National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS 354: Compiler Construction

Class: BSCS-5AB

Lab 10: Introduction to Syntax Directed Translation (SDT)

Date: 13th December, 2018

Time: [09:00am – 12:00pm/2:00 pm – 5:00 pm]

Instructor: Dr. Rabia Irfan

Lab Engineer: Mr. Azaz Farooq

National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Lab 10: Introduction to Syntax Directed Translation (SDT)

Introduction

Syntax-directed translation (SDT) refers to a method of compiler implementation where the source language translation is completely driven by the parser, i.e., based on the syntax of the language The parsing process and parse trees are used to direct semantic analysis and the translation of the source program. Almost all modern compilers are syntax-directed.

Objectives

Successful understanding/implementation of SDT using Bison

Tools/Software Requirement

gcc, g++, Flex and Bison

Description

SDT can be a separate phase of a compiler or we can augment our conventional grammar with information to control the semantic analysis and translation. Such grammars are called attribute grammars. We augment a grammar by associating attributes with each grammar symbol that describes its properties. With each production in a grammar, we give semantic rules/actions (*Syntax Directed Definition (SDD)*), which describe how to compute the attribute values associated with each grammar symbol in a production as shown in Figure 1:

Production	Semantic Rules
1) $L \rightarrow E$ **n**	$L.val = E.val$
2) $E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
3) $E \rightarrow T$	$E.val = T.val$
4) $T \rightarrow T_1 * F$	$T.val = T_1.val \times F.val$
5) $T \rightarrow F$	$T.val = F.val$
6) $F \rightarrow (E)$	$F.val = E.val$
7) $F \rightarrow$ **digit**	$F.val =$ **digit**$.lexval$

Figure 1 Grammar along with the SDD to represent a simple desk calculator

The general approach to SDT is to construct a parse tree or syntax tree and compute the values of attributes at the nodes of the tree by visiting them in some order. In many

cases, translation can be done during parsing without building an explicit tree. Figure 2 shows the syntax directed translation of an expression *3*5+4n* using the simple calculator grammar and the SDD shown in Figure 1:

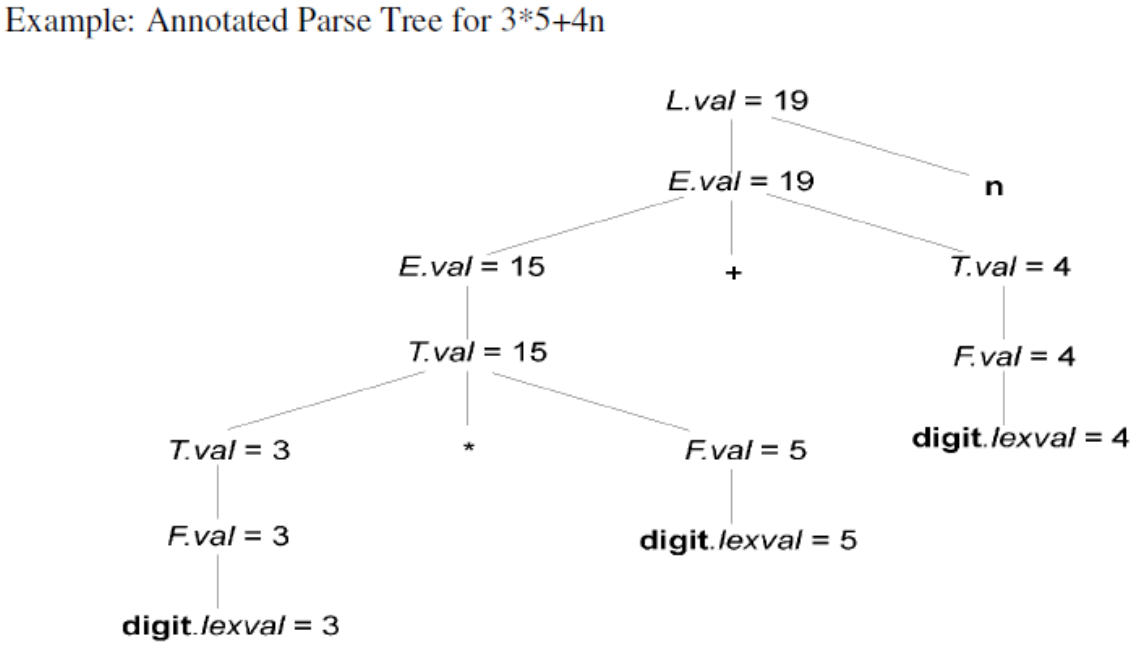

Figure 2 Syntax directed trnalsiation of an expression 3*4+5n

Lab Tasks

Recall that you have performed the task of postfix expression evaluation using stack in *Lab06 Basic Syntax Analysis using Bison.* Based on the knowledge of SDT that you have taken in the class lecture as well as in this lab perform the task of postfix evaluation using SDD. To be exact:

1. First, you have to write/alter the code given in "*Introduction to Bison*" tutorial shared during Lab06 using SDD.
2. After that run the code and perform the evaluation for the string **48+** and print the result onto the console.

HINT: You can take help from the task you have performed in *Lab07* and the help content shared with you along with the Lab07. Just for a quick help please look at the following CFG and how its respective code in Bison could look like:

E	$\longrightarrow$	E_1 * E_2	$\{E.val := E_1.val * E_2.val\}$
E	$\longrightarrow$	$E_1 + E_2$	$\{E.val := E_1.val + E_2.val\}$
E	$\longrightarrow$	int	$\{E.val := \text{int}.val\}$
E	:	E MULT E	$\{\$\$.val = \$1.val * \$3.val\}$
E	:	E PLUS E	$\{\$\$.val = \$1.val + \$3.val\}$
E	:	INT	$\{\$\$.val = \$1.val\}$

Note that MULT, PLUS and INT are of type token that can be defined in the declaration section of the code, else can be written directly as well.

Deliverables

You are required to upload your task (Sources & Word/PDF document) using the link created on LMS followed by a viva.