Code Verification Using Symbolic Execution

(Week 7)

Yulei Sui

School of Computer Science and Engineering

University of New South Wales, Australia

Example 4: Branches

```c
void main(int argv){
    foo(x)
}
void foo(int x){
    int y;
    if(x > 10) {
        y = x + 1;
    }
    else {
        y = 10;
    }
    svf_assert(y >= x + 1);
}
```

Source code

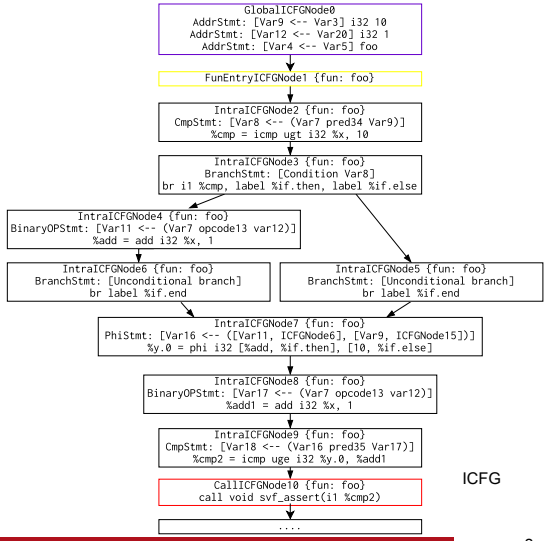
```
define void @foo(i32 %x) #0 {
entry:
  %cmp = icmp ugt i32 %x, 10
  br i1 %cmp, label %if.then, label %if.else

if.then:
  %add = add i32 %x, 1
  br label %if.end

if.else:
  br label %if.end

if.end: = %if.else, %if.then
  %y.0 = phi i32 [%add, %if.then], [10, %if.else]
  %add1 = add i32 %x, 1
  %cmp2 = icmp uge i32 %y.0, %add1
  call void @svf_assert(i1 zeroext %cmp2)
  ret void
}
```

LLVM IR

Example 4: Branches

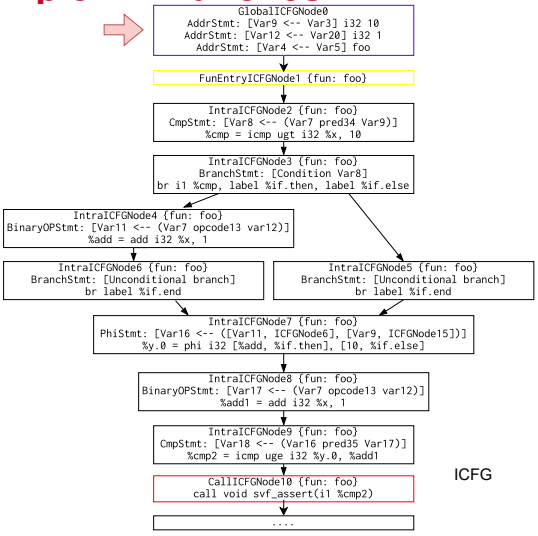
```
1  define void @foo(i32 %x) #0 {
2  entry:
3    %cmp = icmp ugt i32 %x, 10
4    br i1 %cmp, label %if.then, label %if.else
5
6  if.then:
7    %add = add i32 %x, 1
8    br label %if.end
9
10 if.else:
11   br label %if.end
12
13 if.end: = %if.else, %if.then
14   %y.0 = phi i32 [%add, %if.then], [10, %if.else]
15   %add1 = add i32 %x, 1
16   %cmp2 = icmp uge i32 %y.0, %add1
17   call void @svf_assert(i1 zeroext %cmp2)
18   ret void
19 }
```

LLVM IR



ICFG

Example 4: Branches



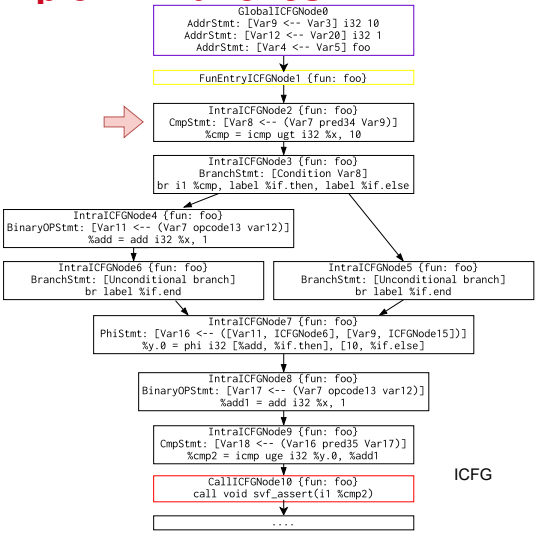
ICFG

Verifying ICFG path: $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow$ *svf_assert* (if.then branch)

```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 10
ValVar12                 Value: 1
                  ...
------------------------------------------
```

The values of Z3 expressions for each SVFVar after analyzing `GlobalICFGNode0`

Example 4: Branches



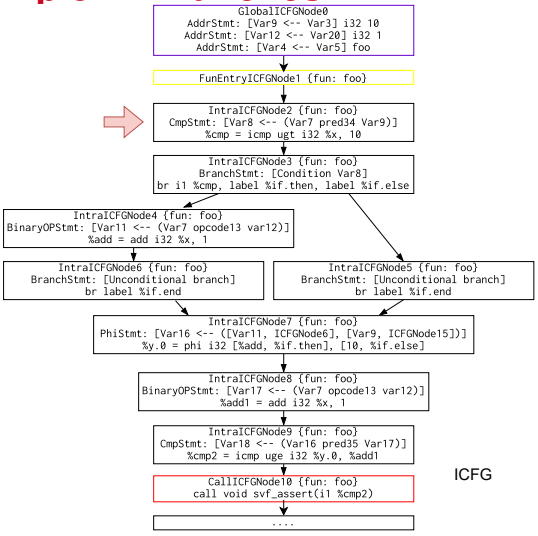
ICFG

Verifying ICFG path: 0 → 1 → 2 → 3 → 4 → 6 → 7 → 8 → 9 → *svf_assert* (if.then branch)

```
## Analyzing IntraICFGNode2 {fun: foo}
CmpStmt: [Var8 <-- (Var7 predicate34 Var9)]
   %cmp = icmp ugt i32 %x, 10
==> (not (<= ValVar7 ValVar9))
==> (= ValVar8 1)
                    ...
```

Code for handling `CmpStmt` has been implemented in the HandleNonBranch() function.

Example 4: Branches



ICFG

Verifying ICFG path: 0 → 1 → 2 → 3 → 4 → 6 → 7 → 8 → 9 → *svf_assert* (if.then branch)

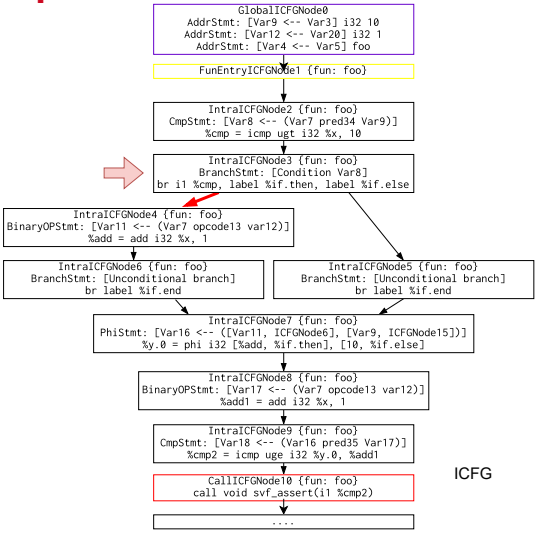
```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 10
ValVar12                 Value: 1
ValVar7                  Value: 11
ValVar8                 Value: 1
                  ...
------------------------------------------
```

Analyzing IntraICFGNode2 {fun: foo}

CmpStmt: [Var8 ← (Var7 pred34 Var9)]

`%cmp = icmp ugt i32 %x, 10`

Example 4: Branches



ICFG

Algorithm 1: 3 `handleIntra(intraEdge)`

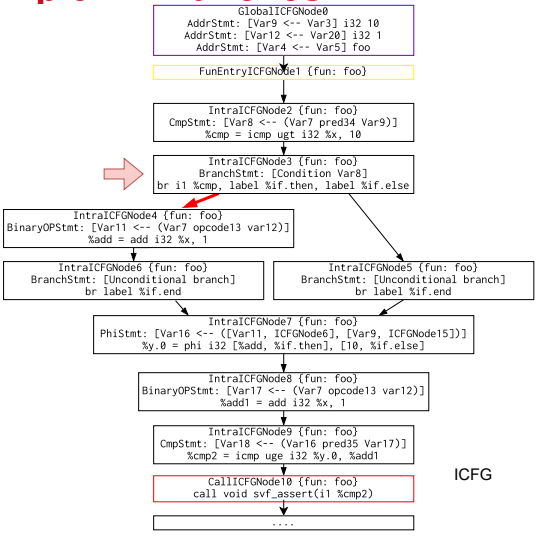
```
2  if intraEdge.getCondition() && !handleBranch(intraEdge) then
4      return false;
6  else
8      handleNonBranch(edge);
```

Algorithm 2: `handleBranch(intraEdge)`

```
1  cond = intraEdge.getCondition();
2  succ = intraEdge.getSuccessorCondValue();
3  getSolver().push();
4  addToSolver(cond == succ);
5  res = getSolver().check();
6  getSolver().pop();
7  if res == unsat then
8      return false;
9  else
10     addToSolver(cond == succ);
11     return true;
```

Note: `getSolver().push()` creates a new stack frame for maintaining the newly added Z3 constraints.

Example 4: Branches



ICFG

Algorithm 3: 3 `handleIntra(intraEdge)`

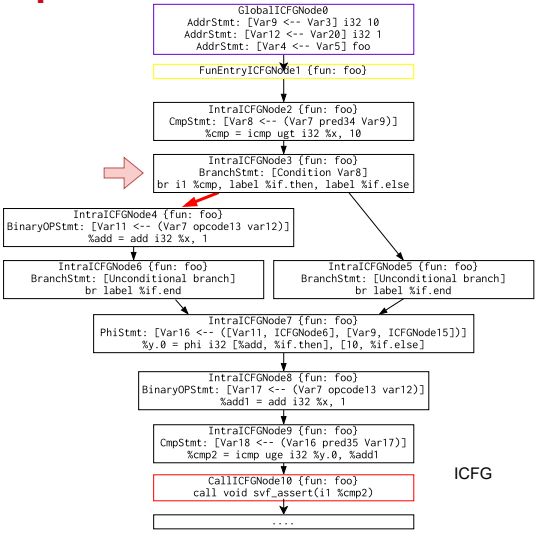
```
2  if intraEdge.getCondition() &&
     !handleBranch(intraEdge) then
4      return false;
6  else
8      handleNonBranch(edge);
```

Algorithm 4: `handleBranch(intraEdge)`

```
1  cond = intraEdge.getCondition();
2  succ = intraEdge.getSuccessorCondValue();
3  getSolver().push();
4  addToSolver(cond == succ);
5  res = getSolver().check();
6  getSolver().pop();
7  if res == unsat then
8      return false;
9  else
10     addToSolver(cond == succ);
11     return true;
```

Note: `getSolver().pop()` pops out the top stack frame, which contains the Z3 constraint `cond == succ`.

Example 4: Branches



ICFG

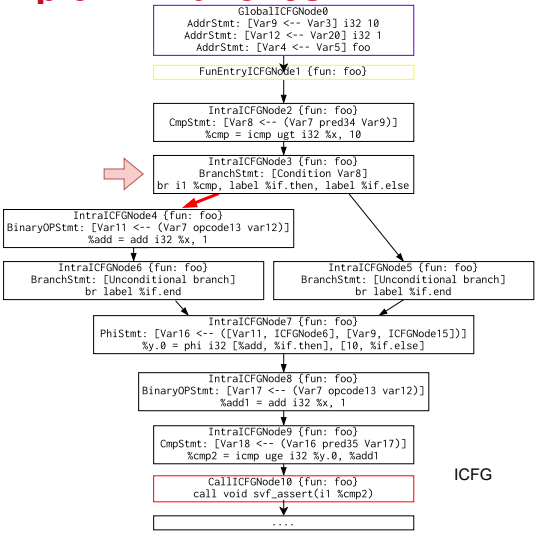
Algorithm 5: 3 `handleIntra(intraEdge)`

```
1  if intraEdge.getCondition() && !handleBranch(intraEdge) then
2      return false;
3  else
4      handleNonBranch(edge);
```

Algorithm 6: `handleBranch(intraEdge)`

```
1  cond = intraEdge.getCondition();
2  succ = intraEdge.getSuccessorCondValue();
3  getSolver().push();
4  addToSolver(cond == succ);
5  res = getSolver().check();
6  getSolver().pop();
7  if res == unsat then
8      return false;
9  else
10     addToSolver(cond == succ);
11     return true;
```

Example 4: Branches



ICFG

Verifying ICFG path: $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow$ *svf_assert* (if.then branch)

```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 10
ValVar12                 Value: 1
ValVar7                  Value: 11
ValVar8                  Value: 1
                  ...
------------------------------------------
```

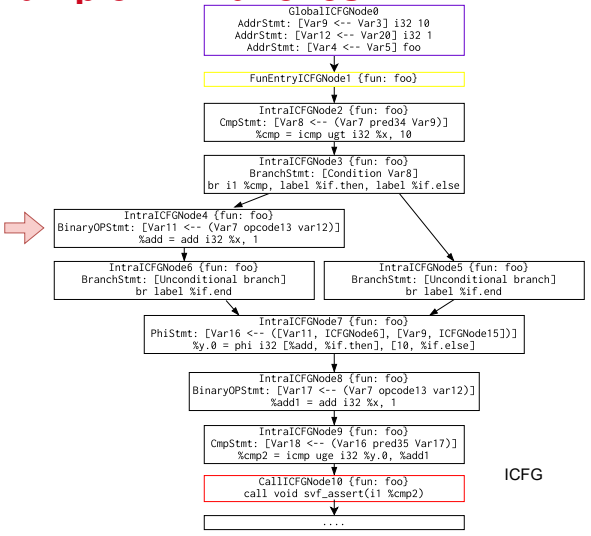
Branch IntraCFGEdge: [ICFGNode4 ← ICFGNode3]

branchCondition: %cmp = icmp ugt i32 %x, 10

(= ValVar8 1)

This conditional ICFGEdge is **feasible**!!

Example 4: Branches



ICFG

Verifying ICFG path: $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow$ *svf_assert* (if.then branch)

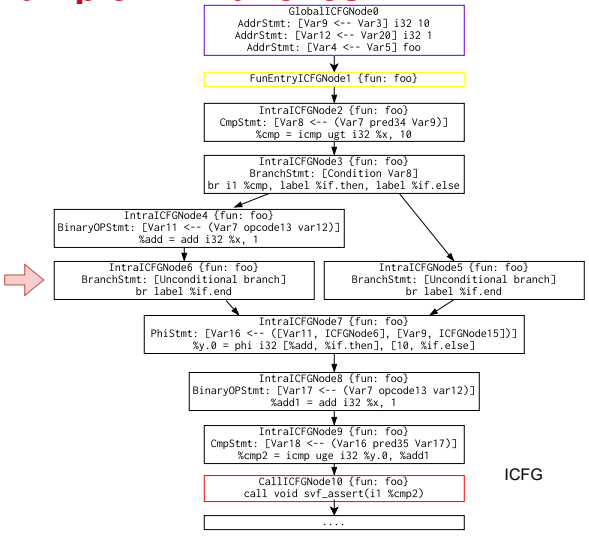
```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 10
ValVar12                 Value: 1
ValVar7                  Value: 11
ValVar8                  Value: 1
ValVar11                 Value: 12
                  ...
------------------------------------------
```

Analyzing IntraICFGNode4 {fun: foo}

BinaryOPStmt: [Var11 $\leftarrow$ (Var7 opcode13 var12)]

`%add = add i32 %x, 1`

Example 4: Branches

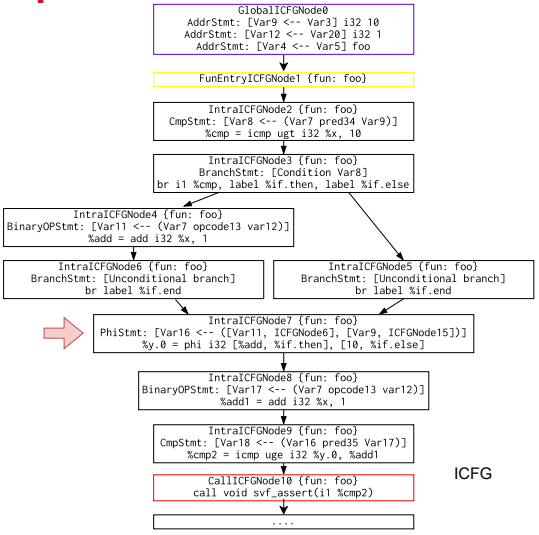


ICFG

Algorithm 7: 3 `handleIntra(intraEdge)`

```
2  if intraEdge.getCondition() &&
     !handleBranch(intraEdge) then
4      return false;
6  else
8      handleNonBranch(edge);
```

Example 4: Branches

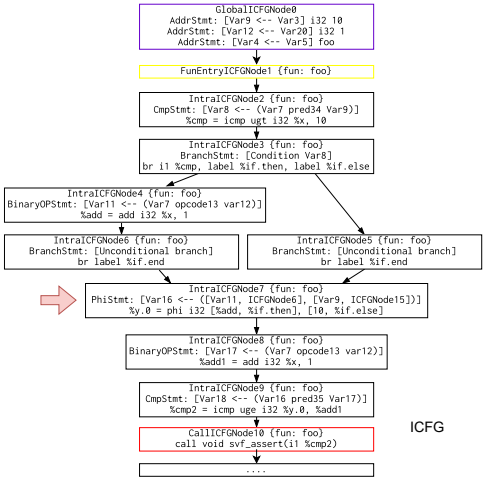


ICFG

Algorithm 8: 3 `handlePhi(intraEdge)`

```
1 res ← getZ3Expr(phi.getResID());
2 opINodeFound ← false;
3 for i ← 0 to phi.getOpVarNum() − 1 do
4     if srcNode.getFun().postDominate(srcNode.getBB(),
5     phi.getOpICFGNode(i).getBB()) then
6         ope ← getZ3Expr(phi.getOpVar(i).getId());
7         addToSolver(res == ope);
8         opINodeFound ← true;
```

Example 4: Branches



ICFG

Verifying ICFG path: 0 → 1 → 2 → 3 → 4 →
6 → 7 → 8 → 9 → *svf_assert* (if.then branch)

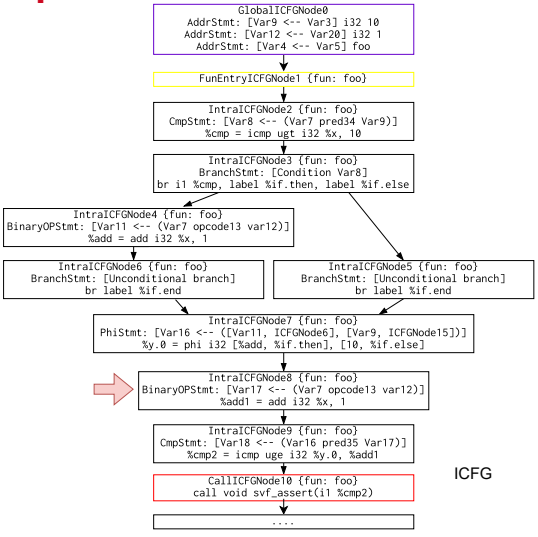
```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)    Value: NULL
ValVar4                 Value: 0x7f000005
ValVar9                 Value: 10
ValVar12                Value: 1
ValVar7                 Value: 11
ValVar8                 Value: 1
ValVar11                Value: 12
ValVar16                Value: 12
                    ...
------------------------------------------
```

Analyzing IntraICFGNode7 {fun: foo}

PhiStmt: [Var16 ← ([Var11, ICFGNode6], [Var9, ICFGNode15])]

$\%y.0 \;=\; \texttt{phi i32}\,[\%\texttt{add},\ \%\texttt{if.then}],\ [10,\ \%\texttt{if.else}]$

Example 4: Branches



ICFG

Verifying ICFG path: 0 → 1 → 2 → 3 → 4 → 6 → 7 → 8 → 9 → *svf_assert* (if.then branch)

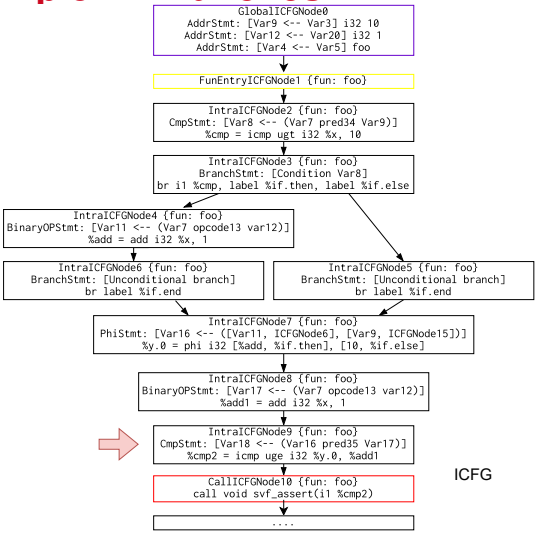
```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 10
ValVar12                 Value: 1
ValVar7                  Value: 11
ValVar8                  Value: 1
ValVar11                 Value: 12
ValVar16                 Value: 12
ValVar17                Value: 12
                    ...
------------------------------------------
```

Analyzing IntraICFGNode8 {fun: foo}

BinaryOPStmt: [Var17 ← (Var7 opcode13 var12)]

`%add1 = add i32 %x, 1`

Example 4: Branches



ICFG

Verifying ICFG path: $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow$ *svf_assert* (if.then branch)

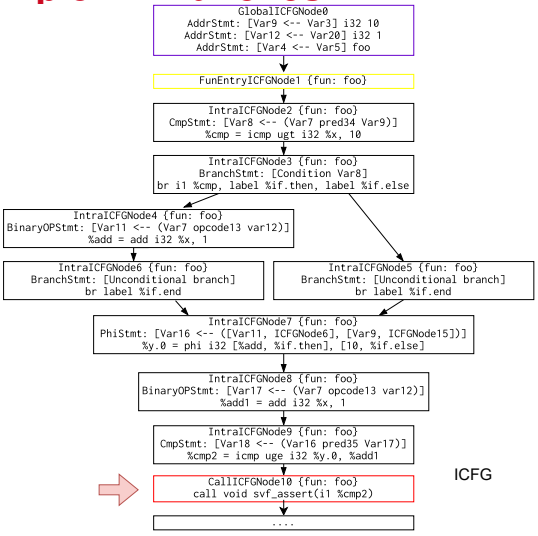
```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 10
ValVar12                 Value: 1
ValVar7                  Value: 11
ValVar8                  Value: 1
ValVar11                 Value: 12
ValVar16                 Value: 12
ValVar17                 Value: 12
ValVar18                Value: 1
                  ...
------------------------------------------
```

Analyzing IntraICFGNode9 {fun: foo}

CmpStmt: [Var18 ← (Var16 pred35 Var17)]

`%cmp2 = icmp uge i32 %y.0, %add1`

Example 4: Branches



ICFG

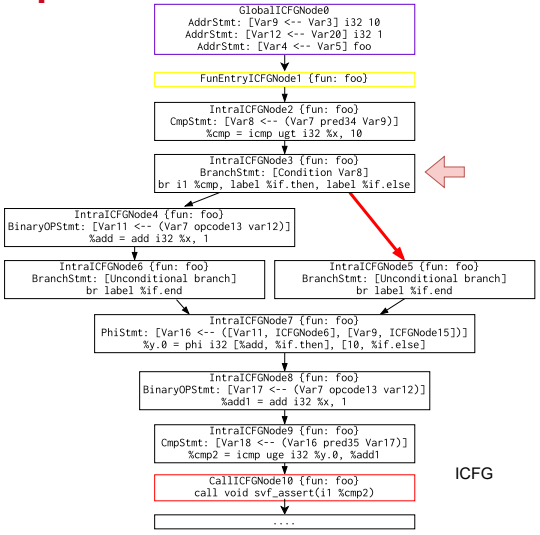
Verifying ICFG path: 0 → 1 → 2 → 3 → 4 → 6 → 7 → 8 → 9 → *svf_assert* (if.then branch)

```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 10
ValVar12                 Value: 1
ValVar7                  Value: 11
ValVar8                  Value: 1
ValVar11                 Value: 12
ValVar16                 Value: 12
ValVar17                 Value: 12
ValVar18                 Value: 1
                  ...
------------------------------------------
```

The assertion is successfully verified!!

START: 0 → 1 → 2 → 3 → 4 → 6 → 7 → 8 → 9 → *svf_assert*

Example 4: Branches



ICFG

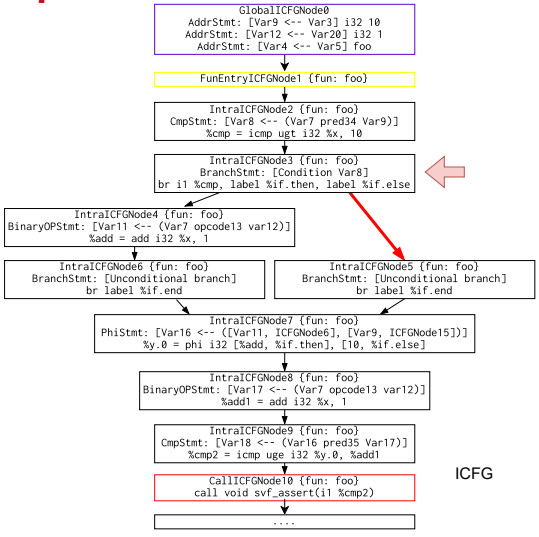
Algorithm 9: 3 `handleIntra(intraEdge)`

```
2  if intraEdge.getCondition() && !handleBranch(intraEdge) then
4      return false;
6  else
8      handleNonBranch(edge);
```

Algorithm 10: `handleBranch(intraEdge)`

```
1  cond = intraEdge.getCondition();
2  succ = intraEdge.getSuccessorCondValue();
3  getSolver().push();
4  addToSolver(cond == succ);
5  res = getSolver().check();
6  getSolver().pop();
7  if res == unsat then
8      return false;
9  else
10     addToSolver(cond == succ);
11     return true;
```

Example 4: Branches



ICFG

Verifying ICFG path: $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow$ *svf_assert* (if.else branch)

```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 10
ValVar12                 Value: 1
ValVar7                  Value: 10
ValVar8                  Value: 0
                  ...
------------------------------------------
```

Branch IntraCFGEdge: [`ICFGNode4` ← `ICFGNode3`]

branchCondition: `%cmp = icmp ugt i32 %x, 10`

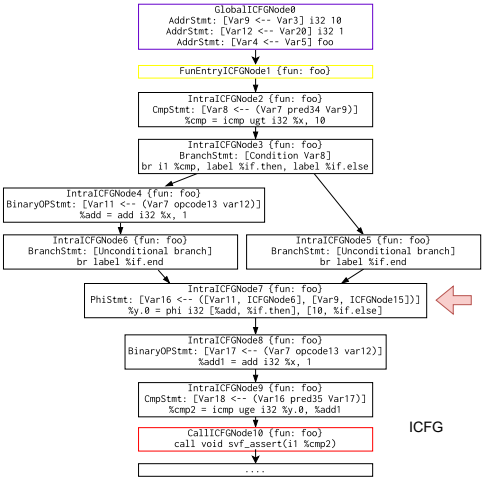
(= `ValVar8 0`)

This conditional ICFGEdge is **feasible**!!

In this path, `ValVar8`'s value is chaged to 0,

therefore, `ValVar7`'s value is chaged to 10.

Example 4: Branches



ICFG

Verifying ICFG path: 0 → 1 → 2 → 3 → 5 →
7 → 8 → 9 → *svf_assert* (if.else branch)

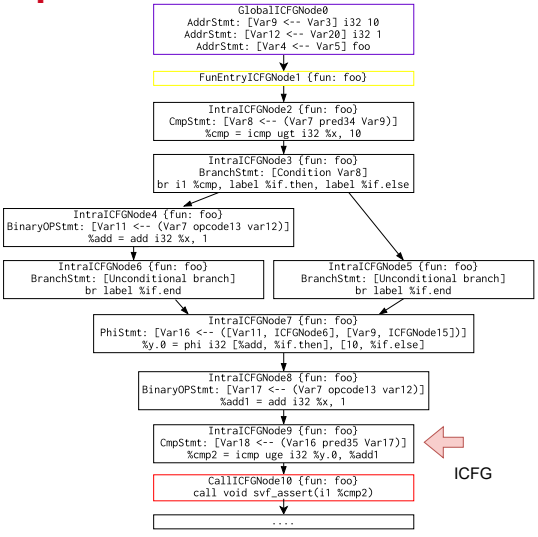
```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)    Value: NULL
ValVar4                 Value: 0x7f000005
ValVar9                 Value: 10
ValVar12                Value: 1
ValVar7                 Value: 10
ValVar8                 Value: 1
ValVar16                Value: 10
                    ...
------------------------------------------
```

Analyzing IntraICFGNode7 {fun: foo}

PhiStmt: [Var16 ← ([Var11, ICFGNode6], [Var9, ICFGNode5])]

`%y.0 = phi i32 [%add, %if.then], [10, %if.else]`

Example 4: Branches



Verifying ICFG path: 0 → 1 → 2 → 3 → 5 → 7 → 8 → 9 → *svf_assert* (if.else branch)

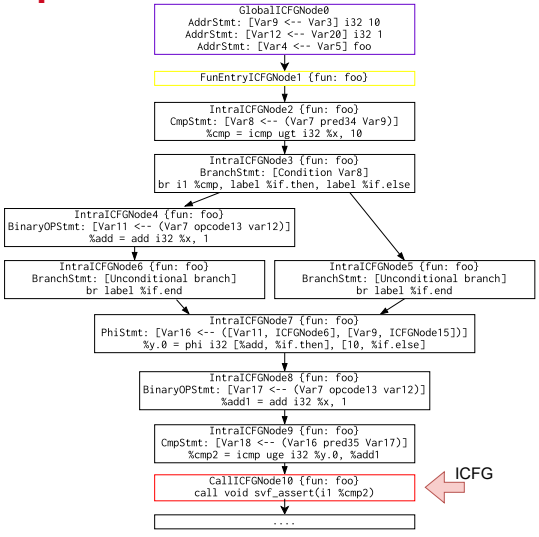
```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)    Value: NULL
ValVar4                 Value: 0x7f000005
ValVar9                 Value: 10
ValVar12                Value: 1
ValVar7                 Value: 11
ValVar8                 Value: 1
ValVar16                Value: 10
ValVar17                Value: 11
ValVar18                Value: 0
                    ...
------------------------------------------
```

Analyzing IntraICFGNode9 {fun: foo}

CmpStmt: [Var18 ← (Var16 pred35 Var17)]

`%cmp2 = icmp uge i32 %y.0, %add1`

Example 4: Branches



Verifying ICFG path: 0 → 1 → 2 → 3 → 5 → 7 → 8 → 9 → *svf_assert* (if.else branch)

```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 10
ValVar12                 Value: 1
ValVar7                  Value: 11
ValVar8                  Value: 1
ValVar16                 Value: 10
ValVar17                 Value: 11
ValVar18                 Value: 0
                  ...
------------------------------------------
```

The assertion fails!!

START: 0 → 1 → 2 → 3 → 5 → 7 → 8 → 9 → *svf_assert*

Example 5: Interprocedural

```
1  int foo(int p) {
2      return p;
3  }
4  int main(int argc) {
5    int x;
6    int y;
7    x = foo(3); // ctx_7
8    y = foo(argc); // ctx_8
9    assert(y == argc);
10 }
```

ctx_7: context calling `foo` at ℓ_7

ctx_8: context calling `foo` at ℓ_8

Example 5: Interprocedural

```
1  int foo(int p) {
2      return p;
3  }
4  int main(int argc) {
5    int x;
6    int y;
7    x = foo(3); // ctx_7
8    y = foo(argc); // ctx_8
9    assert(y == argc);
10 }
```

ctx_7: context calling foo at ℓ_7

ctx_8: context calling foo at ℓ_8

One Concrete Execution
(Concrete states)

One execution:

argc : 0

push calling context (calling foo at ℓ_7)

p : 3

calling context pop (returning from foo at ℓ_2)

x : 3

push calling context (calling foo at ℓ_8)

p : 0

pop calling context (returning from foo ℓ_2)

y : 0

Example 5: Interprocedural

```
1  int foo(int p) {
2      return p;
3  }
4  int main(int argc) {
5    int x;
6    int y;
7    x = foo(3); // ctx_7
8    y = foo(argc); // ctx_8
9    assert(y == argc);
10 }
```

ctx_7: context calling foo at ℓ_7

ctx_8: context calling foo at ℓ_8

One Concrete Execution
(Concrete states)

One execution:

argc : 0

push calling context (calling foo at ℓ_7)

p : 3

calling context pop (returning from foo at ℓ_2)

x : 3

push calling context (calling foo at ℓ_8)

p : 0

pop calling context (returning from foo ℓ_2)

y : 0

Symbolic Execution
(Symbolic states)

One execution:

argc : getZ3Expr(argc)

push calling context (calling foo at ℓ_7)

ctx_7_p : 3

pop calling context (returning from foo at ℓ_2)

x : getZ3Expr(p, ctx_7)

push calling context (calling foo at ℓ_8)

ctx_8_p : getZ3Expr(argc)

pop calling context (returning from foo ℓ_2)

y : getZ3Expr(p, ctx_8)

Example 5: Interprocedural

```
1  int foo(int p) {
2      return p;
3  }
4  int main(int argc) {
5    int x;
6    int y;
7    x = foo(3); // ctx_7
8    y = foo(argc); // ctx_8
9    assert(y == argc);
10 }
```

ctx_7: context calling foo at ℓ_7

ctx_8: context calling foo at ℓ_8

One Concrete Execution
(Concrete states)

One execution:

argc : 0

push calling context (calling foo at ℓ_7)

p : 3

calling context pop (returning from foo at ℓ_2)

x : 3

push calling context (calling foo at ℓ_8)

p : 0

pop calling context (returning from foo ℓ_2)

y : 0

Symbolic Execution
(Symbolic states)

One execution:

argc : getZ3Expr(argc)

push calling context (calling foo at ℓ_7)

ctx_7_p : 3

pop calling context (returning from foo at ℓ_2)

x : getZ3Expr(p, ctx_7)

push calling context (calling foo at ℓ_8)

ctx_8_p : getZ3Expr(argc)

pop calling context (returning from foo ℓ_2)

y : getZ3Expr(p, ctx_8)

Checking non-existence of counterexamples:

$\psi(N_1) \wedge \psi(N_2) \wedge \ldots \psi(N_i) \wedge \neg\psi(Q)$	Satisfiability	Counterexample
ctx_7_p $\equiv 3 \wedge$ x $\equiv$ ctx_7_p $\wedge$ ctx_8_p $\equiv$ argc $\wedge$ y $\equiv$ ctx_7_p $\wedge$ y $\neq$ argc	unsat	$\varnothing$

foo's argument p needs to be **differentiated and renamed** as ctx_7_p and ctx_8_p due to two calling contexts, ctx_7 and ctx_8 to mimic the runtime call stack which holds the local variable p.

Example 5: Interprocedural

```c
int foo(int p) {
    return p;
}
int main(int argc) {
  int x;
  int y;
  x = foo(3);
  y = foo(argc);
  svf_assert(y == argc);
}
```

Source code

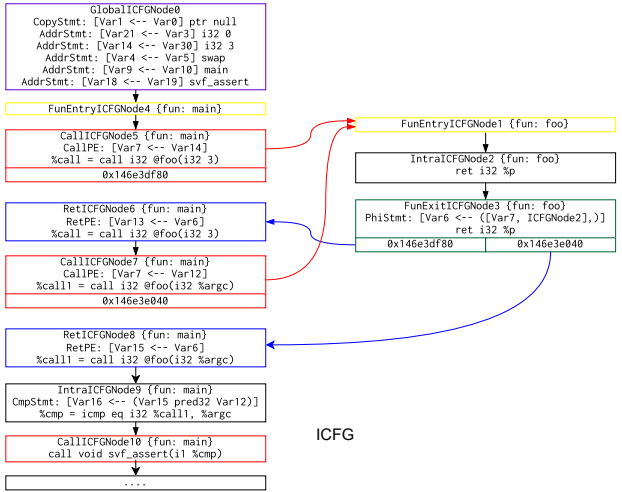
```
define i32 @foo(i32 %p) #0 {
entry:
  ret i32 %p
}

define i32 @main(i32 %argc) #0 {
entry:
  %call = call i32 @foo(i32 3)
  %call1 = call i32 @foo(i32 %argc)
  %cmp = icmp eq i32 %call1, %argc
  call void @svf_assert(i1 zeroext %cmp)
  ret i32 0
}
```

LLVM IR

Example 5: Interprocedural

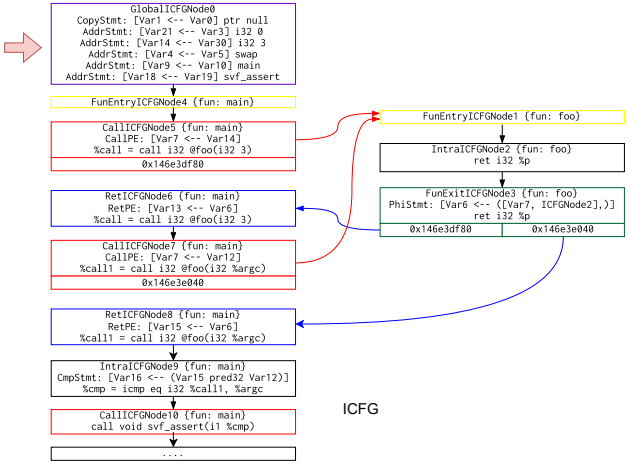
```
1  define i32 @foo(i32 %p) #0 {
2  entry:
3    ret i32 %p
4  }
5
6  define i32 @main(i32 %argc) #0 {
7  entry:
8    %call = call i32 @foo(i32 3)
9    %call1 = call i32 @foo(i32 %argc)
10   %cmp = icmp eq i32 %call1, %argc
11   call void @svf_assert(i1 zeroext %cmp)
12   ret i32 0
13 }
```

LLVM IR



ICFG

Example 5: Interprocedural

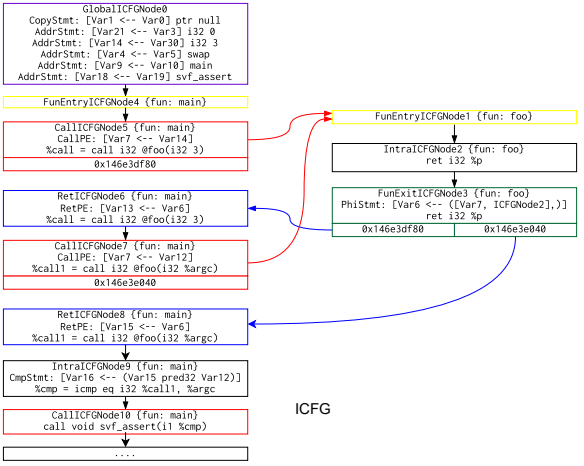


ICFG

```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ObjVar10 (0x7f00000a)    Value: NULL
ObjVar19 (0x7f000013)    Value: NULL
ValVar0                  Value: NULL
ValVar1                  Value: NULL
ValVar21                 Value: 0
ValVar14                 Value: 3
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 0x7f00000e
ValVar18                 Value: 0x7f00001d
                    ...
------------------------------------------
```

The values of Z3 expressions for each SVFVar after analyzing `GlobalICFGNode0` (use `printExprValues()` to print SVFVars and their Values)

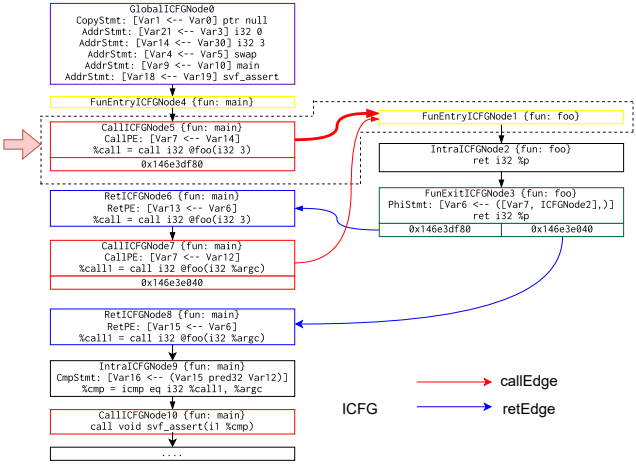
Example 5: Interprocedural



Algorithm 11: 2 `translatePath(path)`

```
2   foreach edge ∈ path do
4       if IntraEdge ← dyn_cast⟨IntraCFGEdge⟩(edge) then
6           if handleIntra(IntraEdge) == false then
8               return false;
10      else if CallEdge ← dyn_cast⟨CallCFGEdge⟩(edge) then
12          handleCall(CallEdge);
14      else if RetEdge ← dyn_cast⟨RetCFGEdge⟩(edge) then
16          handleRet(RetEdge);
18      else
20          assert(false &&"what other edges we have?");
21      return true;
```

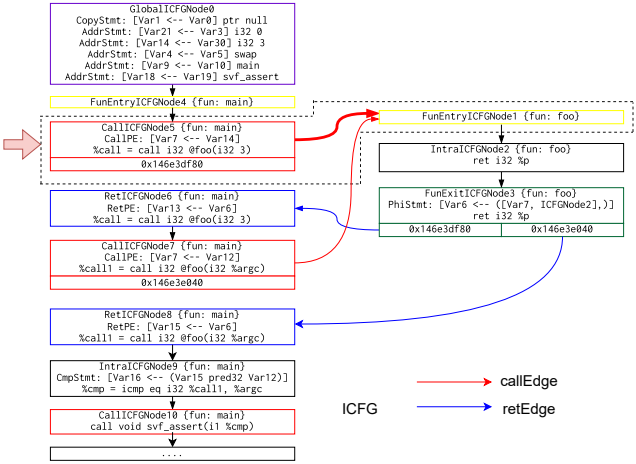
Example 5: Interprocedural

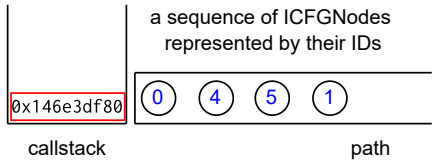


Algorithm 12: `handleCall(callEdge)`

```
1 expr_vector preCtxExprs(getCtx());
2 callPEs ← calledge→getCallPEs();
3 foreach callPE ∈ callPEs do
4     preCtxExprs.push_back(rhs);
5 pushCallingCtx(calledge→getCallSite());
6 for i = 0; i < callPEs.size(); ++i do
7     lhs ← getZ3Expr(callPEs[i]→getLHSVarID());
8     addToSolver(lhs == preCtxExprs[i]);
9 return true;
```

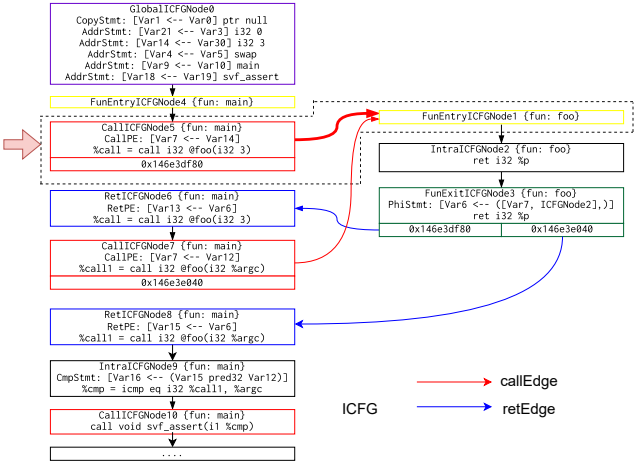
Example 5: Interprocedural





State of callstack after processing call edge between CallCFGNode5 and FunEntryICFGNode1

Example 5: Interprocedural

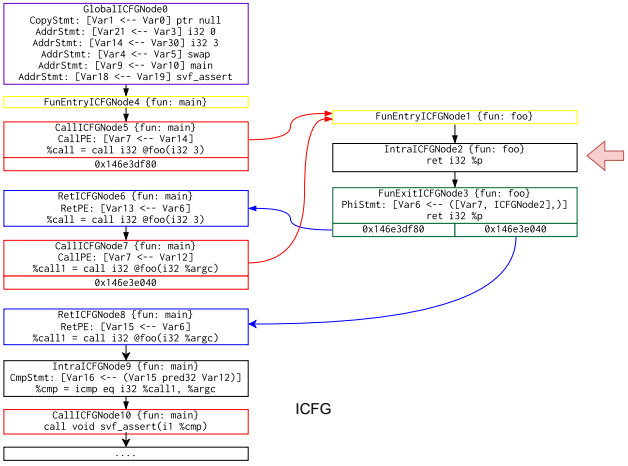



```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ObjVar10 (0x7f00000a)    Value: NULL
ObjVar19 (0x7f000013)    Value: NULL
ValVar0                  Value: NULL
ValVar1                  Value: NULL
ValVar21                 Value: 0
ValVar14                 Value: 3
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 0x7f00000e
ValVar18                 Value: 0x7f00001d
+ValVar7 (ctx:[5])       Value: 3
                  ...
------------------------------------------
```

The values of Z3 expressions for each SVFVar after analyzing `CallICFGNode5`

`ValVar7 (ctx:[5])` has value 3

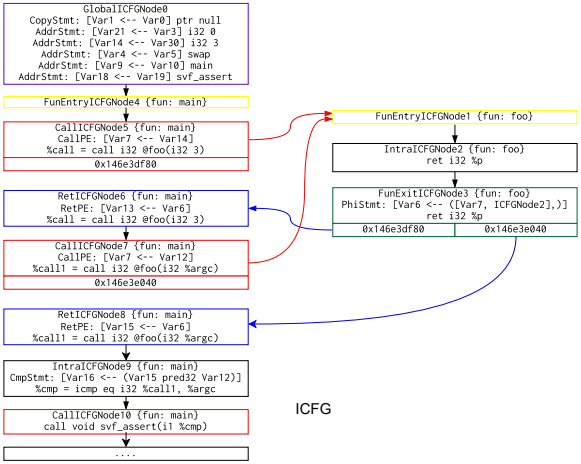
Example 5: Interprocedural



ICFG

`ret i32 %p` instruction.
Nothing needs to be done.
Continue.

Example 5: Interprocedural

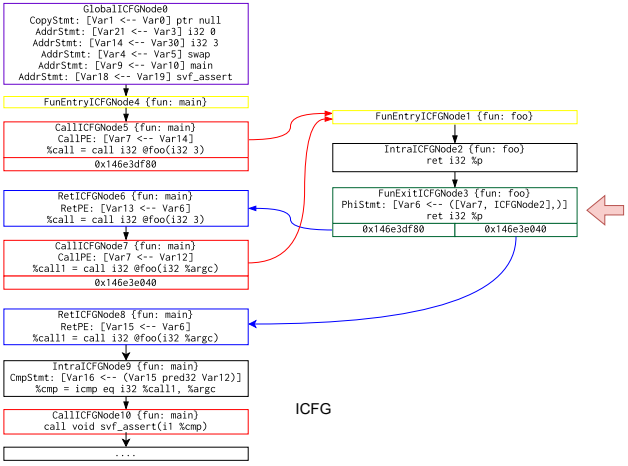


ICFG

Algorithm 13: 3 `handlePhi(intraEdge)`

```
1 res ← getZ3Expr(phi.getResID());
2 opINodeFound ← false;
3 for i ← 0 to phi.getOpVarNum() − 1 do
4     if srcNode.getFun().postDominate(srcNode.getBB(),
5     phi.getOpICFGNode(i).getBB()) then
6         ope ← getZ3Expr(phi.getOpVar(i).getId());
7         addToSolver(res == ope);
8         opINodeFound ← true;
```

Example 5: Interprocedural



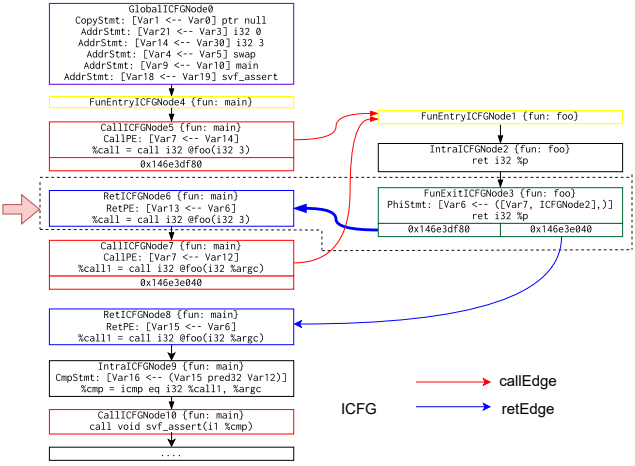
ICFG

```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ObjVar10 (0x7f00000a)    Value: NULL
ObjVar19 (0x7f000013)    Value: NULL
ValVar0                  Value: NULL
ValVar1                  Value: NULL
ValVar21                 Value: 0
ValVar14                 Value: 3
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 0x7f00000e
ValVar18                 Value: 0x7f00001d
ValVar7 (ctx:[5])        Value: 3
+ValVar6 (ctx:[5])       Value: 3
                  ...
------------------------------------------
```

The values of Z3 expressions for each SVFVar after analyzing `FuncExitICFGNode3`

`ValVar6 (ctx:[5])` has value 3

Example 5: Interprocedural

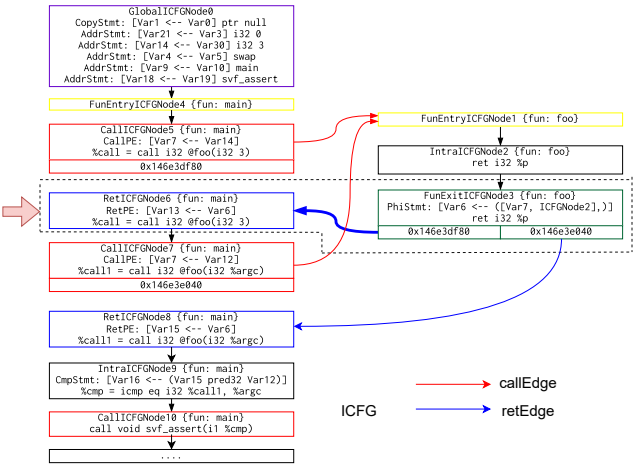


Algorithm 14: `handleRet(retEdge)`

```
1  rhs(getCtx());
2  if retPE ← retEdge.getRetPE() then
3      rhs ← getZ3Expr(retPE.getRHSVarID());
4  popCallingCtx();
5  if retPE ← retEdge.getRetPE() then
6      lhs ← getZ3Expr(retPE.getLHSVarID());
7      addToSolver(lhs == rhs);
8  return true;
```

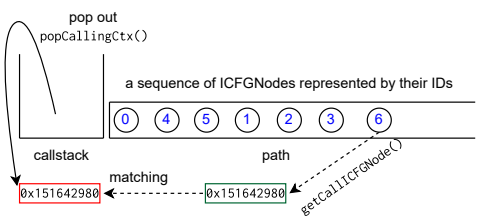
Note: `retPE.getRHSVarID()` returns `ValVar6`
`getZ3Expr(ValVar6)` binds `ValVar6` with the current context and returns the Z3 Expression for `ValVar6 (ctx:[5])`

Example 5: Interprocedural



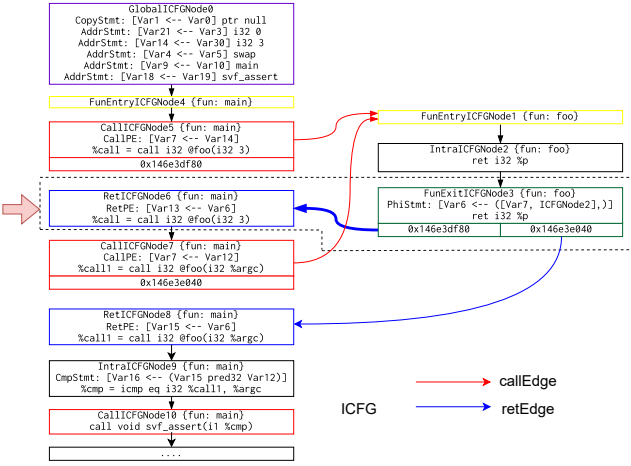
Algorithm 15: `handleRet(retEdge)`

```
1  rhs(getCtx());
2  if retPE ← retEdge.getRetPE() then
3  |  rhs ← getZ3Expr(retPE.getRHSVarID());
4  popCallingCtx();
5  if retPE ← retEdge.getRetPE() then
6  |  lhs ← getZ3Expr(retPE.getLHSVarID());
7  |  addToSolver(lhs == rhs);
8  return true;
```



State of callstack while processing return edge from FunExitICFGNode3 to RetICFGNode6

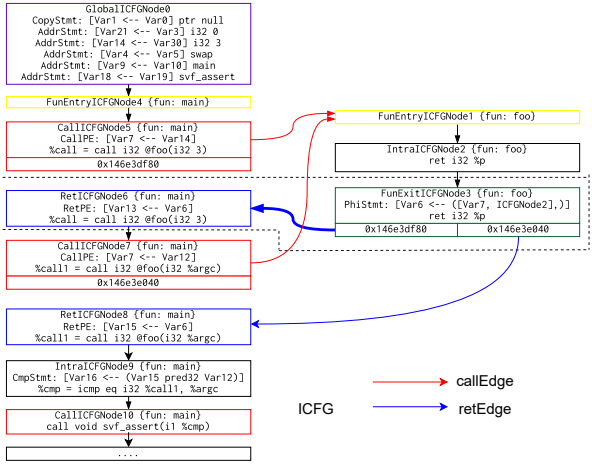
Example 5: Interprocedural



Algorithm 16: `handleRet(retEdge)`

```
1  rhs(getCtx());
2  if retPE ← retEdge.getRetPE() then
3      rhs ← getZ3Expr(retPE.getRHSVarID());
4  popCallingCtx();
5  if retPE ← retEdge.getRetPE() then
6      lhs ← getZ3Expr(retPE.getLHSVarID());
7      addToSolver(lhs == rhs);
8  return true;
```

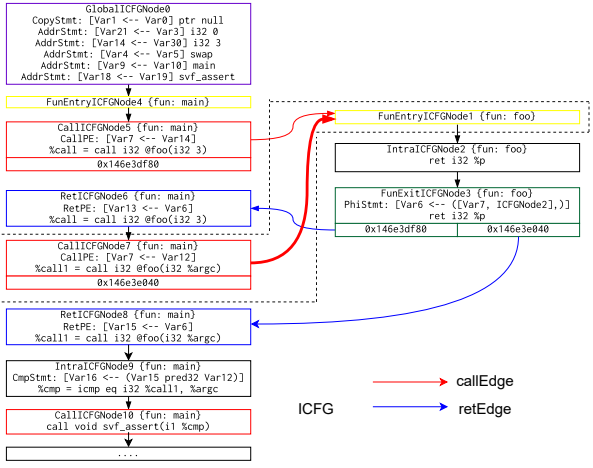
Example 5: Interprocedural




```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ObjVar10 (0x7f00000a)    Value: NULL
ObjVar19 (0x7f000013)    Value: NULL
ValVar0                  Value: NULL
ValVar1                  Value: NULL
ValVar21                 Value: 0
ValVar14                 Value: 3
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 0x7f00000e
ValVar18                 Value: 0x7f00001d
ValVar7 (ctx:[5])        Value: 3
ValVar6 (ctx:[5])        Value: 3
ValVar13                 Value: 3
                  ...
-------------------------------------------
```

The values of Z3 expressions for each SVFVar after analyzing `RetICFGNode6`

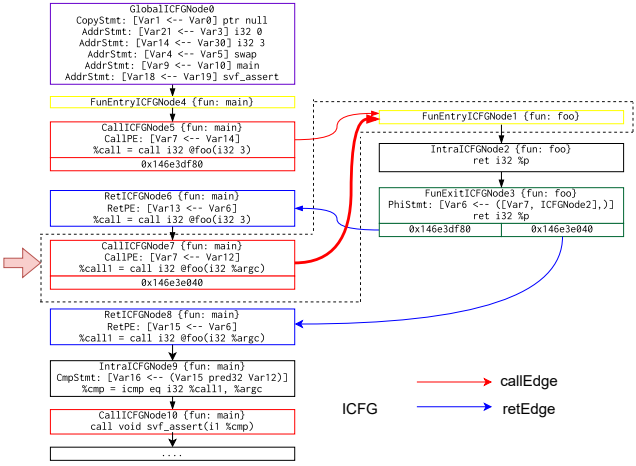
Example 5: Interprocedural

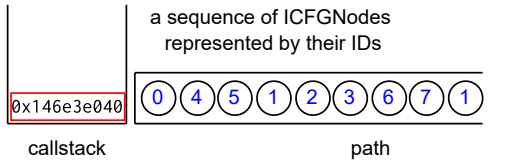


Algorithm 17: `handleCall(callEdge)`

```
1 expr_vector preCtxExprs(getCtx());
2 callPEs ← calledge→getCallPEs();
3 foreach callPE ∈ callPEs do
4     preCtxExprs.push_back(rhs);
5 pushCallingCtx(calledge→getCallSite());
6 for i = 0; i < callPEs.size(); ++i do
7     lhs ← getZ3Expr(callPEs[i]→getLHSVarID());
8     addToSolver(lhs == preCtxExprs[i]);
9 return true;
```

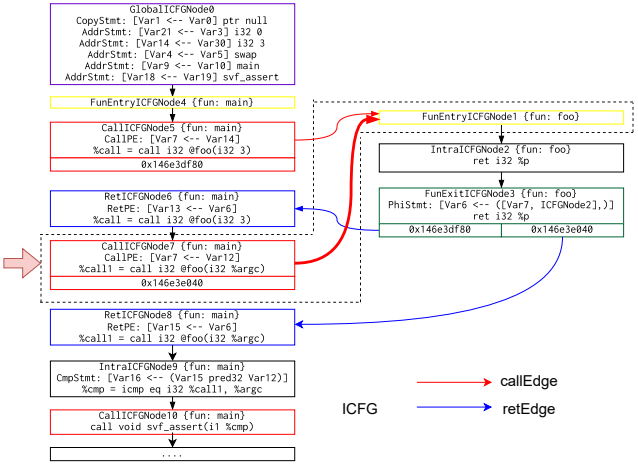
Example 5: Interprocedural





State of callstack after processing call edge between CallCFGNode7 and FunEntryICFGNode1

Example 5: Interprocedural

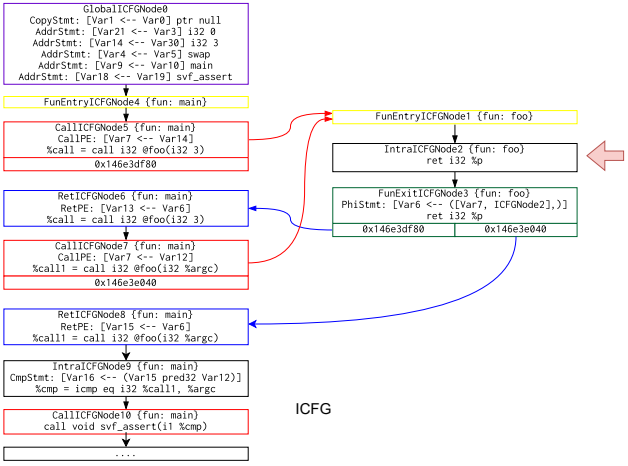



```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ObjVar10 (0x7f00000a)    Value: NULL
ObjVar19 (0x7f000013)    Value: NULL
ValVar0                  Value: NULL
ValVar1                  Value: NULL
ValVar21                 Value: 0
ValVar14                 Value: 3
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 0x7f00000e
ValVar18                 Value: 0x7f00001d
ValVar7 (ctx:[5])        Value: 3
ValVar6 (ctx:[5])        Value: 3
ValVar13                 Value 3
+ValVar12                Value: 0
+ValVar7 (ctx:[7])       Value: 0
                  ...
-----------------------------------------
```

The values of Z3 expressions for each SVFVar after analyzing `CallICFGNode7`

* `ValVar12` is not initialized in the program, thus is evaluated as 0
* `ValVar7 (ctx:[7])` has value 0

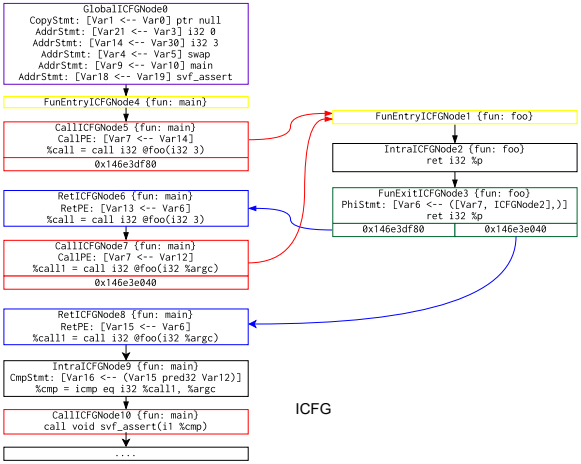
Example 5: Interprocedural



ICFG

`ret i32 %p` instruction.
Nothing needs to be done.
Continue.

Example 5: Interprocedural

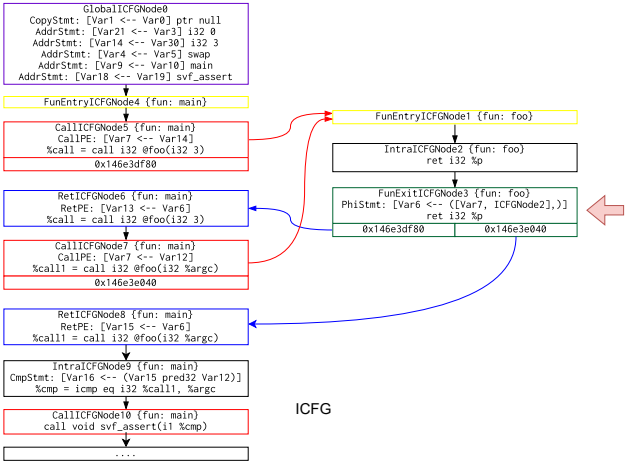


ICFG

Algorithm 18: 3 `handlePhi(intraEdge)`

```
1 res ← getZ3Expr(phi.getResID());
2 opINodeFound ← false;
3 for i ← 0 to phi.getOpVarNum() − 1 do
4     if srcNode.getFun().postDominate(srcNode.getBB(),
5     phi.getOpICFGNode(i).getBB()) then
6         ope ← getZ3Expr(phi.getOpVar(i).getId());
7         addToSolver(res == ope);
8         opINodeFound ← true;
```

Example 5: Interprocedural



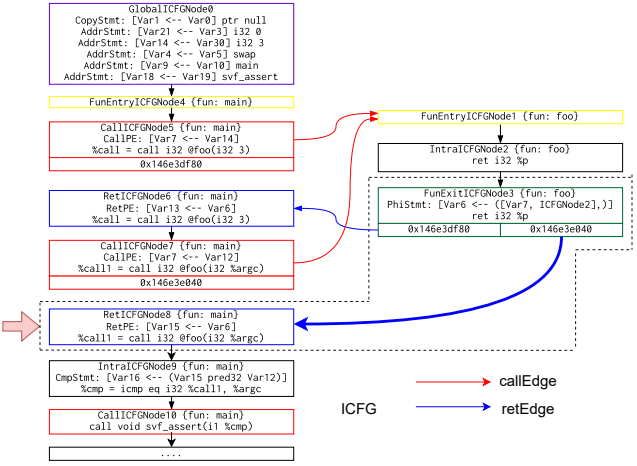
ICFG

```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ObjVar10 (0x7f00000a)    Value: NULL
ObjVar19 (0x7f000013)    Value: NULL
ValVar0                  Value: NULL
ValVar1                  Value: NULL
ValVar21                 Value: 0
ValVar14                 Value: 3
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 0x7f00000e
ValVar18                 Value: 0x7f00001d
ValVar7 (ctx:[5])        Value: 3
ValVar6 (ctx:[5])        Value: 3
ValVar13                 Value: 3
ValVar12                 Value: 0
ValVar7 (ctx:[7])        Value: 0
+ValVar6 (ctx:[7])       Value: 0
                  ...
------------------------------------------
```

The values of Z3 expressions for each SVFVar after analyzing `FuncExitICFGNode3`

`ValVar6 (ctx:[7])` has value 0

Example 5: Interprocedural

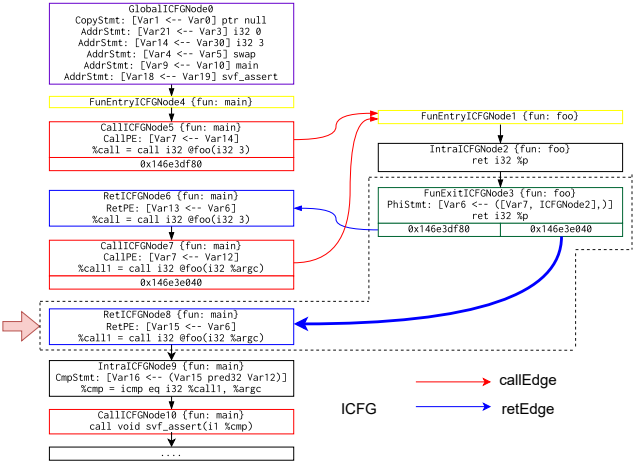


Algorithm 19: `handleRet(retEdge)`

```
1  rhs(getCtx());
2  if retPE ← retEdge.getRetPE() then
3      rhs ← getZ3Expr(retPE.getRHSVarID());
4  popCallingCtx();
5  if retPE ← retEdge.getRetPE() then
6      lhs ← getZ3Expr(retPE.getLHSVarID());
7      addToSolver(lhs == rhs);
8  return true;
```

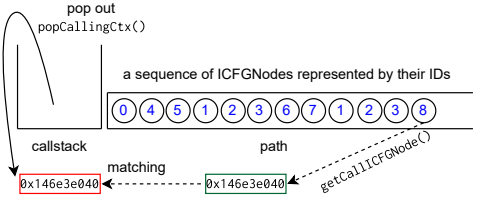
Note: `retPE.getRHSVarID()` returns `ValVar6`
`getZ3Expr(ValVar6)` binds `ValVar6` with the current context and returns the Z3 Expression for `ValVar6 (ctx:[7])`

Example 5: Interprocedural



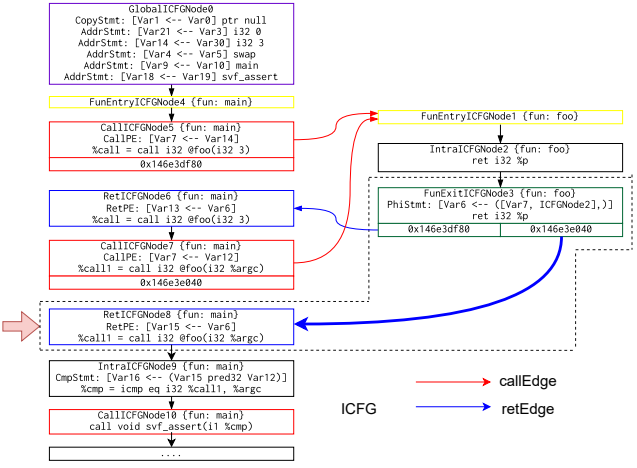
Algorithm 20: `handleRet(retEdge)`

```
1  rhs(getCtx());
2  if retPE ← retEdge.getRetPE() then
3  |  rhs ← getZ3Expr(retPE.getRHSVarID());
4  popCallingCtx();
5  if retPE ← retEdge.getRetPE() then
6  |  lhs ← getZ3Expr(retPE.getLHSVarID());
7  |  addToSolver(lhs == rhs);
8  return true;
```



State of callstack while processing return edge from FunExitICFGNode3 to RetICFGNode8

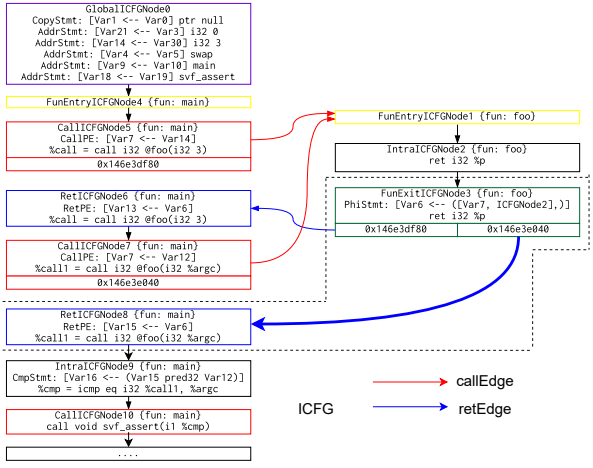
Example 5: Interprocedural



Algorithm 21: `handleRet(retEdge)`

```
1  rhs(getCtx());
2  if retPE ← retEdge.getRetPE() then
3      rhs ← getZ3Expr(retPE.getRHSVarID());
4  popCallingCtx();
5  if retPE ← retEdge.getRetPE() then
6      lhs ← getZ3Expr(retPE.getLHSVarID());
7      addToSolver(lhs == rhs);
8  return true;
```

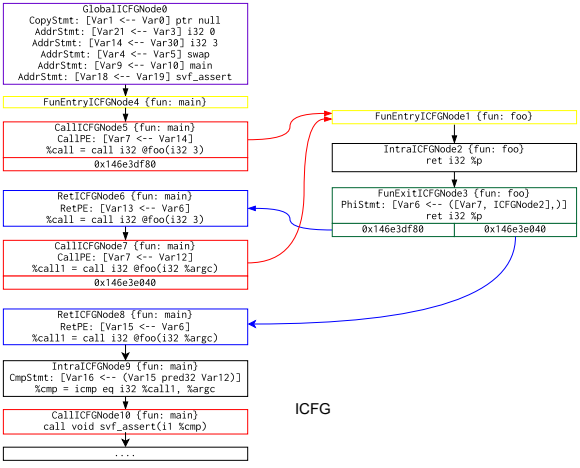
Example 5: Interprocedural




```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ObjVar10 (0x7f00000a)    Value: NULL
ObjVar19 (0x7f000013)    Value: NULL
ValVar0                  Value: NULL
ValVar1                  Value: NULL
ValVar21                 Value: 0
ValVar14                 Value: 3
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 0x7f00000e
ValVar18                 Value: 0x7f00001d
ValVar7 (ctx:[5])        Value: 3
ValVar6 (ctx:[5])        Value: 3
ValVar13                 Value: 3
ValVar12                 Value: 0
ValVar7 (ctx:[7])        Value: 0
ValVar6 (ctx:[7])        Value: 0
+ValVar15                Value: 0
                  ...
------------------------------------------
```

The values of Z3 expressions for each SVFVar after analyzing `RetICFGNode8`

Example 5: Interprocedural

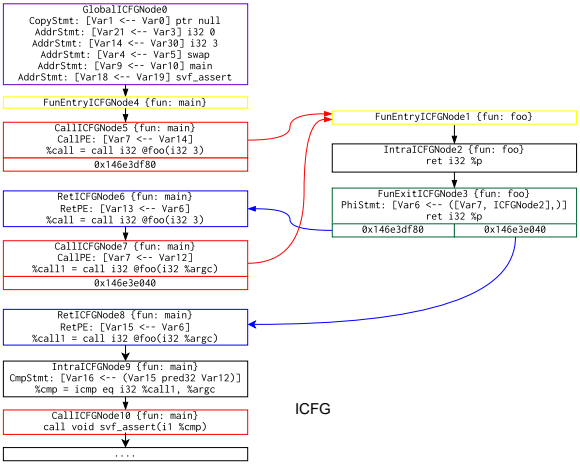


ICFG

```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ObjVar10 (0x7f00000a)    Value: NULL
ObjVar19 (0x7f000013)    Value: NULL
ValVar0                  Value: NULL
ValVar1                  Value: NULL
ValVar21                 Value: 0
ValVar14                 Value: 3
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 0x7f00000e
ValVar18                 Value: 0x7f00001d
ValVar7 (ctx:[5])        Value: 3
ValVar6 (ctx:[5])        Value: 3
ValVar13                 Value: 3
ValVar12                 Value: 0
ValVar7 (ctx:[7])        Value: 0
ValVar6 (ctx:[7])        Value: 0
ValVar15                 Value: 0
+ValVar16                Value: 1
                  ...
------------------------------------------
```

The values of Z3 expressions for each SVFVar after analyzing `IntraICFGNode9`

Example 5: Interprocedural



ICFG

```
-------------SVFVar and Value-------------
ObjVar5 (0x7f000005)     Value: NULL
ObjVar10 (0x7f00000a)    Value: NULL
ObjVar19 (0x7f000013)    Value: NULL
ValVar0                  Value: NULL
ValVar1                  Value: NULL
ValVar21                 Value: 0
ValVar14                 Value: 3
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 0x7f00000e
ValVar18                 Value: 0x7f00001d
ValVar7 (ctx:[5])        Value: 3
ValVar6 (ctx:[5])        Value: 3
ValVar13                 Value: 3
ValVar12                 Value: 0
ValVar7 (ctx:[7])        Value: 0
ValVar6 (ctx:[7])        Value: 0
ValVar15                 Value: 0
ValVar16                 Value: 1
                  ...
------------------------------------------
```

The assertion is successfully verified!!

START: 0 → 4 → 5 → 1 → 2 → 3 → 6 → 7 → 1 → 2

→ 3 → 8 → 9 →→ *svf_assert*

What's next?

- (1) Understand SSE algorithms and examples in the slides
- (2) Finish implementing the automated translation from code to Z3 formulas using `SSE` and `Z3SSEMgr` in Assignment 2