richardwu.ca

CS 444/644 Course Notes

Compiler Construction

Ondřej Lhoták • Winter 2019 • University of Waterloo

Last Revision: January 9, 2019

Table of Contents

Abstract

These notes are intended as a resource for myself; past, present, or future students of this course, and anyone interested in the material. The goal is to provide an end-to-end resource that covers all material discussed in the course displayed in an organized manner. These notes are my interpretation and transcription of the content covered in lectures. The instructor has not verified or confirmed the accuracy of these notes, and any discrepancies, misunderstandings, typos, etc. as these notes relate to course's content is not the responsibility of the instructor. If you spot any errors or would like to contribute, please contact me directly.

1 January 7, 2019

1.1 Basic overview of a compiler

A **compiler** takes a source language and translates it to a target language. The source language could be, for example, C, Java, or JVM bytecode and the target language could be, for example, machine language or JVM bytecode.
A compiler could be divided into two parts:

Front-end analysis Front-end could be further divided into two parts: the first being scanning and parsing (assignment 1) and the second being context-sensitive analysis (assignment 2,3,4).

Some refer to context-sensitive analysis as "middle-end".

Back-end synthesis The backend could also be divided into two parts: the first being optimization (CS 744) and the second being code generation (assignment 5).

1.2 Overview of front-end analysis

Goal: is the input a valid program? An auxiliary step is to also generate information about the program for use in synthesis later on.
There are several steps in the front-end:

Scanning Split sequence of characters into sequence of tokens. Each token consists of its lexeme (actual characters) and its kind.

There are tools for generating the DFA expressions from a regular language e.g. lex.

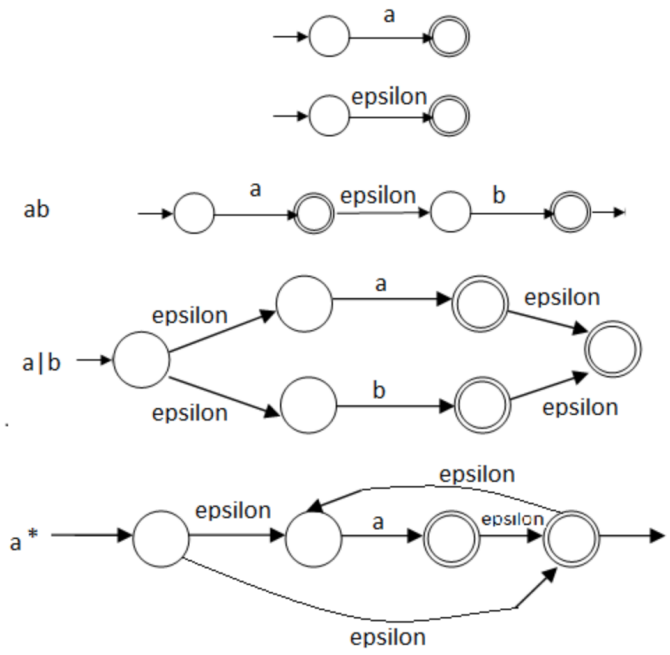
2 January 9, 2019

2.1 Scanning tools (lex)

Our goal is to specify our grammar in terms of regular expressions (regex) which lex can convert into a scanning DFA.
Review of regex to language (set of words):

RE	$L(e)$
$\varnothing$	$\{\}$
ϵ	$\{\epsilon\}$
$a \in \Sigma$	$\{a\}$
$e_1 e_2$	$\{xy \mid x \in L(e_1), y \in L(e_2)\}$
$e_1 \mid e_2$	$L(e_1) \cup L(e_2)$
e^*	$L(\epsilon \mid e \mid ee \mid eee \mid \ldots)$

The corresponding NFAs we can construct per each regex rule are



Let Σ be the set of characters, Q the set of states, q_0 the initial state, A the accepting states, and δ the transition function, an NFA is a 5-tuple $(\Sigma, Q, q_0, A, \delta)$ where

$$\text{NFA} : \delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q \text{ (subset of Q)}$$
$$\text{DFA} : \delta : Q \times \Sigma \to Q$$

i.e. the transition function of an NFA returns a subset of states in Q.
We note in our above NFAs some accepting states are equivalent (connected by an ϵ transition).
We define

Definition 2.1 (ϵ-closure I)**.** The ϵ-closure(S) of a set of states S is the set of states reachable from S by (0 or more) ϵ-transitions.

Another equivalent recursive definition

Definition 2.2 (ϵ-closure II)**.** Smallest set S' such that

$$S' \supseteq S$$
$$S' \supseteq \{q \mid q' \in S', q \in \delta(q', \epsilon)\}$$

We note that any NFA can be converted in a corresponding DFA. The input is an NFA $(\Sigma, Q, q_0, A, \delta)$ and we'd like to get a DFA $(\Sigma, Q', q_0', A', \delta')$ where

$$q_0' = \epsilon\text{-closure}(\{q_0\})$$
$$\delta'(q', a) = \epsilon\text{-closure}\Big(\bigcup_{q \in q'} \delta(q, a)\Big)$$

we note that each state of our DFA is a set of states in the original NFA e.g. $\{1, 2, 4\}$ may be a state in the DFA.

We generate more states in our DFA by iterating through every $a \in \Sigma$ and applying rule two to our initial state q'_0. We do this until no further states can be generated from existing DFA states and all $a \in \Sigma$ have been exhausted. We note that the entire set of states Q' in the DFA can be recursively defined as the smallest set of subsets of Q such that

$$Q' \supseteq \{q'_0\}$$
$$Q' \supseteq \{\delta'(q', a) \mid q' \in Q'\} \qquad \forall a \in \Sigma$$

We note that if any accepting state is included in a state of the DFA, we can accept it (since we can reach the corresponding accepting state in the NFA). Thus

$$A' = \{q' \in Q' \mid q' \cap A \neq \varnothing\}$$