

Detection of Variable Misuse Using Static Analysis Combined with Machine Learning

G. Morgachev, V. Ignatyev
A. Belevantsev

ISP RAS

December 5, 2019

BadCopyPaste

```
if (g1 != IntPtr.Zero)
    m_scene.actor_name_map.TryGetValue(g1, out p1);

if (g2 != IntPtr.Zero)
    m_scene.actor_name_map.TryGetValue(g1, out p2);
```

SwappedArgument

```
ClassifyStandard(destination, source);
ClassifyStandard(source, destination);
```

- a lot of unclassifiable errors;
- static analyzers use heuristics to find such errors;
- only small subset of alg. errors can be detected;
- useful information is contained in names of classes, methods, variables, etc.

BadCopyPaste

```
if (g1 != IntPtr.Zero)
    m_scene.actor_name_map.TryGetValue(g1, out p1);

if (g2 != IntPtr.Zero)
    m_scene.actor_name_map.TryGetValue(g1, out p2);
```

- *slot* - read access to a variable;
- *candidate for a slot* - variable, field and property which can be used in slot from compiler point of view;
- $cand(5) = \{g1, g2\}$.

Algorithmic error detection

Goal

To generalize variable misuse detection methods.

VarMisUse problem

Detect if the used one candidate is wrong for a given slot.

Approach

Neural network application for a graph representation of a program. The representation includes information from the both natural language in code and multiple levels of static program analysis.

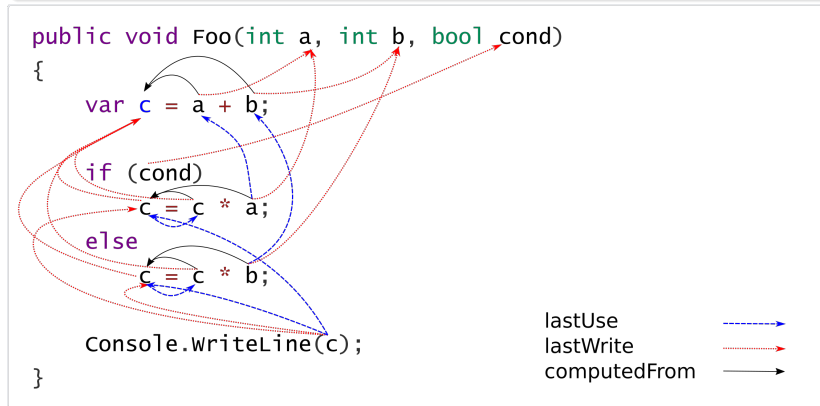
¹**Learning to Represent Programs with Graphs**

M. Allamanis and M. Brockschmidt and M. Khademi

- 1 Data preparation
 - slots selection;
 - finding of candidates for each slot;
 - computation of edges for each candidate.
- 2 Data processing
 - building of graphs for candidates;
 - features encoding.
- 3 Prediction of candidate's probabilities
- 4 Decision on the correctness of the used candidate

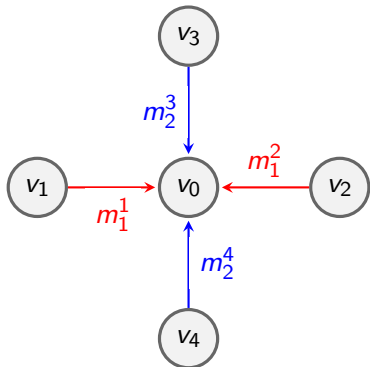
Program representation

The program graph is based on an abstract syntax tree with additional edges: LastWrite, LastUse, LastLexicalUse, NextOperand, FormalArgs, ComputedFrom.



Neural network architecture

Message Passing Neural Network



- for each graph node associate a representation vector $h^{(v)}$;
- messages are passed along edges;
 $m_k^v = f_k(h^{(v)})$ - the message from node v of kind k ;
- f_k - linear learnable function;
- the state $h^{(v)} = RNN(m_{in}, h^{(v)})$ is updated a fixed number of times.

Tokens Node

- Token
- SyntaxKind
- Type (*if exists*)

AST Node

- Unknown token
- SyntaxKind
- Type (*if exists*)

Embedding

- split tokens by *CamelCase* and *under_scope*
- train **word2vec** on a big corpus of source code;

Neural network architecture

- **GRU**;
- **LSTM**.

Usage of a **LSTM** cell shows slightly better results ($\sim 3\%$).

Graph representation

We remove **AST** nodes and edges from the representation.

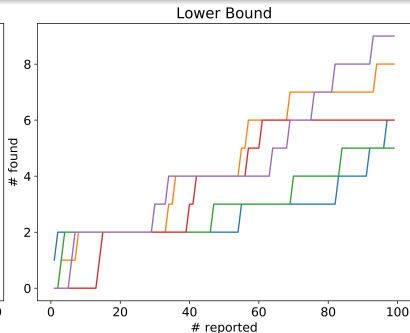
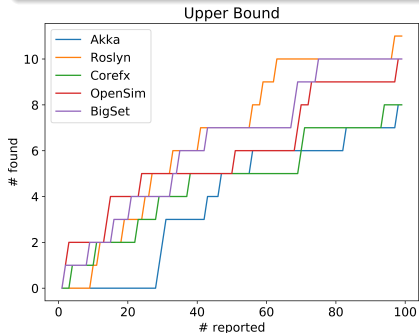
- 3 times smaller graph size;
- more than 3 times faster training;
- the same score.

Detected errors

The example of ambiguously classified warning

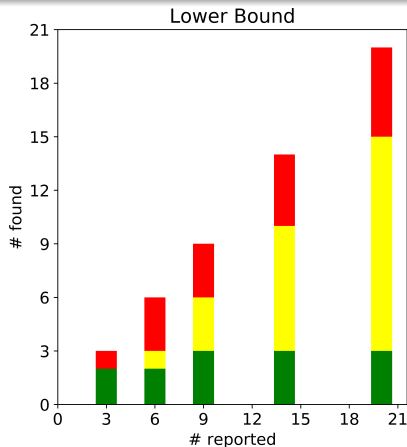
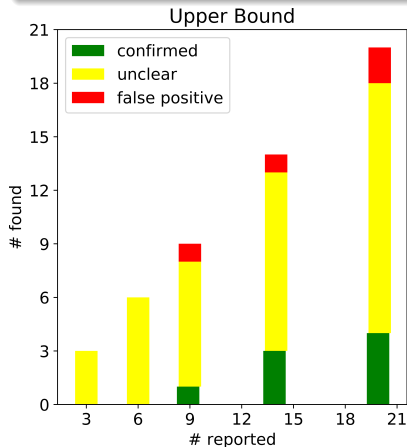
Error criteria

- 1 Exactly right candidate
- 2 Critical Zone
- 3 Lower Bound
- 4 Upper Bound



Real test

25 VarMisUse errors were added in OpenSim project.



```
public GenericMonitor(  
    Scene scene,  
    string name,  
    string friendlyName,  
    Func<GenericMonitor, double> getValueAction,  
    Func<GenericMonitor, string> getFriendlyValueAction) {  
    Scene = scene;  
    Name = name;  
    FriendlyName = name;  
    m_getFriendlyValueAction = getFriendlyValueAction;  
    m_getValueAction = getValueAction;  
}  
...  
m_staticMonitors.Add(new GenericMonitor(m_scene,  
    name: "TimeDilationMonitor", friendlyName: "Time Dilation",  
    m => m.Scene.StatsReporter.LastReportedSimStats[0],  
    m => m.GetValue().ToString()));
```

Detected errors

The example of ambiguously classified warning

```
public bool AckPacket(uint packet) {
    if (!complete) {
        if ((Data.Length - DataPointer) > 1000) {
            byte[] transferData = new byte[1000];
            Array.Copy(Data, DataPointer, transferData, 0, 1000);
            Client.SendXferPacket(XferID, Packet, transferData, isTaskInventory);
            Packet++;
            DataPointer += 1000;
        }
        else {
            byte[] transferData = new byte[Data.Length - DataPointer];
            Array.Copy(Data, DataPointer, transferData, 0, Data.Length - DataPointer);
            uint endPacket = Packet |= (uint) 0x80000000;
            Client.SendXferPacket(XferID, endPacket, transferData, isTaskInventory);
            Packet++;
            DataPointer += (Data.Length - DataPointer);
            complete = true;
        }
    }
    return complete;
}
```

Results

Accuracy for different projects

Project	Akka.Net	Cassandra	Lucene	Opensim	Netoffice	Openbve	Corefx	Roslyn
Accuracy	91.3%	96.1%	87.5%	83.6%	99.9%	83.6%	83.6%	86.2%

- better results for projects with small average method length;
- average numbers of candidates is 3.6;
- various training set allows to find more real errors.

Future work

- replacing syntax connection with the results of another compiler analyse;
- usage **PDG** to represent program;
- usage another neural network architecture.