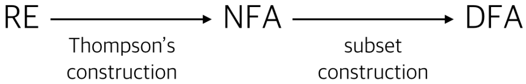
COSE312: Compilers

Lecture 4 — Lexical Analysis (3)

Hakjoo Oh
2017 Spring

Part 3: Automation

Transform the lexical specification into an executable string recognizers:



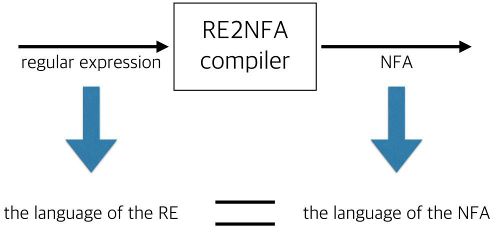
RE → NFA → DFA

Thompson's construction

subset construction

From REs to NFAs

Transform a given regular expression into a semantically equivalent NFAs:



An instance of "compilation":

- The source language is regular expressions and the target language is NFAs.
- The correctness is defined by the equivalence of the denoted languages.

Principles of Compilation

Every automatic compilation

1. is done “compositionally”, and
2. maintains some “invariants” during compilation.

Compilation of regular expressions, e.g., $R_1|R_2$:

1. The compilation of $R_1|R_2$ is defined in terms of the compilation of R_1 and R_2.
2. Compiled NFAs for R_1 and R_2 satisfy the invariants:
    - an NFA has exactly only one accepting state,
    - no arcs into the initial state, and
    - no arcs out of the accepting state.

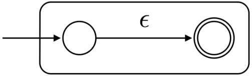
The Source Language

$$\begin{array}{rcl} R & \rightarrow & \emptyset \\ & \mid & \epsilon \\ & \mid & a \in \Sigma \\ & \mid & R_1 \mid R_2 \\ & \mid & R_1 \cdot R_2 \\ & \mid & R_1^* \\ & \mid & (R) \end{array}$$

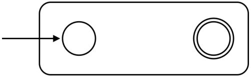
Compilation

Base cases:

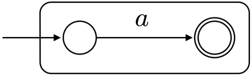
- $R = \epsilon$:



- $R = \emptyset$



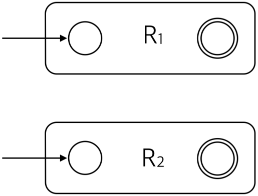
- $R = a\ (\in \Sigma)$



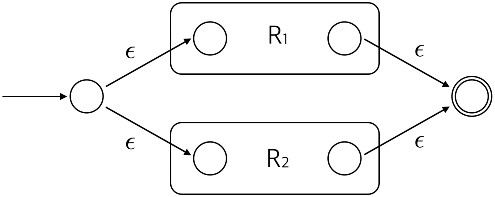
Compilation

Inductive cases:

- $R = R_1 | R_2$:
  1. Compile R_1 and R_2:

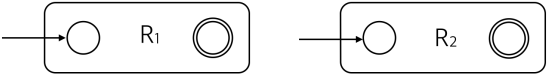


  2. Compile $R_1 | R_2$ using the results:

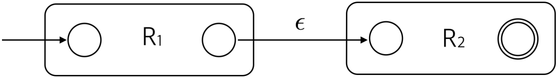


Compilation

- $R = R_1 \cdot R_2$:
  1. Compile R_1 and R_2:

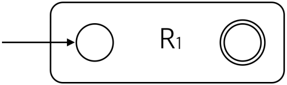


  2. Compile $R_1 \cdot R_2$ using the results:

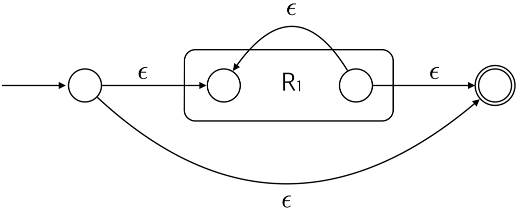


Compilation

- $R = R_1^*$:
  1. Compile R_1:



  2. Compile R_1^* using the results:



Examples

- $\mathbf{0} \cdot \mathbf{1}^*$:

- $(\mathbf{0}|\mathbf{1}) \cdot \mathbf{0} \cdot \mathbf{1}$:

- $(\mathbf{0}|\mathbf{1})^* \cdot \mathbf{1} \cdot (\mathbf{0}|\mathbf{1})$: