Static code analysis tool for C#

Vladimir Koshelev
Senior Researcher
ISP RAS
SharpChecker is production ready static code analysis tool for C#

• SharpChecker is based on the .NET Compiler Platform (“Roslyn”)

• SharpChecker performs complex interprocedural program analysis.

• It has good performance, for example, SharpChecker analyzes Roslyn 1.3.2 (prox. 2 millions lines of C# code) in 35 minutes, using intel i7 CPU.
SharpChecker architecture

SharpChecker

SMT Solver
Z3

Analyses
Symbolic Execution
Data-flow
AST
Call Graph
Inter-procedural summaries

AST + Semantic
Roslyn

Warning Postprocessing
Svace

SharpChecker ISP RAS
AST Analyzers

• 38 different AST Analyzers are implemented in SharpChecker

• Examples:
  • Heuristic infinite loop detection (INFINITE_LOOP)
  • Using obsolete crypto algorithm (OBSOLETE_CRYPTO)
  • Copy paste errors (BAD_COPY_PASTE)
Bad Copy Paste Example (openBVE project)

```csharp
for (int j = 0; j < Prototype.Mesh.Materials.Length; j++) {

    if (DaytimeTexture != null) {
    } else {
    }
}

// In the expression DaytimeTexture != null variable DaytimeTexture possibly need to be replaced with NighttimeTexture after copy paste

if (DaytimeTexture != null) {
} else {
}
```
Bad Copy Paste Example (openBVE project)

```csharp
for (int j = 0; j < Prototype.Mesh.Materials.Length; j++) {

    if (DaytimeTexture != null) {
    } else {
    }

    if (NighttimeTexture != null) {
    } else {
    }
}
```
Data-flow analyzers

• Unused value analyzer:
  • Detects unused variable definitions

• Necessary condition analyzer (constant expression detection):
  • Calculates necessary condition for an entry point of each basic block
  • To express conditions we use predicate abstraction
  • To check that the expression is always constant we employ SMT provers.
public static string FormatTimeSecondsMinutes(int seconds)
{
    string time;
    if (seconds < 60)
    {
        time = seconds + " seconds";
    }
    else
    {
        var mins = seconds / 60;
        time = mins + (mins == 1 ? " minute" : " minutes");
        if (seconds % 60 != 0)
        {
            // Expression seconds == 1 is always false.
            time += " and " + seconds + (seconds == 1 ? " second" : " seconds");
        }
    }
    return time;
}
Symbolic Execution

• We consider each method’s entry point as an entry point for symbolic execution.

• We analyze all paths that have at most one back edge.

• To reduce the complexity of path exploration, we merge symbolic states in basic blocks that have more than one predecessor.
Memory model

• We parameterize program state at entry point by set of symbolic variables.

• We use optimistic assumption that all references at entry point are not aliases.

• $S$ is a set of symbolic variables. We describe memory as two maps:
  \[
  \langle \text{Local} \mapsto S, S \times \text{Field} \mapsto S \rangle
  \]

The first map symbolically models read/write operations with local variables.

The second map symbolically models read/write operations with the heap.
Conditions

• To achieve path sensitivity we model origin arithmetic operations by symbolic expressions

• For each basic block we calculate precise path condition

• For specified path and basic block, a path condition defines a set of concrete input states which leads the execution to the basic block following the path
Symbolic execution example

```c
int Foo(Baz baz, int a) baz→ baz, a→ a, (baz,X)→ x,(baz,Y)→ y {
    baz.X = a;
    if (baz.X + baz.Y > 0)
        return baz.X;
    else
        return baz.Y;
}
Bold expressions are symbolic variables
```
Symbolic execution example

```c
int Foo(Baz baz, int a)
{
    baz.X = a;
    if (baz.X + baz.Y > 0)
        return baz.X;
    else
        return baz.Y;
}
```

Bold expressions are symbolic variables
Symbolic execution example

```c
int Foo(Baz baz, int a)
{
    baz.X = a;
    if (baz.X + baz.Y > 0)
        return baz.X;
    else
        return baz.Y;
}
```

---

**Bold expressions are symbolic variables**

Path Condition: $a + y > 0$ ($\langle baz, X \rangle \rightarrow a, \langle baz, Y \rangle \rightarrow y$)
Symbolic execution example

```
int Foo(Baz baz, int a)
{
    baz.X = a;
    if (baz.X + baz.Y > 0)
        return baz.X;
    else
        return baz.Y;
}
```

Bold expressions are symbolic variables

Path Condition: \( a + y > 0 \) \((\text{baz,} X \rightarrow a, \langle \text{baz,} Y \rangle \rightarrow y)\)

Path Condition: \( a + y \leq 0 \) \((\text{baz,} X \rightarrow a, \langle \text{baz,} Y \rangle \rightarrow y)\)
Null dereference examples

// NullReferenceException only when baz == null

public string Foo(Baz baz)
{
    // Should not report
    return baz.ToString();
}

// always NullReferenceException

public string Foo(Baz baz)
{
    baz = null;
    // Must report
    return baz.ToString();
}
Null dereference examples 2

// NullReferenceException only when // f = true, f2 = true
class Example
{
    public string Foo(Baz baz, bool f, bool f2)
    {
        if (f) {
            baz = null;
        }
        if (f2) // Should report?
        {
            return baz.ToString();
        }
        return null;
    }
}

// baz can be null, however no // path have null in baz and deref of // baz.
class Example
{
    public string Foo(Baz baz, bool f)
    {
        if (f) {
            if (baz == null) Bar();
        } else {// Should report?
            return baz.ToString();
        }
        return null;
    }
}
Error criteria for defect detection

• Common error criteria for null dereference:

• Let's assume that symbolic variable $x$ is dereferenced in the basic block $B$.

• $P_B$ is a path in a CFG that reaches basic block $B$

• $\text{Cond}_{P_B}(\bar{s})$ – path condition

• Common error criteria for null dereference:

\[
\exists P_B \left( \exists \bar{s} \text{ Cond}_{P_B}(\bar{s}) \right) \land \forall \bar{s} \left( P_B(\bar{s}) \rightarrow x = \text{null} \right)
\]
Symbolic execution checkers

• Null dereference
  • Explicit null dereference
  • Dereference after comparison
  • Reverse dereference
  • Interprocedural dereference
  ...

• Resource Leak

• Invalid Cast

• Deadlock
Null dereference example (WCell project)

```csharp
List<Exception> innerExceptions = null;

Step 1: Entering the foreach:
foreach (var inner in aggregateException.InnerExceptions)
{
    AggregateException innerAsAggregate = inner as AggregateException;
    Step 2: Condition innerAsAggregate != null taking false branch
    if (innerAsAggregate != null)
    {
        AggregateException newChildAggregate = HandleRecursively(innerAsAggregate, predicate);
        if (newChildAggregate != null)
        {
            if (innerExceptions != null) innerExceptions = new List<Exception>();
            innerExceptions.Add(newChildAggregate);
        }
    }
    else if (!predicate(inner))
    {
        Step 3: Condition !predicate(inner) taking true branch
    }
    Step 4: Condition innerExceptions != null taking false branch
    if (innerExceptions != null) innerExceptions = new List<Exception>();
    Value innerExceptions, which has null value, is dereferenced in method call innerExceptions.Add();
    innerExceptions.Add(inner);
```
Null dereference example (WCell project)

```csharp
declared at
List<Exception> innerExceptions = null;

Step 1: Entering the foreach:
foreach (var inner in aggregateException.InnerException)
{
    AggregateException innerAsAggregate = inner as AggregateException;
    Step 2: Condition innerAsAggregate != null taking false branch
    if (innerAsAggregate != null)
    {
        AggregateException newChildAggregate = HandleRecursively(innerAsAggregate, predicate);
        if (newChildAggregate != null)
        {
            if (innerExceptions != null) innerExceptions = new List<Exception>();
            innerExceptions.Add(newChildAggregate);
        }
    }
    Step 3: Condition !predicate(inner) taking true branch
    else if (!predicate(inner))
    {
    }
    Step 4: Condition innerExceptions != null taking false branch
    if (innerExceptions != null) innerExceptions = new List<Exception>();
    Value innerExceptions, which has null value, is dereferenced in method call innerExceptions.Add()
    innerExceptions.Add(inner);
```

SharpChecker ISP RAS
Null dereference example (WCell project)

```csharp
List<Exception> innerExceptions = null;

Step 1: Entering the foreach:
foreach (var inner in aggregateException.InnerExceptions)
{
    AggregateException innerAsAggregate = inner as AggregateException;
    Step 2: Condition innerAsAggregate != null taking false branch
    if (innerAsAggregate != null)
    {
        AggregateException newChildAggregate = HandleRecursively(innerAsAggregate, predicate);
        if (newChildAggregate != null)
        {
            if (innerExceptions != null) innerExceptions = new List<Exception>();
            innerExceptions.Add(newChildAggregate);
        }
    }
    Step 3: Condition !predicate(inner) taking true branch
    else if (!predicate(inner))
    {
    }
    Step 4: Condition innerExceptions != null taking false branch
    if (innerExceptions != null) innerExceptions = new List<Exception>();
    Value innerExceptions, which has null value, is dereferenced in method call innerExceptions.Add()
    innerExceptions.Add(inner);
```
Null dereference example (WCell project)

```csharp
List<Exception> innerExceptions = null;

Step 1: Entering the foreach:
foreach (var inner in aggregateException.InnerExceptions)
{
    AggregateException innerAsAggregate = inner as AggregateException;

    Step 2: Condition innerAsAggregate != null taking false branch
    if (innerAsAggregate != null)
    {
        AggregateException newChildAggregate = HandleRecursively(innerAsAggregate, predicate);
        if (newChildAggregate != null)
        {
            if (innerExceptions != null) innerExceptions = new List<Exception>();
            innerExceptions.Add(newChildAggregate);
        }
    }

    Step 3: Condition !predicate(inner) taking true branch
    else if (!predicate(inner))
    {
        Step 4: Condition innerExceptions != null taking false branch
        if (innerExceptions != null) innerExceptions = new List<Exception>();
        Value innerExceptions, which has null value, is dereferenced in method call innerExceptions.Add()
        innerExceptions.Add(inner);
    }

Also dereferenced in method call innerExceptions.Add()
innerExceptions.Add(newChildAggregate);
```

SharpChecker ISP RAS
Null dereference example (WCell project)

```csharp
List<Exception> innerExceptions = null;

Step 1: Entering the foreach:
foreach (var inner in aggregateException.InnerExceptions)
{
    AggregateException innerAsAggregate = inner as AggregateException;
    Step 2: Condition innerAsAggregate != null taking false branch
    if (innerAsAggregate != null)
    {
        AggregateException newChildAggregate = HandleRecursively(innerAsAggregate, predicate);
        if (newChildAggregate != null)
        {
            if (innerExceptions != null) innerExceptions = new List<Exception>();
            innerExceptions.Add(newChildAggregate);
        }
    }
    Also dereferenced in method call innerExceptions.Add()
    innerExceptions.Add(inner);
    Step 3: Condition !predicate(inner) taking true branch
    else if (!predicate(inner))
    {
    }
    Step 4: Condition innerExceptions != null taking false branch
    if (innerExceptions != null) innerExceptions = new List<Exception>();
    Value innerExceptions, which has null value, is dereferenced in method call innerExceptions.Add()
    innerExceptions.Add(inner);
```
Null dereference example (WCell project)

```
List<Exception> innerExceptions = null;

Step 1: Entering the foreach:
foreach (var inner in aggregateException.InnerExceptions)
{
    AggregateException innerAsAggregate = inner as AggregateException;
    Step 2: Condition innerAsAggregate != null taking false branch
    if (innerAsAggregate != null)
    {
        AggregateException newChildAggregate = HandleRecursively(innerAsAggregate, predicate);
        if (newChildAggregate != null)
        {
            if (innerExceptions != null) innerExceptions = new List<Exception>();
            innerExceptions.Add(newChildAggregate);
        }
    }
    Step 3: Condition !predicate(inner) taking true branch
    else if (!predicate(inner))
    {
    }
    Step 4: Condition innerExceptions != null taking false branch
    if (innerExceptions != null) innerExceptions = new List<Exception>();
    Value innerExceptions, which has null value, is dereferenced in method call innerExceptions.Add();
    innerExceptions.Add(inner);
```

SharpChecker ISP RAS
Null dereference example (WCell project)

```
List<Exception> innerExceptions = null;

Step 1: Entering the foreach:
foreach (var inner in aggregateException.InnerExceptions)
{
    AggregateException innerAsAggregate = inner as AggregateException;

    Step 2: Condition innerAsAggregate != null taking false branch
    if (innerAsAggregate != null)
    {
        AggregateException newChildAggregate = HandleRecursively(innerAsAggregate, predicate);
        if (newChildAggregate != null)
        {
            if (innerExceptions != null) innerExceptions = new List<Exception>();
            innerExceptions.Add(newChildAggregate);
        }
    }

    Step 3: Condition !predicate(inner) taking true branch
    else if (!predicate(inner))
    {
        Step 4: Condition innerExceptions != null taking false branch
        if (innerExceptions != null) innerExceptions = new List<Exception>();
        Value innerExceptions, which has null value, is dereferenced in method call innerExceptions.Add();
        innerExceptions.Add(inner);
    }
```

```
SharpChecker ISP RAS 26
```
Questions?

http://sharpchecker.ispras.ru

Email: sharpchecker@ispras.ru