Implementation of a compiler for an imperative language IMP

Remy Detobel & Denis Hoornaert

October 7, 2017

Contents

1 Introduction

The project aim is to implement a compiler for a 'simple' imperative language named *IMP*. Like any imperative programming language, *IMP* is structured of mainstream features such as *keywords* (`if`, `while`, ... statements), the use of *variables*, the use *numbers* and the use of *comments*. The form of these features follows some defined rules :

- a *variable* is a sequence of alphanumeric characters that must start by a letter.
- a *number* is a sequence of one or more digits.
- a *comment* must start by the combination `(*` and ends by the reversed combination `*)`.

The compilation scheme is generally divided in three main phases : analysis, synthesis and optimization. The phases are themselves composes of different steps. For instance, the analysis phase is composed of *lexical analysing* step (or *scanning*), a *syntax analysing* step (or *parsing*) and a *semantic analysing* step. In this assignment, the focus is set on the *analysis phase*.

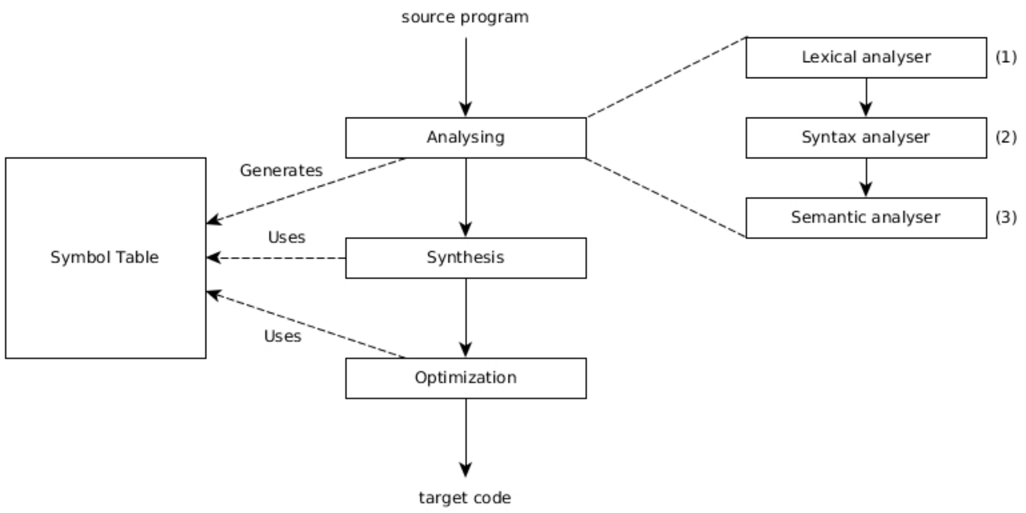

Figure 1 - Compilation phases.

2 Implementation of the lexical analyser

In the so called "Dragon book"[1] the *lexical anlyser* is defined as follow :

[1] V. Aho, A., 2007. *Compilers : Principles, techniques, & Tools.* 2nd ed. New York: Pearson.