Program Transformation

December 28, 2015

Algorithm 1 analyse a module

```
1: function ANALYSE(M)
2:     for all F ∈ M do
3:         controlStructs ← getTopLevelControlStructs(F.BList)        ▷ ControlStruct is a conditional branch or loop
4:         for all S ∈ 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 ← getTopLevelControlStructs(S.BList)
6:         for all S′ ∈ 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 ∀var ∈ def(S), var ∉ OUT[S] then
3:         return true
4:     else
5:         return false
6:     end if
7: end function
```

Algorithm 4 transform a control structure

```
1: function transform(S)
2:     for all B ∈ S do
3:         for all I ∈ B do
4:             if isCritical(I) then                ▷ check if dereference or getElementOfArray
5:                 insert I before S
6:             end if
7:         end for
8:     end for
9:     B ← getSuccessor(S)
10:    insert unconditional branch br label B before S
11: end function
```

Algorithm 5 get a conditional structure from the conditional branch block

```
1: function getCondStruct(B)
2:     B' ← postdominator of B
3:     blocks ← getBlocksBetween(B, B')        ▷ traverse CFG from B to B'
4:     S ← ControlStruct(blocks)
5:     return S
6: end function
```

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

```
1: function getTopLevelControlStructs(Blist)
2:     ret ← empty ControlStruct list
3:     for all B ∈ BList do
4:         if B is loop header then
5:             LS ← getLoopStruct(B)
6:             if LS has one successor then
7:                 add LS to ret
8:                 skipBlocksIn(LS)
9:             else
10:                CS ← 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 ← 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

```
1  class ControlStruct {
2    vector<BasicBlock *> blocks;
3    BasicBlock *header;
4    BasicBlock *exit;
5
6  public:
7    ControlStruct(const vector<BasicBlock *> &basicblocks);
8
9    bool isTransformable() const;
10
11   Type getType() const; // Condition or Loop
12
13   vector<ControlStruct> getTopLevelControlStructs() const;
14
15   static vector<ControlStruct> getTopLevelControlStructs(const
         Function &F);
16 }
```

Design of Transformation class

```
1  class Transformation {
2  public:
3  static void analyse(Module &M);
4
5  static void analyse(ControlStruct &S);
6
7  static void transform(ControlStruct &S);
8  }
```