Documentation of Project Implementation for IPP 2018/2019

Name and surname: Juraj Holub
Login: xholub40

1 Proposal of solution

This section introduces the proposed implementation. Figure 1 shows the scheme of the implemented interpret. The program starts with parsing source IPPcode19 program in XML representation to program inner representation. In this phase program done all necessary syntactic analyses. In the next step, application execute a parsed sequence of IPPcode19 instructions. Interpret done all semantic analyses in runtime. An output of the program is produced to a standard output stream. In case of any syntactic or semantic error, interpret stop execute source code and return a specific error value. Interpret also generate a brief error message to the standard error stream.

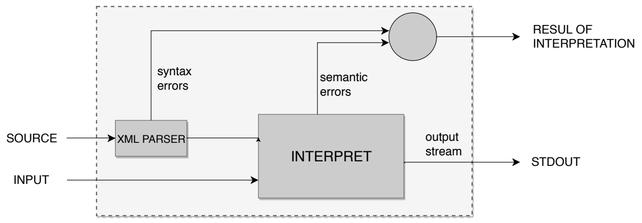

Figure 1: Scheme of IPPcode19 program interpretation.

2 Implementation details

Implementation of the solution proposed in section 1 is parted to logical blocks described in this section.

- **XML parser**: Class parse raw XML representation of source program to the ordered list of IPPcode19 instructions. Parser class uses python3 library The ElementTree XML API which provide all necessary syntactic analyses of input XML. Class also verifies syntactic correctness of every instruction (for example XML representation of instruction is correct but IPPcode19 syntactic meaning is incorrect). If any error was found then program finish and interpretation itself never occurs.
- **Interpret**: At the beginning, the interpret iterate all instructions and done fast analyse. If analysed instruction is label then interpret saves information about position and name. After this phase program launch interpretation and one by one execute instructions. At this point, if an actually executed command is an instruction of jump then interpret knows about the existence of requested label and his position in the code. For every instruction, there is implemented one method which executes it. Interpret class save data about all variables, their frames, about data stack and call stack. All instruction operands data are converted to python data types. This approach allows transfer responsibility of runtime type conversion and analyses to python interpret.
- **Error handling**: Interpret raises the specific exception in case of any runtime error. Every semantic error is represented by a specific user-defined exception. These exceptions hold data about the actual situation in the program. Interpret inform a user to standard error stream about type of error, data which caused it and position in the code.
- **String convertor**: Source programming language allows declaring string literals with escape sequences. String converter class transform input string literal with escape sequences to a human-readable format. The converter is implemented like a finite state machine.