Assignment 1

1. Intra Procedural Dominator Analysis

Dominator analysis is an important problem, with applications ranging from optimizing compilers to network flow analysis. The analysis determines for each node in a directed graph by which other nodes it is dominated by. Here we use the Control Flow Graph (CFG) of programs to detetermine for each node the set of dominating nodes. A number of compiler optimizations rely on dominator analysis to determine guaranteed execution relationships of blocks in a CFG.

Let labels of statements and expressions denote the corresponding nodes in the Control Flow Graph (CFG):

- Assume every CFG has start node s_0 with no predecessors.
- Node d dominates node n if every path of directed edges from s_0 to n must go through d.
- Every node dominates itself

Example for the WHILE language[1] :

```
program DominatorAnalysis
[begin]^0
  [a := 1]^1;
  [b := a]^2;
  while [a < 10]^3 do (
    if[a < b]^4 then
      [a := a+1]^5;
    else
      [b := b+1]^6;
    [c := a+b]^7;
  )
[end]^8
```

ℓ	$DA_\circ(\ell)$	$DA_\bullet(\ell)$
0	{?,}	{?,}
1	{?,}	{?,1,}
2	{?,1,}	{?,1,2,}
3	{?,1,2,}	{?,1,2,3,}
4	{?,1,2,3,}	{?,1,2,3,4,}
5	{?,1,2,3,4,}	{?,1,2,3,4,5,}
6	{?,1,2,3,4,}	{?,1,2,3,4,6,}
7	{?,1,2,3,4,}	{?,1,2,3,4,7,}
8	{?,1,2,3}	

[1]For 'begin' and 'end' we do not update the analysis information in an intra procedural analysis.

Specify the Dominator Analysis for WHILE:

a) Define $\text{kill}_{\text{DA}}(\ell)$ and $\text{gen}_{\text{DA}}(\ell)$.

b) Define the equations for $\text{DA}_\circ(\ell), \text{DA}_\bullet(\ell) : \text{Lab}_\star \to \mathcal{P}(\text{Lab}_\star^?)$

2. Intra Procedural Dominator Analysis with PAG

Specify a Dominator Analysis for the language SL_1 with PAG and create an analyzer 'da' that takes as input an SL_1 program and generates as output a gdl file, describing the CFG and the analysis result for each program point, named 'da_result.gdl'.
The language SL_1 is defined as a subset of C++. Every SL_1 program is a legal C++ program. The SL_1 language corresponds to the WHILE language by representing all WHILE language constructs with the corresponding C++ constructs. In contrast to WHILE, in SL_1 all variables must be declared before they can be used. The only types that are allowed for variables in SL_1 are 'int' and 'bool'. There exists only one function without parameters, called 'main' with a return type of 'int'. The function main should return an int value as last statement.

SL_1 Example: (corresponding program to the WHILE example)

```
int main() {
  int a,b,c;
  a=1;
  b=a;
  while(a<10) {
    if(a<b) {
      a=a+1;
    } else {
      b=b+1;
    }
    c=a+b;
  }
  return 0;
}
```

3. Hand in

- Send your answers for questions 1a, 1b, and the PAG specification for 2 per e-mail to `markus@complang.tuwien.ac.at`
- The e-mail must have as subject "OPTUB:Assignment 1, <LastName>" where <LastName> is replaced with your last name. The PAG specification should be attached as tgz file containing all required files for creating the analyzer. The answers should be provided as text in the email body or alternatively as attached postscript or pdf files (included in the tgz file).
- Deadline: 2pm October 31, 2007.

For further information on how to create a new analysis, additional information on the program representation, and how to view analyses read the FAQ on the OPTUB website.