CS 335 Semester 2019–2020-II: Assignment 3

24th February 2020

Due Your assignment is due by Mar 8th 2020 11:59 PM IST.

General Policies

- You should do this assignment ALONE.
- Do not plagiarize or turn in solutions from other sources. You will be PENALIZED if caught.

Submission

- Submission will be through Canvas.
- Create a zip file named "`cs335_<roll>.zip`". The zipped file should contain a folder `assign2` with the following files:
  - Submit a `PDF` file with name "<roll-no>.pdf".
  - You SHOULD USE LaTeX typesetting system for generating the `PDF` file.
- Submitting your assignments late will mean losing marks automatically. You will lose 20% for each day that you miss, for up to two days.

Question 1 **[50 marks]**

Consider a computation with three operators: α, β, and γ. The inputs can be of two types: A and B.

Operator	# Inputs	Input Types	# Outputs	Output Types
α	1	A	1	A
β	2	B,B	1	B
γ	3	A,A,A or B,B,B	1	B

(a) Propose a context-free grammar (CFG) to generate expressions of the desired form (a couple of examples follow),

(b) Define a SDT based on your grammar from part (a) for type checking expressions,

(c) Draw the annotated parse tree for the expression:

$$\gamma(\gamma(\alpha(x1), x2, \alpha(x2)), \beta(y1, y2), \beta(y2, y3))$$

Given $type(x) = A$ and $type(y) = B$, the following is an example of a type-correct expression: $\beta(\gamma(\alpha(x), x, x), y)$.

Question 2 **[50 marks]**

Construct a SDT scheme that translates roman numerals into integers. The grammar and a reference table are given below.

$$\begin{aligned}
rnum &\rightarrow thousand\ hundred\ ten\ digit\\
thousand &\rightarrow \mathrm{M} \mid \mathrm{M\,M} \mid \mathrm{M\,M\,M} \mid \epsilon\\
hundred &\rightarrow smallHundred \mid \mathrm{C\,D} \mid \mathrm{D}\ smallHundred \mid \mathrm{C\,M}\\
smallHundred &\rightarrow \mathrm{C} \mid \mathrm{C\,C} \mid \mathrm{C\,C\,C} \mid \epsilon\\
ten &\rightarrow smallTen \mid \mathrm{X\,L} \mid \mathrm{L}\ smallTen \mid \mathrm{X\,C}\\
smallTen &\rightarrow \mathrm{X} \mid \mathrm{X\,X} \mid \mathrm{X\,X\,X} \mid \epsilon\\
digit &\rightarrow smallDigit \mid \mathrm{I\,V} \mid \mathrm{V}\ smallDigit \mid \mathrm{I\,X}\\
smallDigit &\rightarrow \mathrm{I} \mid \mathrm{II} \mid \mathrm{III} \mid \epsilon
\end{aligned}$$

Roman numeral	Decimal value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Do not worry about numbers that the given grammar cannot generate.

Question 3 **[50 marks]**

A program P is a sequence of statements separated by semicolons. Each statement assigns the value of an expression E to THE variable x. An expression is either the sum of two expressions, multiplication of two expressions, the constant 1, or the current value of x.

Statements are evaluated in left-to-right order.

- For the i^{th} statement $x = E_i$, the value of references to x inside E_i is the value assigned to x in the previous statement $x = E_{i-1}$,
- For the first statement $x = E_1$, the value of references to x in E_1 is 0,
- The value of a program is the value assigned to x by the last statement.

Answer the following:

(i) Propose a CFG to represent programs generated by the above specification,

(ii) Propose an SDT to compute the value of the program generated by P. Your solution should assign attribute $P.val$ the value of the program generated by P.

(iii) Indicate for each attribute whether it is inherited or synthesized.

Your solution should not use any global state.

Question 4 **[50 marks]**

Consider a programming language where reading from a variable *before* it has been assigned is an error. You are required to design an "undefined variable" checker for the language.

Your SDT should support the following requirements:

- If a statement S contains an expression E, and E references a variable that maybe undefined before reaching S, print the error message "A variable may be undefined". You need not print which variable (or variables) is undefined.
- If v is defined before a statement S, then v is also defined after S.
- Variable v is defined after the statement $v = E$.
- A variable defined inside an `if` is defined after the `if` when it is defined in BOTH branches.
- In a statement sequence $S1; S2$, variables defined after $S1$ are defined before $S2$.

$$\begin{aligned}
&\mathit{stmt} \rightarrow \mathit{var} = \mathit{expr} \\
&\mathit{stmt} \rightarrow \mathit{stmt}\ ;\ \mathit{stmt} \\
&\mathit{stmt} \rightarrow \textbf{if}\ \mathit{expr}\ \textbf{then}\ \mathit{stmt}\ \textbf{else}\ \mathit{stmt}\ \textbf{fi} \\
&\mathit{expr} \rightarrow \mathit{expr} + \mathit{expr} \\
&\mathit{expr} \rightarrow \mathit{expr} < \mathit{expr} \\
&\mathit{expr} \rightarrow \mathit{var} \\
&\mathit{expr} \rightarrow \textbf{int\_const}
\end{aligned}$$

Your solution should include the following attributes.

`var.name` is a string containing the variable's name. This is defined by the lexer, so you do not need to compute it.

`expr.refd` is the set of variables referenced inside the expression.

`stmt.indefs` is the set of variables defined at the beginning of the statement.

`stmt.outdefs` is the set of variables defined at the end of the statement.