utbm

SY48: Compilation and Language Theory - Final Exam A2025

Duration: 1h30. No document allowed. No electronic device allowed.
English recommended, French accepted.

Part 1: Introduction (5 points)

Question 1.1:

Consider the TypeScript language (extension of Javascript and Python). Is this programming language type-safe? Explain why.

Question 1.2:

What is a semantic rule during the syntax analysis? Explain briefly the concept and how it is used by a compiler.

Question 1.3:

What are the names the different parts of the program memory in a standard execution environment? Explain briefly the role of each part of the program memory.

Part 2: Tiny-Python (10 points)

In this part, you must consider a simplified version of the language Python, that has the name Tiny-Python and is explained below. In Tiny-Python, the functions are defined as:
`def <name> ( <param`$_1$`>, <param`$_2$`>, ..., <param`$_n$`> ) : <expression>`
where:

- `<name>` is an identifier that is the name of the function
- `<param`$_i$`>` is the formal parameter of the function at the position i. It is always an identifier
- `<expression>` is the body of the function as an expression composed of:
  - arithmetic expressions with floating point numbers, the operators `+`, `-`, `*`, `/`, and the open and close parentheses
  - power expressions with the operator `^`; such that `a^b` corresponds to `a*a*...*a` with `b` times the multiplication
  - function calls of the form: `<name> (<arg`$_1$`>, <arg`$_2$`>, ...)`

- "Conditional" statement:
  `if <condition> : <expression1> else : <expression2>`
  This statement evaluates to condition expression. If it is evaluated to true, the value of `<expression1>` is replied. Otherwise the value of `<expression2>` is replied.

Recommandation for answering:

You could answer on your paper sheet to the three following questions in a single Syntax Directed Definition (SDD) that is containing both the grammar rules in Backus-Naur Form (BNF) and the SDD semantic rules.

Question 2.1:

Write the grammar rules using BNF for recognizing the nonterminal `<expression>` for the Tiny-Python language. Your grammar will be analyzed with LL(0) approach. So that, you should avoid left-recursions and remove ambiguities from the grammar productions.

Question 2.2:

Extend the previous BNF grammar for recognizing the "function declaration" for the Tiny-Python language.

Question 2.3:

Write the semantic rules of the Syntax Direction Definition (SDD) to interprete the code and compute the numerical values for each element of the grammar. The goal of the SDD rules is to execute the Tiny-Python code and reply the computed numerical value. To do so, you must follow the points:

1. List the attributes for each nonterminal. Are they synthesized or inherited?
2. Write the semantic rules for each nonterminal that is computing the values of the attributes.

Part 3: Code Generation (5 points)

In this part, you must focus on the generation of the form of three-address code that is named "quadruple." The language to translate is the one described in the part 2.

Remainder:

In the generated three-address code, you may not define the addresses themselves, but use symbolic names in place.

Question 3.1:

In this question, you must assume that you have **NOT** obtained a syntax tree from the parsing stage. The generation of the quadruples must be directly put in the semantic rules of the SDD.

1. Define the additional attributes that are required to generate the "quadruples".
2. Write the SDD to generate the "quadruples" for all the nonterminals of the grammar productions that are defined in Part 2.