Methodologies for Software Processes
Seminar 2
Weeks: 11.03.2019-22.03.2019

Dataflow Programming Project

Each group must implement this project.
Deadline is either 16.05.2019 or 23.05.2019, at the seminar.
After 23.05.2019, the project cannot be submitted anymore.

Please implement in OCAML the backward dataflow analysis framework and then use it for live variables analysis. The input of your dataflow analysis framework is the abstract syntax tree of a program written in a small language called ToyWhile language. First you have to build the control flow graph from the input abstract syntax tree of the program and then you have to apply the live variables analysis to determine the set of live variables at each program point. The output of your implementation is the set of live variables at each point of the input program.
A program (Prg) in ToyWhile language consists of a statement (Stmt) as follows:
Prg := Stmt where the symbol "::=" means "a Prg is defined as a Stmt".

A statement can be either a compound statement (CompStmt), or an assignment statement (AssignStmt), or a print statement (PrintStmt), or a conditional statement (IfStmt), or a while statement (WhileStmt) as follows:

```
Stmt ::= Stmt1 ; Stmt2                          /* (CompStmt)*/
          |  Id = Exp                            /* (AssignStmt)*/
          |  Print(Exp)                          /* (PrintStmt)*/
          |  If  Expr Then Stmt1  Else Stmt2     /*   (IfStmt)*/
          | while Expr do Stmt                  /*(WhileStmt) */
```

where the symbol "|" denotes the possible definition alternatives.

An expression (Exp) can be either an integer number (Const), or a variable name (Var), or an arithmetic expression (ArithExp) as follows:

```
Exp ::= Number                         /*(Const)*/
        |  Id                          /*(Var)*/
        |  Exp1 + Exp2                 /*(ArithExp)*/
        |  Exp1  - Exp2
        |  Exp1 * Exp2
        | Exp1 / Exp2
```

where Number denotes an integer constant, and Id denotes a variable name. When an expression is used as a condition in IfStmt and WhileStmt we use C-like convention, that means if expression value is zero then that expression as condition is considered False otherwise if expression value is not zero then that expression as condition is considered True.
The program is introduced as an abstract syntax tree directly coded in your main function of your implementation. You do NOT need to implement the lexical and syntax analysis (parser) in order to read the input program as a text program.

In a summary you have to do the followings:

1. define the datatypes for the abstract syntax tree for the TOY Language
2. define the datatypes for the control flow graph
3. implement the function that transforms the abstract syntax tree in to the control flow graph
4. define and implement the general backward dataflow analysis framework
5. instantiate the general backward dataflow analysis framework with the live variables analysis
6. buid the set of live variables for each node of the control flow graph
7. display the results of the step 6 on the control flow graph