Lab: Software Verification and Z3 Theorem Prover

(Week 7)

Yulei Sui

School of Computer Science and Engineering

University of New South Wales, Australia

Quiz-2 + Lab-Exercise-2 + Assignment-2

- A set of quizzes on WebCMS (5 points)
  - Logical formula and predicate logic
  - Z3's knowledge and translation rules
- Lab-Exercise-2 (5 points)
  - **Goal:** Manually translate code into z3 formulas/constraints and verify the assertions embedded in the code.
  - **Specification:** `https://github.com/SVF-tools/Software-Security-Analysis/wiki/Lab-Exercise-2`
  - **SVF Z3 APIs:** `https://github.com/SVF-tools/Software-Security-Analysis/wiki/SVF-Z3-API`
- Assignment-2 (25 points)
  - **Goal:** automatically perform assertion-based verification for code using static symbolic execution.
  - **Specification:** `https://github.com/SVF-tools/Software-Security-Analysis/wiki/Assignment-2`

Methods to Be Implemented

You need to implement the following four functions in `Assignment-4.cpp`:

- `SSE::translatePath`
- `SSE::handleNonBranch`
- `SSE::handleCall`
- `SSE::handleRet`
- `SSE::handleBranch`
- Remember to put your previously implemented Assignment-2.cpp in place (under the Assignment-2 folder).
- The required implementation parts are indicated with TODO comments and you only need to fill up the code template if a method is partially implemented.

In the following slides, we provide several examples to assist your understanding of SSE.

Interprocedural Example

```c
void foo(int* p) {
  *p = 1;
}
int main() {
    int a = 0;
    foo(&a);
    svf_assert(a == 1);
}
```

compile

```
void @foo(i32* %p) {
entry:
  store i32 1, i32* %p
  ret void
}
i32 @main() {
entry:
  %a = alloca i32
  store i32 0, i32* %a
  call void @foo(i32* %a)
  %0 = load i32, i32* %a
  %cmp = icmp eq i32 %0, 1
  call void @svf_assert(i1 zeroext %cmp)
  ret i32 0
}
```

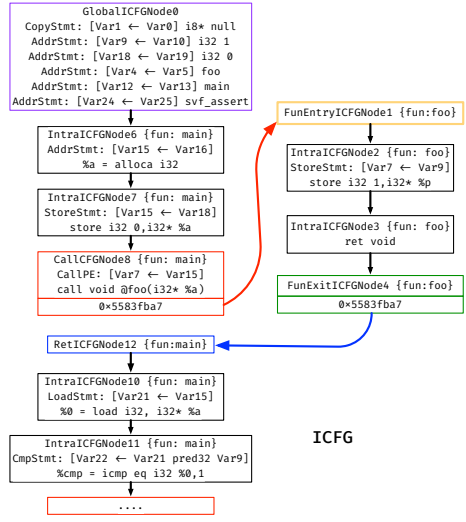
Interprocedural Example

```
void foo(int* p) {
  *p = 1;
}
int main() {
    int a = 0;
    foo(&a);
    svf_assert(a == 1);
}
```

compile

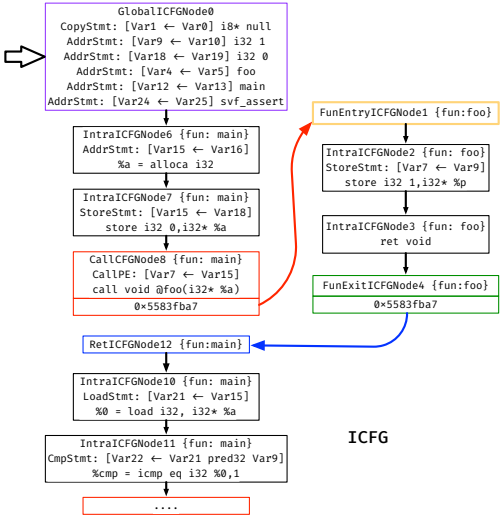
```
void @foo(i32* %p) {
entry:
  store i32 1, i32* %p
  ret void
}
i32 @main() {
entry:
  %a = alloca i32
  store i32 0, i32* %a
  call void @foo(i32* %a)
  %0 = load i32, i32* %a
  %cmp = icmp eq i32 %0, 1
  call void @svf_assert(i1 zeroext %cmp)
  ret i32 0
}
```

SVF



ICFG

Interprocedural Example

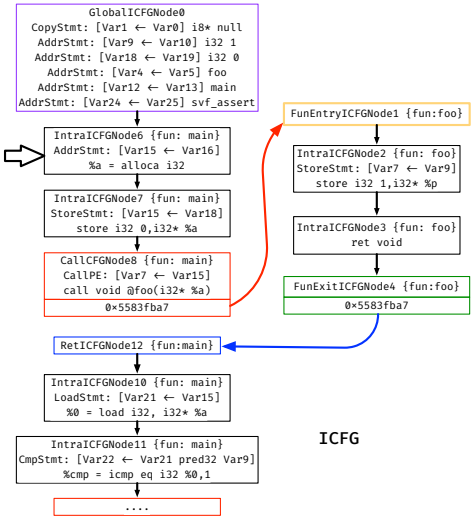


ICFG

```
-------------SVFVar and Value-------------
ObjVar25 (0x7f000019)    Value: NULL
ObjVar19 (0x7f000013)    Value: 0
ObjVar16 (0x7f000010)    Value: NULL
ObjVar13 (0x7f00000d)    Value: NULL
ObjVar10 (0x7f00000a)    Value: 1
ObjVar5 (0x7f000005)     Value: NULL
ValVar24                 Value: 0x7f000019
ObjVar2 (0x7f000002)     Value: NULL
ObjVar3 (0x7f000003)     Value: NULL
ValVar1                  Value: 2
ValVar0                  Value: 2
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 1
ValVar12                 Value: 0x7f00000d
ValVar18                 Value: 0
                  ...
-----------------------------------------
```

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

Interprocedural Example

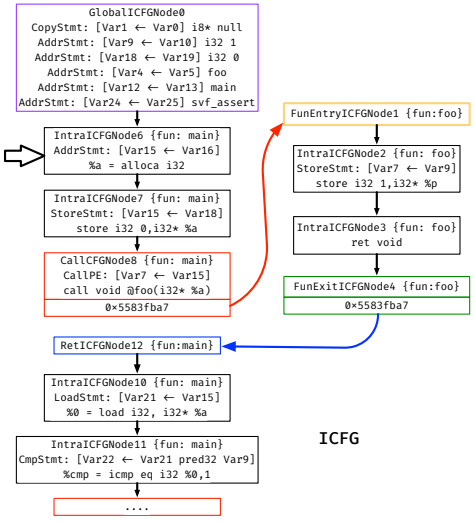


ICFG

Algorithm 1: 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;
```

Interprocedural Example



ICFG

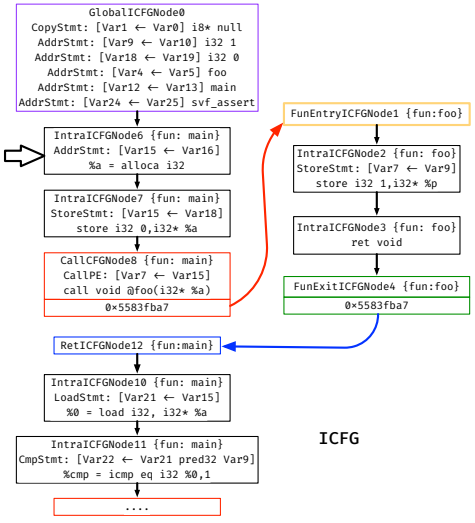
Algorithm 2: 3 `handleIntra(intraEdge)`

```
2  if intraEdge.getCondition() && !handleBranch(intraEdge)
     then
4      return false;
6  else
8      handleNonBranch(edge);
9
```

Algorithm 2: `HandleNonBranch(intraEdge)`

```
2   dst ← intraEdge.getDstNode();
    src ← intraEdge.getSrcNode();
4   foreach stmt ∈ dst.getSVFStmts() do
5     if addr ∈ dyn_cast⟨AddrStmt⟩(stmt) then
8       obj ← getMemObjAddress(addr.getRHSVarID());
10      lhs ← getZ3Expr(addr.getLHSVarID());
12      addToSolver(obj == lhs);
18    else if copy ∈ dyn_cast⟨CopyStmt⟩(stmt) then
16      lhs ← getZ3Expr(copy.getLHSVarID());
18      rhs ← getZ3Expr(copy.getRHSVarID());
20      addToSolver(rhs == lhs);
22    else if load ∈ dyn_cast⟨LoadStmt⟩(stmt) then
24      lhs ← getZ3Expr(load.getLHSVarID());
26      rhs ← getZ3Expr(load.getRHSVarID());
28      addToSolver(lhs == z3Mgr.loadValue(rhs));
29    ...
```

Interprocedural Example



ICFG

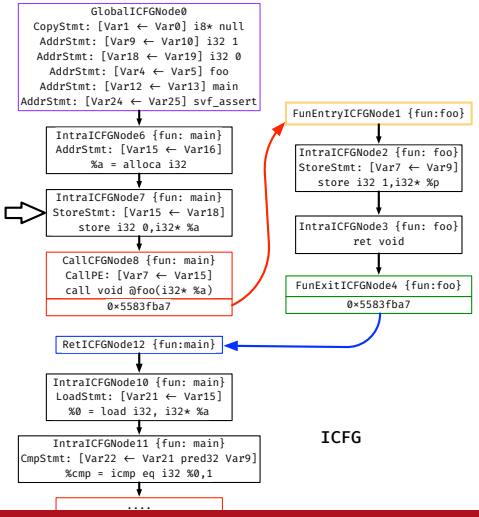
```
-------------SVFVar and Value-------------
ObjVar25 (0x7f000019)    Value: NULL
ObjVar19 (0x7f000013)    Value: 0
ObjVar16 (0x7f000010)     Value:  NULL
ObjVar13 (0x7f00000d)    Value: NULL
ObjVar10 (0x7f00000a)    Value: 1
ObjVar5 (0x7f000005)     Value: NULL
ValVar24                 Value: 0x7f000019
ObjVar2 (0x7f000002)     Value: NULL
ObjVar3 (0x7f000003)     Value: NULL
ValVar1                  Value: 2
ValVar0                  Value: 2
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 1
ValVar12                 Value: 0x7f00000d
ValVar18                 Value: 0
+ValVar15               Value:  0x7f000010
                    ...
------------------------------------------
```

Analyzing IntraICFGNode6 {fun: main}

AddrStmt: [Var14 ← Var15]

`%a = alloca i32`

Interprocedural Example



ICFG

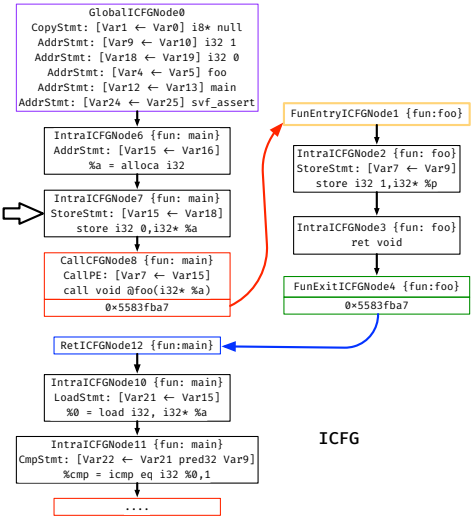
Algorithm 3: 3 `handleIntra(intraEdge)`

```
2  if intraEdge.getCondition() && !handleBranch(intraEdge)
     then
4      return false;
6  else
8      handleNonBranch(edge);
9
```

Algorithm 3: `HandleNonBranch(intraEdge)`

```
11   dst ← intraEdge.getDstNode();
     src ← intraEdge.getSrcNode();
12   foreach stmt ∈ dst.getSVFStmts() do
13     ...
15     else if load ∈ dyn_cast⟨LoadStmt⟩(stmt) then
17         lhs ← getZ3Expr(load.getLHSVarID());
19         rhs ← getZ3Expr(load.getRHSVarID());
21         addToSolver(lhs == z3Mgr.loadValue(rhs));
23     else if store ∈ dyn_cast⟨StoreStmt⟩(stmt) then
25         lhs ← getZ3Expr(store.getLHSVarID());
27         rhs ← getZ3Expr(store.getRHSVarID());
29         z3Mgr.storeValue(lhs, rhs);
30     else if gep ∈ dyn_cast⟨GepStmt⟩(stmt) then
33         lhs ← getZ3Expr(gep.getLHSVarID());
35         rhs ← getZ3Expr(gep.getRHSVarID());
37         offset = z3Mgr.getGepOffset(gep);
39         gepAddress = z3Mgr.getGepObjAddress(rhs, offset);
41         addToSolver(lhs == gepAddress);
```

Interprocedural Example



ICFG

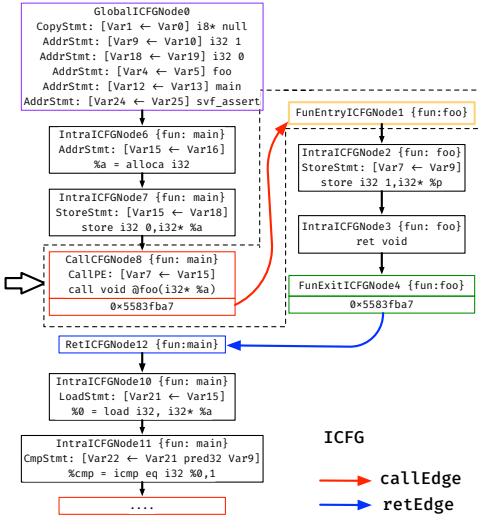
```
-------------SVFVar and Value-------------
ObjVar25 (0x7f000019)    Value: NULL
ObjVar19 (0x7f000013)    Value: 0
ObjVar16 (0x7f000010)     Value:  0
ObjVar13 (0x7f00000d)    Value: NULL
ObjVar10 (0x7f00000a)    Value: 1
ObjVar5 (0x7f000005)     Value: NULL
ValVar24                 Value: 0x7f000019
ObjVar2 (0x7f000002)     Value: NULL
ValVar15                Value:  0x7f000010
ObjVar3 (0x7f000003)     Value: NULL
ValVar1                  Value: 2
ValVar0                  Value: 2
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 1
ValVar12                 Value: 0x7f00000d
+ValVar18               Value:  0
                    ...
-----------------------------------------
```

Analyzing IntraICFGNode6 {fun: main}

StoreStmt: [Var15 ← Var18]

`store i32 0, i32 * %a`

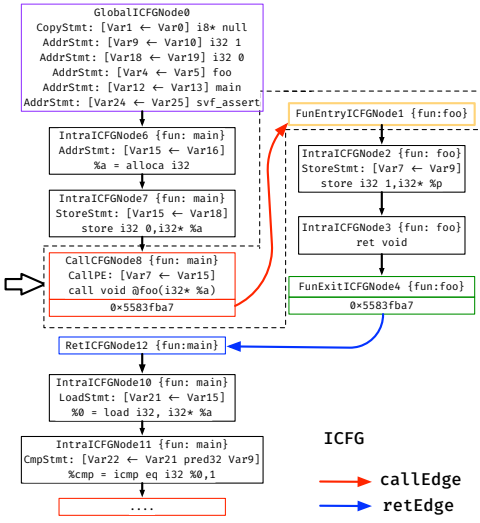
Interprocedural Example

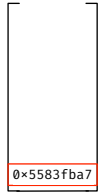


Algorithm 4: 4 `handleCall(callEdge)`

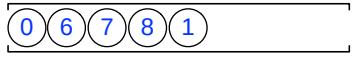
```
2   callNode ← callEdge.getSrcNode();;
4   FunEntryNode ←callEdge.getDstNode();
6   getSolver().push();
8   foreach callPE ∈ calledge.getCallPEs() do
10      lhs ← getZ3Expr(callPE.getLHSVarID());;
12      rhs ← getZ3Expr(callPE.getRHSVarID());;
14      addToSolver(lhs == rhs);;
16  return true;;
```

Interprocedural Example





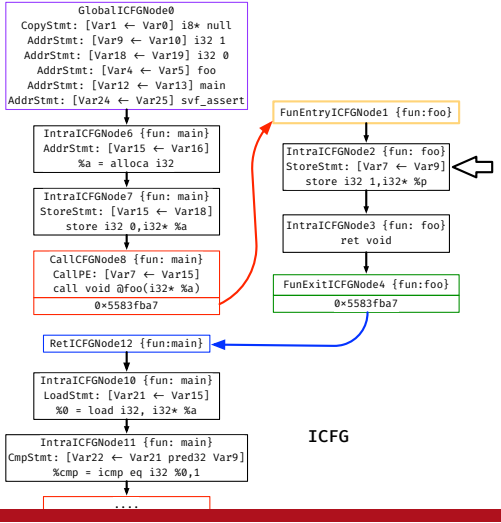
callstack



path (a sequence of ICFGNodes represented by their IDs)

State of callstack after processing call edge between CallCFGNode8 and FunEntryICFGNode1

Interprocedural Example



ICFG

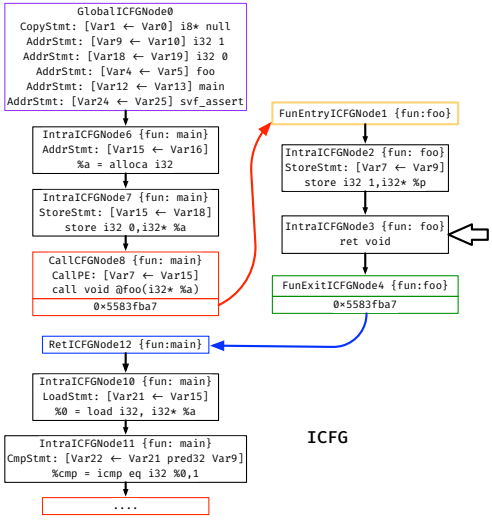
Algorithm 5: 3 `handleIntra(intraEdge)`

```
2  if intraEdge.getCondition() && !handleBranch(intraEdge)
     then
4      return false;
6  else
8      handleNonBranch(edge);
9
```

Algorithm 5: `HandleNonBranch(intraEdge)`

```
11   dst ← intraEdge.getDstNode();
     src ← intraEdge.getSrcNode();
12   foreach stmt ∈ dst.getSVFStmts() do
13     ...
15     else if load ∈ dyn_cast⟨LoadStmt⟩(stmt) then
17         lhs ← getZ3Expr(load.getLHSVarID());
19         rhs ← getZ3Expr(load.getRHSVarID());
21         addToSolver(lhs == z3Mgr.loadValue(rhs));
23     else if store ∈ dyn_cast⟨StoreStmt⟩(stmt) then
25         lhs ← getZ3Expr(store.getLHSVarID());
27         rhs ← getZ3Expr(store.getRHSVarID());
29         z3Mgr.storeValue(lhs, rhs);
30     else if gep ∈ dyn_cast⟨GepStmt⟩(stmt) then
33         lhs ← getZ3Expr(gep.getLHSVarID());
35         rhs ← getZ3Expr(gep.getRHSVarID());
37         offset = z3Mgr.getGepOffset(gep);
39         gepAddress = z3Mgr.getGepObjAddress(rhs, offset);
41         addToSolver(lhs == gepAddress);
```

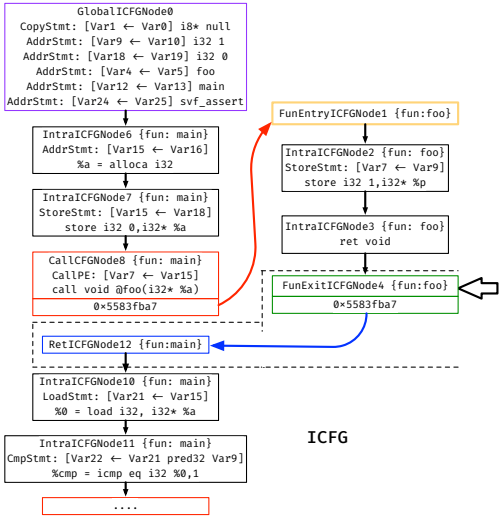
Interprocedural Example



ICFG

`ret void` instruction.
Nothing needs to be done.
Continue.

Interprocedural Example

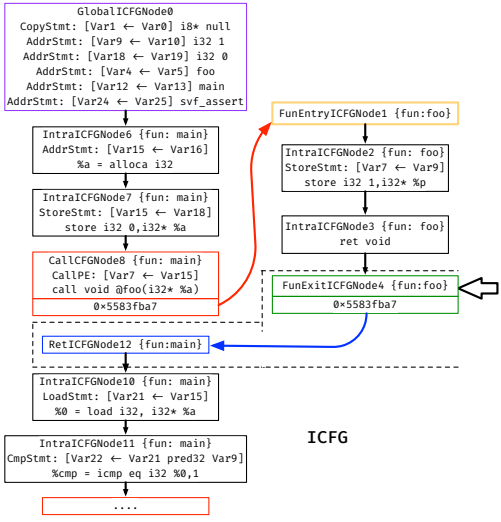


ICFG

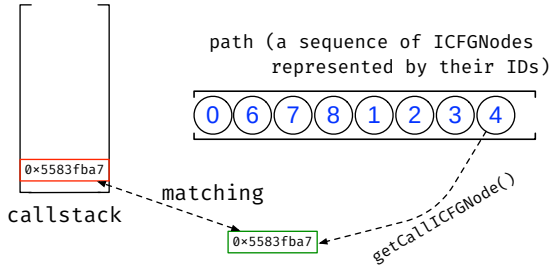
Algorithm 6: 5 `handleRet(retEdge)`

```
2   rhs(getCtx());;
4   if retPE ← retEdge.getRetPE() then
6       rhs ← getEvalExpr(getZ3Expr(retPE.getRHSVarID()));;
8   getSolver().pop();;
10  if retPE ← retEdge.getRetPE() then
12      lhs ← getZ3Expr(retPE.getLHSVarID());;
14      addToSolver(lhs == rhs);;
16  return true;;
```

Interprocedural Example

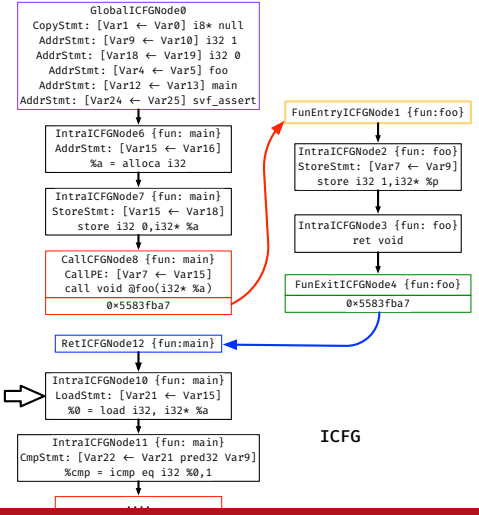


ICFG



State of callstack while processing return edge from FunExitICFGNode4 to RetICFGNode12

Interprocedural Example



ICFG

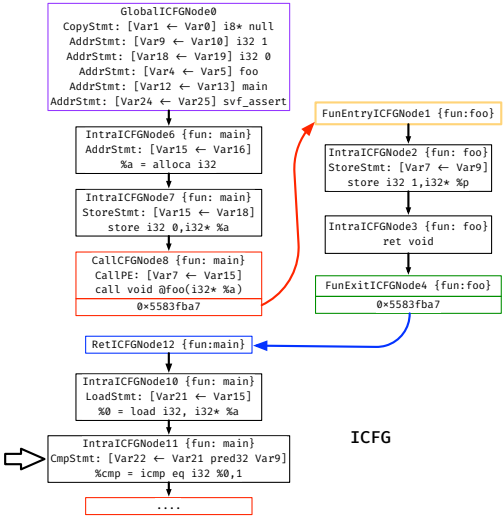
Algorithm 7: 3 `handleIntra(intraEdge)`

```
2  if intraEdge.getCondition() && !handleBranch(intraEdge)
     then
4      return false;
6  else
8      handleNonBranch(edge);
9
```

Algorithm 7: `HandleNonBranch(intraEdge)`

```
11   dst ← intraEdge.getDstNode();
     src ← intraEdge.getSrcNode();
12   foreach stmt ∈ dst.getSVFStmts() do
13     ...
15     else if load ∈ dyn_cast⟨LoadStmt⟩(stmt) then
17         lhs ← getZ3Expr(load.getLHSVarID());
19         rhs ← getZ3Expr(load.getRHSVarID());
21         addToSolver(lhs == z3Mgr.loadValue(rhs));
22     else if store ∈ dyn_cast⟨StoreStmt⟩(stmt) then
25         lhs ← getZ3Expr(store.getLHSVarID());
27         rhs ← getZ3Expr(store.getRHSVarID());
29         z3Mgr.storeValue(lhs, rhs);
31     else if gep ∈ dyn_cast⟨GepStmt⟩(stmt) then
33         lhs ← getZ3Expr(gep.getLHSVarID());
35         rhs ← getZ3Expr(gep.getRHSVarID());
37         offset = z3Mgr.getGepOffset(gep);
39         gepAddress = z3Mgr.getGepObjAddress(rhs, offset);
41         addToSolver(lhs == gepAddress);
```

Interprocedural Example



ICFG

```
-------------SVFVar and Value-------------
ObjVar25 (0x7f000019)    Value: NULL
ObjVar19 (0x7f000013)    Value: 0
ObjVar16 (0x7f000010)    Value: 1
ObjVar13 (0x7f00000d)    Value: NULL
ObjVar10 (0x7f00000a)    Value: 1
ObjVar5 (0x7f000005)     Value: NULL
ValVar24                 Value: 0x7f000019
ValVar22                 Value: 1
ValVar21                 Value: 1
ObjVar2 (0x7f000002)     Value: NULL
ValVar15                 Value: 0x7f000010
ObjVar3 (0x7f000003)     Value: NULL
ValVar1                  Value: 2
ValVar0                  Value: 2
ValVar4                  Value: 0x7f000005
ValVar9                  Value: 1
ValVar12                 Value: 0x7f00000d
ValVar18                 Value: 0
-----------------------------------------
```

The assertion is successfully verified!!

START: 0 → 6 → 7 → 8 → 1 → 2 → 3 → 4 → 12 → 10 → 11 → … → *END*

Branch Example

```c
int main(){
    int x = 1, y = 1;
    int a = 1, b = 2;
    if (a > b) {
        y++;
    } else {
        x++;
        svf_assert (x == 2);
    }
    return 0;
}
```



```
i32 @main() {
entry:
  %cmp = icmp sgt i32 1, 2
  br i1 %cmp, label %if.then, label %if.else
if.then:
  %inc = add nsw i32 1, 1
  br label %if.end
if.else:
  %inc1 = add nsw i32 1, 1
  %cmp2 = icmp eq i32 %inc1, 2
  call void @svf_assert(i1 zeroext %cmp2)
  br label %if.end
if.end:
ret i32 0
}
```

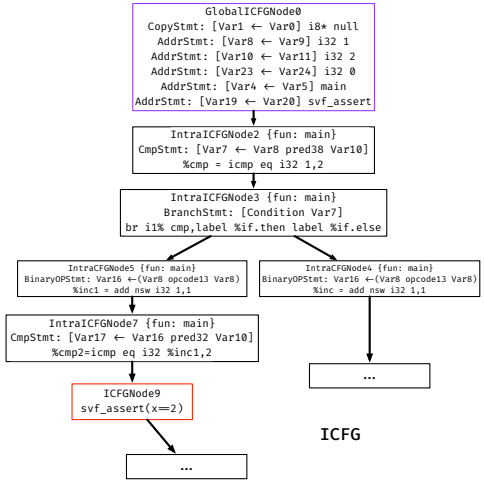
Branch Example

```c
int main(){
    int x = 1, y = 1;
    int a = 1, b = 2;
    if (a > b) {
        y++;
    } else {
        x++;
        svf_assert (x == 2);
    }
    return 0;
}
```

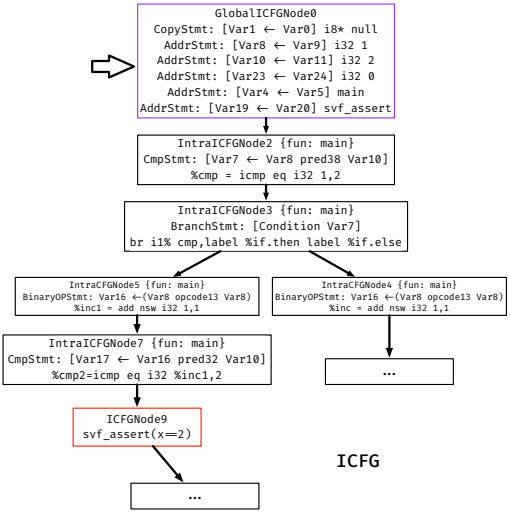
compile

```
i32 @main() {
entry:
  %cmp = icmp sgt i32 1, 2
  br i1 %cmp, label %if.then, label %if.else
if.then:
  %inc = add nsw i32 1, 1
  br label %if.end
if.else:
  %inc1 = add nsw i32 1, 1
  %cmp2 = icmp eq i32 %inc1, 2
  call void @svf_assert(i1 zeroext %cmp2)
  br label %if.end
if.end:
ret i32 0
}
```

SVF



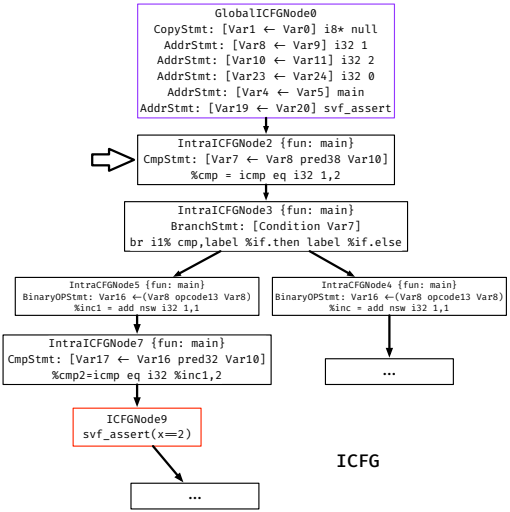
Branch Example




```
-------------SVFVar and Value-------------
ObjVar20 (0x7f000014)    Value: NULL
ObjVar24 (0x7f000018)    Value: 0
ObjVar11 (0x7f00000b)    Value: 2
ObjVar9 (0x7f000009)     Value: 1
ObjVar5 (0x7f000005)     Value: NULL
ValVar19                 Value: 0x7f000014
ValVar23                 Value: 0
ObjVar2 (0x7f000002)     Value: NULL
ObjVar3 (0x7f000003)     Value: NULL
ValVar1                  Value: 3
ValVar0                  Value: 3
ValVar4                  Value: 0x7f000005
ValVar8                  Value: 1
ValVar10                 Value: 2
                  ...
-----------------------------------------
```

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

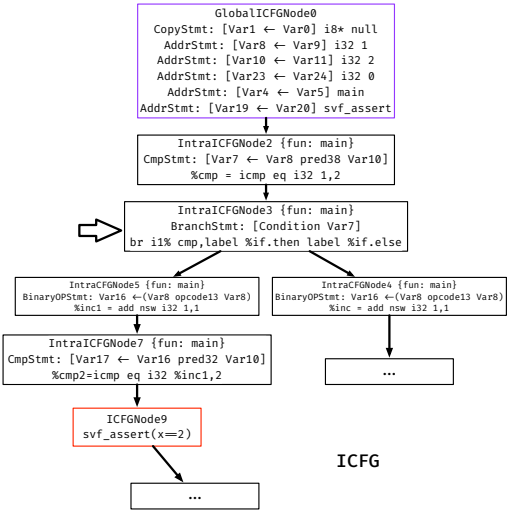
Branch Example




```
## Analyzing IntraICFGNode2 {fun: main}
CmpStmt: [Var7 <-- (Var8 predicate38 Var10)]
   %cmp = icmp sgt i32 1, 2
==> (not (<= ValVar8 ValVar10))
==> (= ValVar7 0)
                    ...
```

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

Branch Example



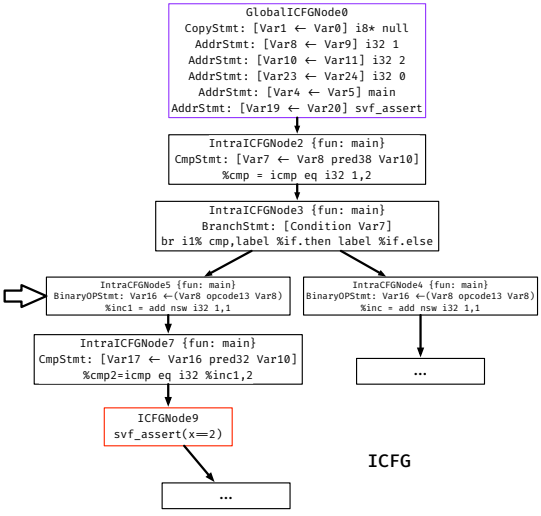
Algorithm 8: 3 `handleIntra(intraEdge)`

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

Algorithm 8: `handleBranch(intraEdge)`

```
2    cond = intraEdge.getCondition();
4    successorVal = intraEdge.getSuccessorCondValue();
6    res = getEvalExpr(cond == suc);
8  if res.is_false() then
10     addToSolver(cond! = suc);
12     return false;
14 else if res.is_true() then
16     addToSolver(cond == suc);
18     return true;
20 else
22     return true;
```

Branch Example



ICFG

```
------------SVFVar and Value------------
ObjVar20 (0x7f000014)    Value: NULL
ObjVar24 (0x7f000018)    Value: 0
ObjVar11 (0x7f00000b)    Value: 2
ObjVar9 (0x7f000009)     Value: 1
ObjVar5 (0x7f000005)     Value: NULL
ValVar19                 Value: 0x7f000014
ValVar23                 Value: 0
ObjVar2 (0x7f000002)     Value: NULL
ObjVar3 (0x7f000003)     Value: NULL
ValVar1                  Value: 3
ValVar0                  Value: 3
ValVar4                  Value: 0x7f000005
ValVar8                  Value: 1
ValVar10                 Value: 2
ValVar7                  Value: 0
                  ...
----------------------------------------
```

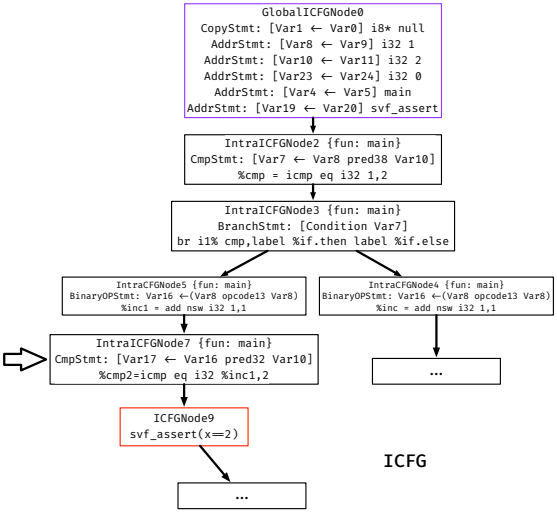
Branch IntraCFGEdge: [`ICFGNode5` ← `ICFGNode3`]

branchCondition: `%cmp = icmp sgt i32 1, 2`

(= `ValVar7 0`)

This conditional ICFGEdge is **feasible**!!

Branch Example

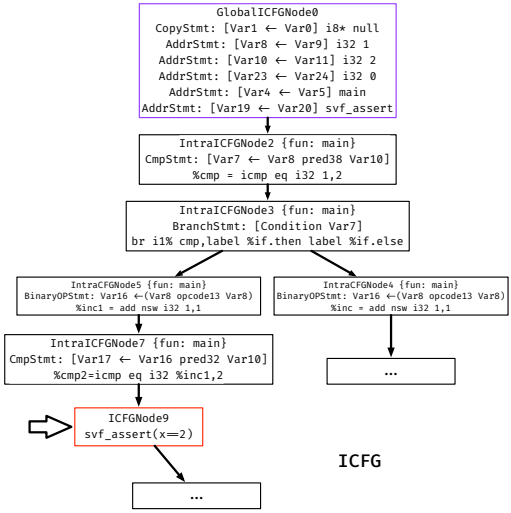



```
-------------SVFVar and Value-------------
ObjVar20 (0x7f000014)    Value: NULL
ObjVar24 (0x7f000018)     Value: 0
ObjVar11 (0x7f00000b)    Value: 2
ObjVar9 (0x7f000009)     Value: 1
ObjVar5 (0x7f000005)     Value: NULL
ValVar19                 Value: 0x7f000014
ValVar23                 Value: 0
ValVar17                 Value: 1
ObjVar2 (0x7f000002)     Value: NULL
ObjVar3 (0x7f000003)     Value: NULL
ValVar16                 Value: 2
ValVar1                  Value: 3
ValVar0                  Value: 3
ValVar4                  Value: 0x7f000005
ValVar8                  Value: 1
ValVar10                 Value: 2
ValVar7                  Value: 0
                  ...
-----------------------------------------
```

Analyzing IntraICFGNode7 fun: main

CmpStmt: [Var17 ← (Var16 predicate32 Var10)]

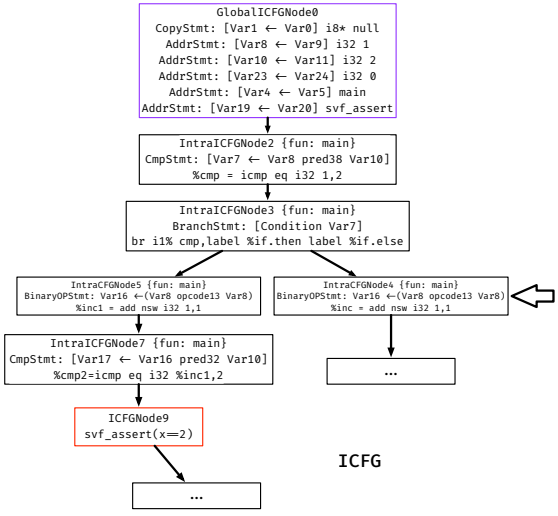
Branch Example



The assertion is successfully verified!!

START: $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 9 \rightarrow$ *END*

Branch Example




```
------------SVFVar and Value------------
ObjVar20 (0x7f000014)    Value: NULL
ObjVar24 (0x7f000018)    Value: 0
ObjVar11 (0x7f00000b)    Value: 2
ObjVar9 (0x7f000009)     Value: 1
ObjVar5 (0x7f000005)     Value: NULL
ValVar19                 Value: 0x7f000014
ValVar23                 Value: 0
ObjVar2 (0x7f000002)     Value: NULL
ObjVar3 (0x7f000003)     Value: NULL
ValVar1                  Value: 3
ValVar0                  Value: 3
ValVar4                  Value: 0x7f000005
ValVar8                  Value: 1
ValVar10                 Value: 2
ValVar7                  Value: 0
                  ...
----------------------------------------
```

Branch IntraCFGEdge: [ICFGNode4 ← ICFGNode3]

branchCondition: %cmp = icmp sgt i32 1, 2

(= ValVar7 1)

This conditional ICFGEdge is **infeasible**!!