# Program Transformation

December 28, 2015

**Algorithm 1** analyse a module

```
1: function ANALYSE($M$)
2:     for all $F \in M$ do
3:         $controlStructs \leftarrow getTopLevelControlStructs(F.BList)$
4:         for all $S \in controlStructs$ do
5:             $analyse(S)$
6:         end for
7:     end for
8: end function
```

**Algorithm 2** analyse a control structure

```
1: function ANALYSE($S$)
2:     if $isTransformable(S)$ then
3:         $transform(S)$
4:     else
5:         $controlStructs \leftarrow getTopLevelControlStructs(S.BList)$
6:         for all $S' \in controlStructs$ do
7:             $analyse(S')$                                ▷ do recursive analysis
8:         end for
9:     end if
10: end function
```

**Algorithm 3** check if a control structure is transformable

```
1: function ISTRANSFORMABLE($S$)
2:     if $\forall var \in def(S),\ var \notin OUT[S]$ then
3:         return true
4:     else
5:         return false
6:     end if
7: end function
```

**Algorithm 4** transform a control structure

```
function TRANSFORM($S$)
    $TI \leftarrow S.header.getTerminator$
    for all $B \in S$ do
        for all $I \in B$ do
            if $isCritical(I)$ then        ▷ check if dereference or getElementOfArray
                $insert\ I\ before\ TI$
            end if
        end for
    end for
    $B \leftarrow S.exit$
    $I \leftarrow$ br label B
    $replace\ TI\ with\ I$
end function
```

**Algorithm 5** get a conditional structure from the conditional branch block

```
function GETCONDSTRUCT($B$)
    $B' \leftarrow postdominator\ of\ B$
    $blocks \leftarrow getBlocksBetween(B,\ B')$        ▷ traverse CFG from B to B'
    $S \leftarrow ControlStruct(blocks)$
    return $S$
end function
```

**Algorithm 6** get all of the top level control structures in the basic block list

```
1: function getTopLevelControlStructs($Blist$)
2:     $ret \leftarrow empty\ ControlStruct\ list$
3:     for all $B \in BList$ do
4:         if $B\ is\ loop\ header$ then
5:             $LS \leftarrow getLoopStruct(B)$
6:             if $LS\ has\ one\ successor$ then
7:                 $add\ LS\ to\ ret$
8:                 $skipBlocksIn(LS)$                ▷ don't analyse blocks in LS
9:             else
10:                $CS \leftarrow getCondStruct(B)$
11:                $add\ CS\ to\ ret$
12:                $skipBlocksIn(CS)$
13:            end if
14:        else if $B\ is\ conditional\ branch\ block$ then       ▷ B is 'if' or 'switch' conditional block
15:            $CS \leftarrow getCondStruct(B)$
16:            $add\ CS\ to\ ret$
17:            $skipBlocksIn(CS)$
18:        end if
19:    end for
20:    return $ret$
21: end function
```

## Design of ControlStruct class

```
class ControlStruct {
  BasicBlock *_header;
  BasicBlock *_exit;
  vector<BasicBlock *> _blocks;

public:
  ControlStruct(BasicBlock *header,
                BasicBlock *exit,
                const vector<BasicBlock *> &blocks);

  BasicBlock *header();

  BasicBlock *exit();

  bool isTransformable() const;

  Type getType() const; // Condition or Loop

  vector<ControlStruct> getTopLevelControlStructs() const;

  static vector<ControlStruct> getTopLevelControlStructs(const
       Function &F);
}
```

## Design of Transformation class

```
class Transformation {
public:
  static void analyse(Module &M);

  static void analyse(ControlStruct &S);

  static void transform(ControlStruct &S);
}
```