma, \delta, I, F)$ we can compute a deterministic automaton $\mathcal{P}(A)$ with

$$\mathcal{L}(A) = \mathcal{L}(\mathcal{P}(A))$$



Powerset Construction

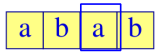
Observation:

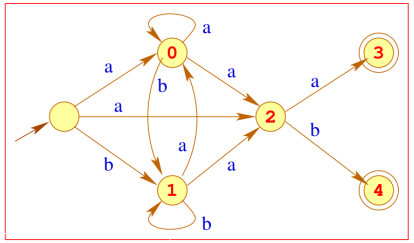
There are exponentially many powersets of Q

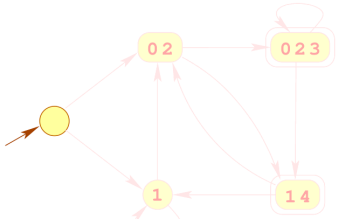
- Idea: Consider only contributing powersets. Starting with the set $Q_{\mathcal{P}} = \{I\}$ we only add further states by need ...
- i.e., whenever we can reach them from a state in $Q_{\mathcal{P}}$
- However, the resulting automaton can become enormously huge ... which is (sort of) not happening in practice

Powerset Construction

... for example:







Remarks:

- For an input sequence of length n, maximally $\mathcal{O}(n)$ sets are generated
- Once a set/edge of the DFA is generated, they are stored within a hash-table.
- Before generating a new transition, we check this table for already existing edges with the desired label.

Remarks:

- For an input sequence of length n, maximally $\mathcal{O}(n)$ sets are generated
- Once a set/edge of the DFA is generated, they are stored within a hash-table.
- Before generating a new transition, we check this table for already existing edges with the desired label.

Summary:

Theorem:

For each regular expression e we can compute a deterministic automaton $A = \mathcal{P}(A_e)$ with

$$\mathcal{L}(A) = [\![e]\!]$$