Using Logic Programming to Recover C++ Classes and Methods from Compiled Executables

Edward J. Schwartz, Cory F. Cohen, Michael Duggan,
Jeffrey Gennari, Jeffrey S. Havrilla, Charles Hines

Software Engineering Institute

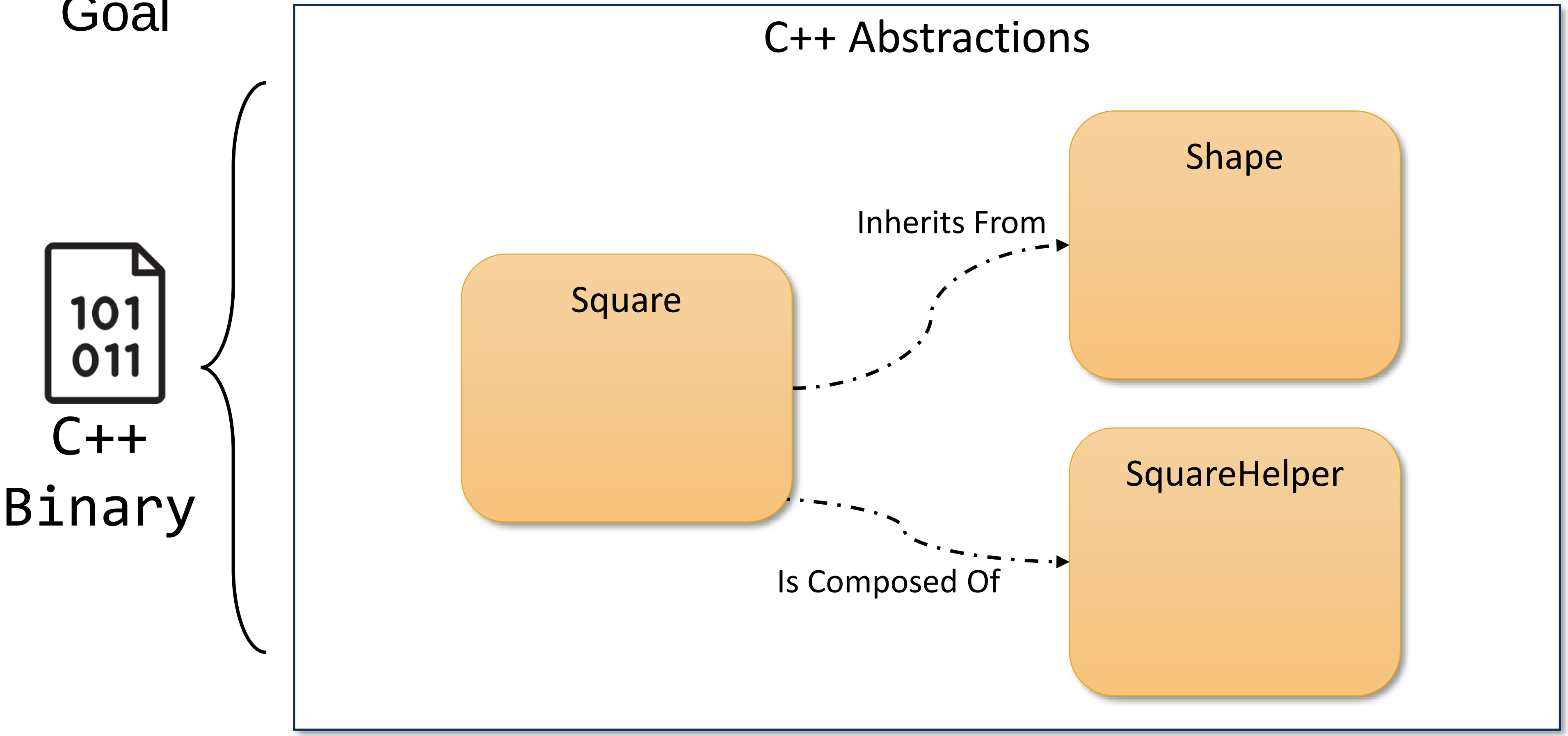
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

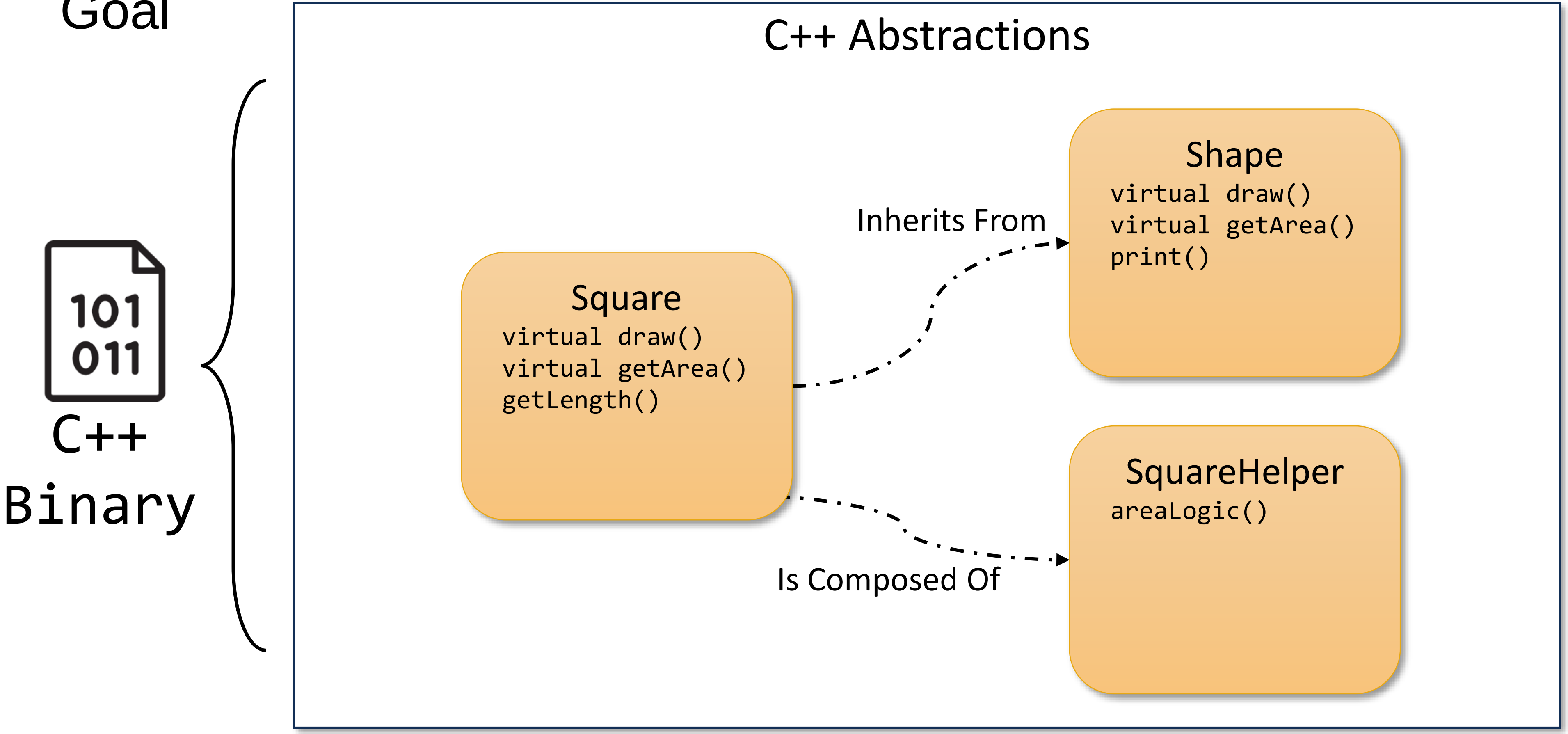
DM18-0828

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

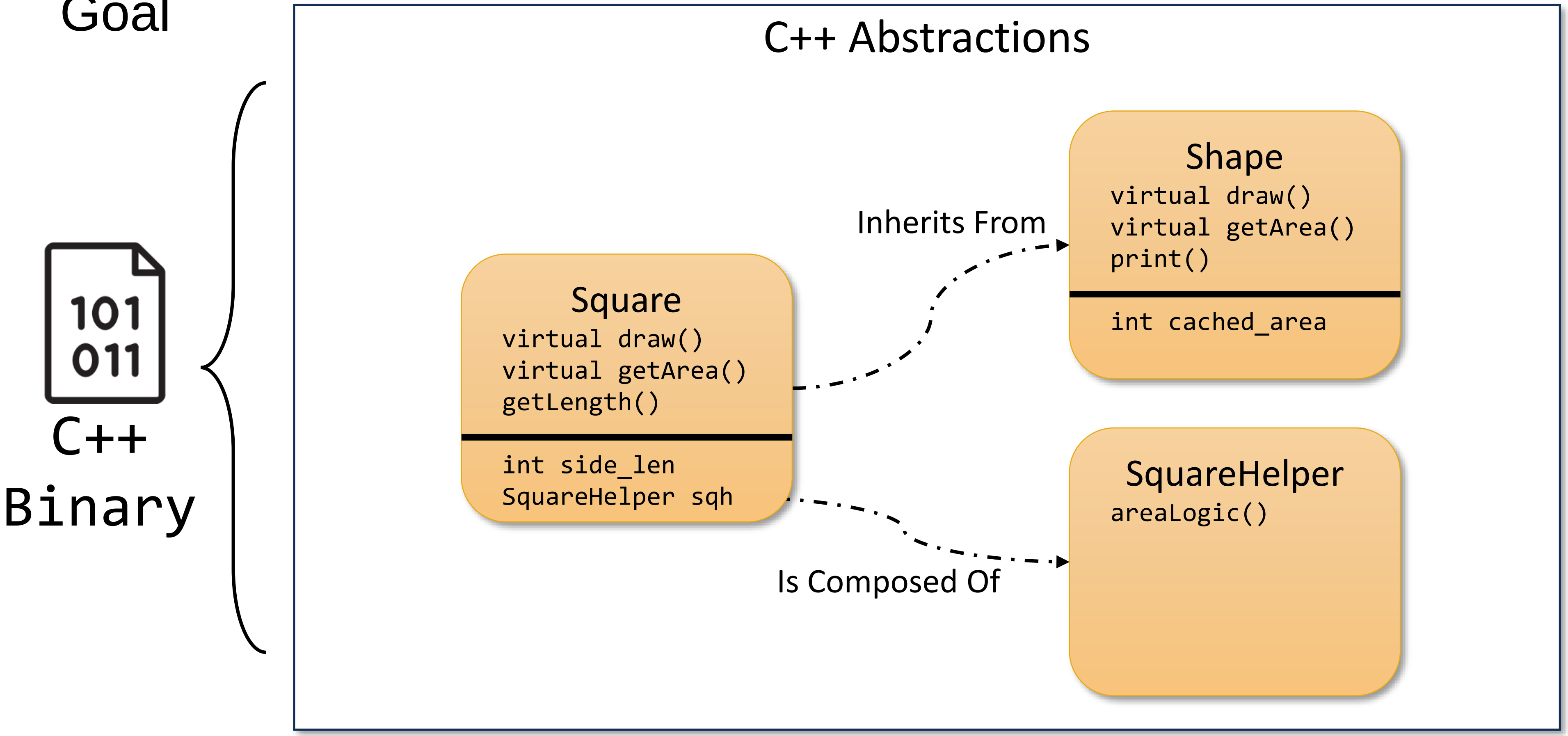
Goal


C++
Binary
C++ Abstractions
Shape
Inherits From
Square
SquareHelper
Is Composed Of

Goal


101
011
C++
Binary
C++ Abstractions
Square
virtual draw()
virtual getArea()
getLength()
Inherits From
Shape
virtual draw()
virtual getArea()
print()
Is Composed Of
SquareHelper
areaLogic()

Goal


101
011
C++
Binary
C++ Abstractions
Square
virtual draw()
virtual getArea()
getLength()
int side_len
SquareHelper sqh
Inherits From
Shape
virtual draw()
virtual getArea()
print()
int cached_area
Is Composed Of
SquareHelper
areaLogic()

Goals: How?

- Static
  - Analyze program without executing it
  - Can be applied to unknown software
  - No need for test cases

- Recover all classes and all methods

- Do not rely on RTTI metadata
  - Only available for polymorphic classes
  - Often removed or corrupted in malware

Existing Work

System	Static	All classes	All methods	Works w/o RTTI
SmartDec Fokin 2010	✓	✗	✓	✓
SecondWrite Yoo 2014	✓	✗	✗	✗
Various CFI systems Pawlowski 2017, etc.	✓	✗	✗	✓

These systems primarily recover information from virtual function tables

Virtual Function Tables

Shape's VFTable

```
0: Shape::draw
1: Shape::getArea
```

Square's VFTable

```
0: Square::draw
1: Square::getArea
```

VFTables enumerate the virtual functions of each class... but only for polymorphic classes.

Existing Work

System	Static	All classes	All methods	Works w/o RTTI
SmartDec Fokin 2010	✓	✗	✓	✓
SecondWrite Yoo 2014	✓	✗	✗	✗
Various CFI systems Pawlowski 2017, etc.	✓	✗	✗	✓
Lego Srinivasan 2014	✗	✓	✓	✓
ObjDigger Jin 2014	✓	✓	✓	✓

ObjDigger

ObjDigger uses <u>object pointer tracking</u> to recover non-polymorphic classes too

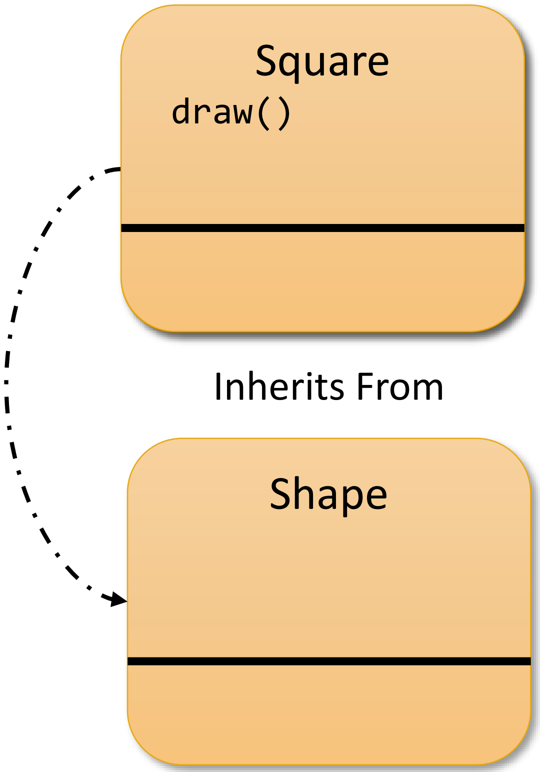
```
void foobar () {
        Obj *x = new Obj();
        x->foo ();
        func1(x);
}
```

foo is either defined on Obj class or inherited

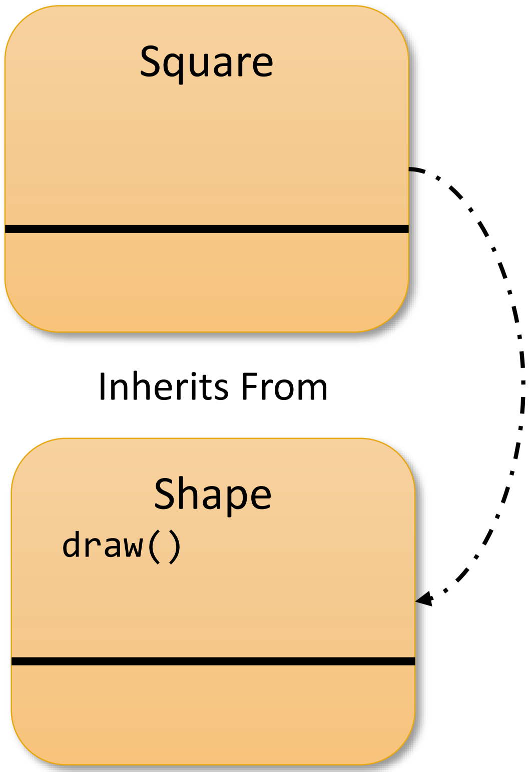
bar is either defined on Obj class or inherited

If the Obj class does not inherit, it defines foo and bar

Ambiguous Scenarios




```
int main() {
  Square sq;
  sq.draw ();
}
```



ObjDigger's Shortcomings

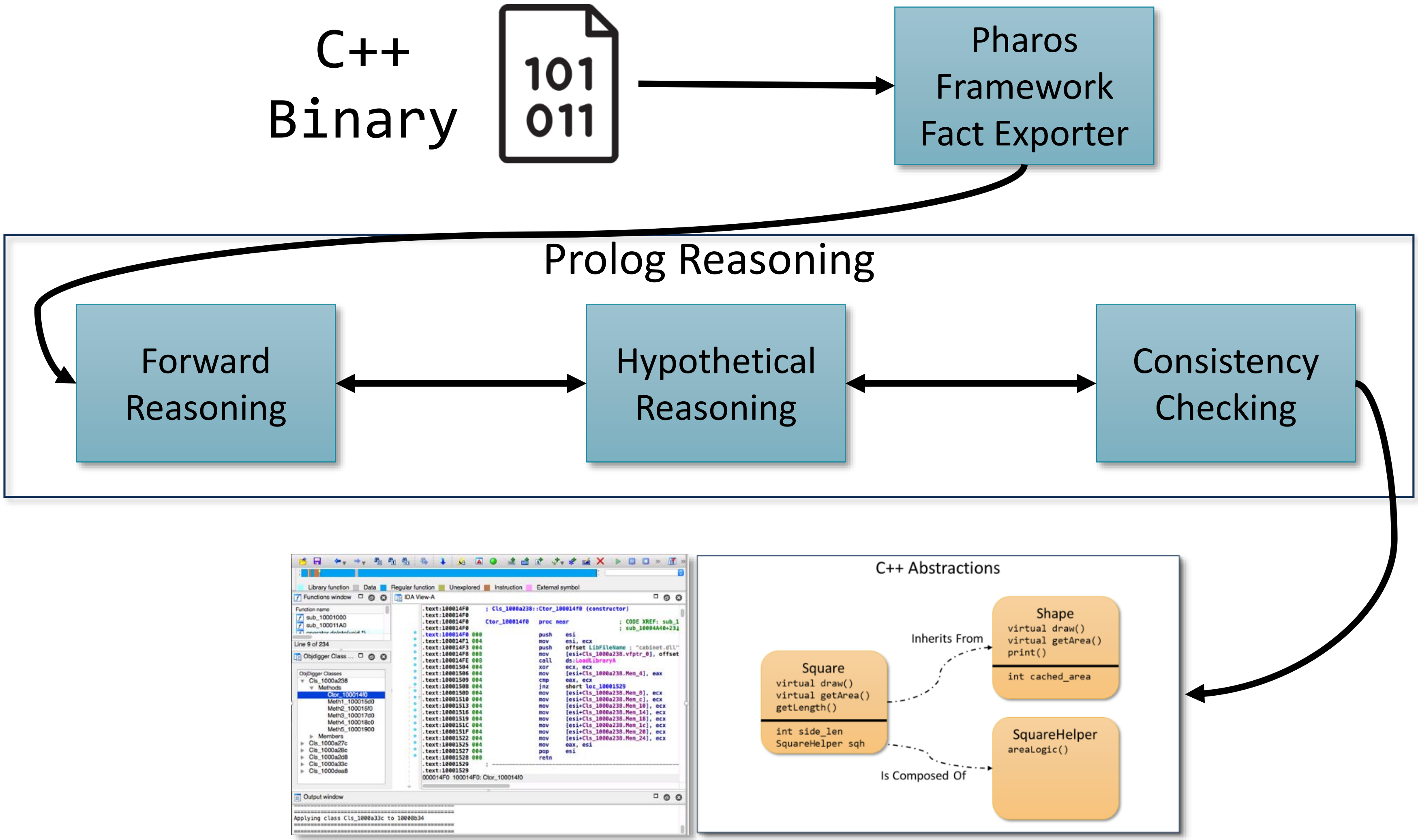
- ObjDigger makes mistakes on ambiguous scenarios
  - We tuned it so it was "usually" correct
  - But mistakes <u>propagated</u> and caused more serious problems
- ObjDigger was written <u>procedurally</u> in C++
  - `if (ambiguous_scenario) { classes[foo].add_method(bar); } // guess!`
  - There was no mechanism to detect mistakes and repair them
- Improving accuracy became more and more difficult

OOAnalyzer

OOAnalyzer was designed to overcome ObjDigger's shortcomings

- Reasoning implemented in Prolog
- Hypothetical reasoning
  - Detect ambiguous scenarios, and make best guess first
  - New: detect when guesses introduce inconsistency and undo them
  - Implemented using Prolog backtracking
- Reasoning is defined using declarative rules
  - Much easier to understand and maintain

OOAnalyzer System Design


C++
Binary
101
011
Pharos
Framework
Fact Exporter
Prolog Reasoning
Forward
Reasoning
Hypothetical
Reasoning
Consistency
Checking
C++ Abstractions
Shape
virtual draw()
virtual getArea()
print()
int cached_area
Inherits From
Square
virtual draw()
virtual getArea()
getLength()
int side_len
SquareHelper sqh
SquareHelper
areaLogic()
Is Composed Of

Fact Exporter

- Uses conventional binary analysis to produce <u>initial</u> facts about the program
  - Initial facts describe low-level program behaviors

- Simple symbolic analysis
  - Intentionally favors scalability over accuracy
  - Does not use constraint solvers
  - Symbolic memory aliases if memory addresses are equal after simplification
  - Path sensitive up to a threshold

- Sufficient because reasoning system can cope with mistakes

Initial Facts

Initial facts describe low-level program behaviors and form the basis upon which OOAnalyzer's reasoning system operates

Fact Name	Description
ObjPtrAllocation(I, F, P, S)	Instruction I in function F allocates S bytes of memory for the object pointed to by P.
ObjPtrInvoke(I, F, P, M)	Instruction I in function F calls method M on the object pointed to by P.
ObjPtrOffset(P_1, O, P_2)	Object pointer P_2 points to P_1 + O.
MemberAccess(I, M, O, S)	Instruction I in method M accesses S bytes of memory at offset O from the current object's pointer.
ThisCallMethod(M, P)	Method M receives the object pointed to by P in the ecx register.
NoCallsBefore(M)	No methods are called on any object pointer before method M.
ReturnsSelf(M)	Method M returns the object pointer that was passed as a parameter.
UninitializedReads(M)	Method M reads memory that was not written to by M.
PossibleVFTableEntry(VFT, O, M)	Method M may be at offset O in vftable VFT.

Entity Facts

Entity facts are produced during the reasoning process and describe the high-level model of the program being analyzed

Fact Name	Description
Method(M)	Method M is an OO method on a class or struct.
Constructor(M)	Method M is an object constructor.
Destructor(M)	Method M is an object destructor.
$Cl_a = Cl_b$	The sets of methods Cl_a and Cl_b both represent methods from the same class. These sets should be combined into a single class.
$Cl_a \leq Cl_b$	The sets of methods Cl_a and Cl_b either both represent methods from the same class, or, the methods in Cl_b are inherited from Cl_a.
$M \in Cl$	Method M is defined directly on class Cl.
ClassCallsMethod(Cl, M)	An instance of class Cl calls method M.

Other categories include virtual functions, class relationships, and sizes of classes and tables.

Reasoning

$$\frac{P_1 \qquad P_2 \qquad \ldots \qquad P_n}{C}$$

- Forward reasoning
  - Unambiguous scenarios
  - Interpretation: If P_1, P_2, … and P_n are satisfied, then C is true
  - If inconsistency is reached, P_1, P_2, … or P_n must not be true

- Hypothetical reasoning
  - Ambiguous scenarios
  - Interpretation: If P_1, P_2, … and P_n are satisfied, then <u>guess</u> C is true
  - If inconsistency is reached, then retract C and assume $\neg C$
  - If inconsistency is still reached, P_1, P_2, … or P_n must not be true

Forward Reasoning

If a method is directly called on a base class, it cannot be directly defined on the derived class.

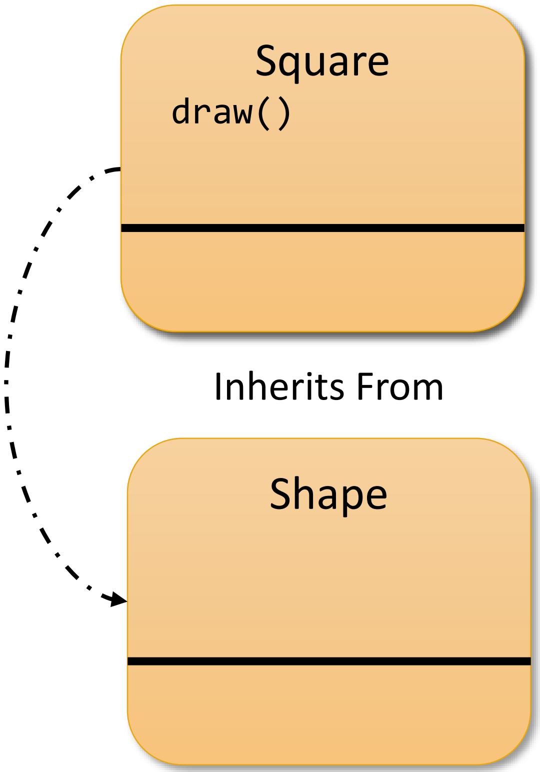
$$\frac{\begin{array}{c} \mathrm{Constructor}(M_d) \qquad M_d \in Cl_d \\ \mathrm{Constructor}(M_b) \qquad M_b \in Cl_b \\ \mathrm{ClassCallsMethod}(Cl_d, M) \\ \mathrm{ClassCallsMethod}(Cl_b, M) \qquad M_d \neq M_b \\ M \in Cl_m \qquad Cl_d \neq Cl_b \qquad \mathrm{DerivedClass}(Cl_d, Cl_b, \_) \end{array}}{Cl_m \neq Cl_d}$$

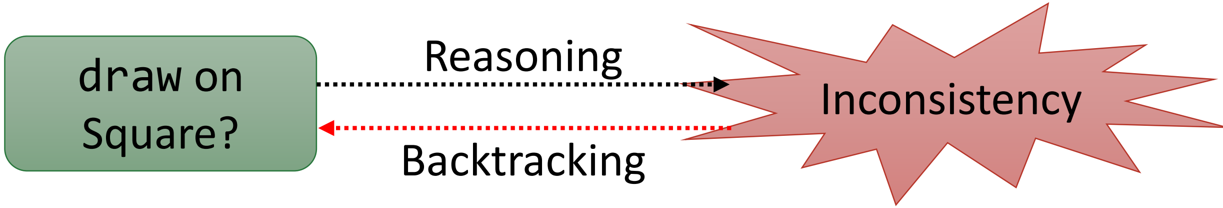
Hypothetical Reasoning

If a method is called on a derived class but not a base class, (first) assume the method is defined on the derived class.

$$\frac{\mathrm{ClassCallsMethod}(\mathrm{Cl}_d, M) \qquad \neg\mathrm{ClassCallsMethod}(\mathrm{Cl}_b, M) \qquad M \in \mathrm{Cl} \qquad \mathrm{DerivedClass}(\mathrm{Cl}_d, \mathrm{Cl}_b, \_)}{\mathrm{Cl}_d = \mathrm{Cl}}$$

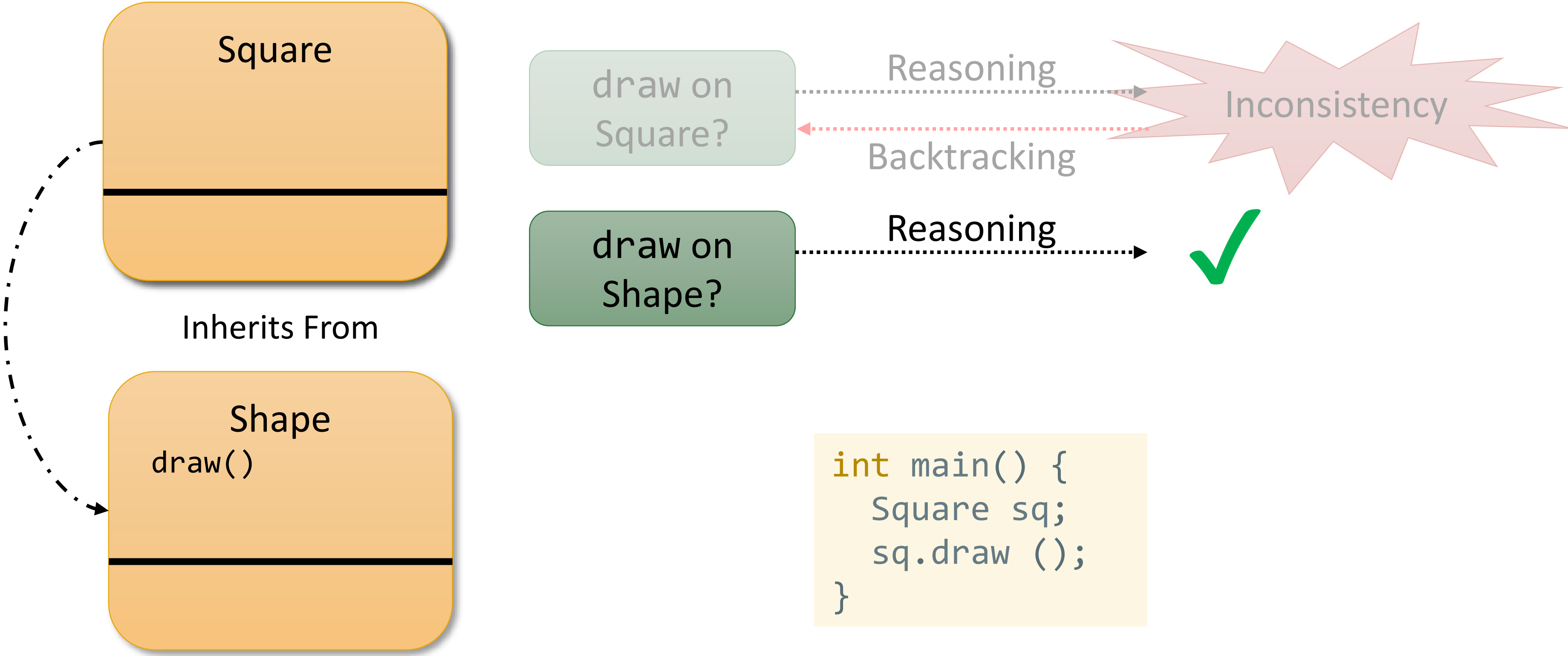
Hypothetical Reasoning






```
int main() {
  Square sq;
  sq.draw ();
}
```

Hypothetical Reasoning



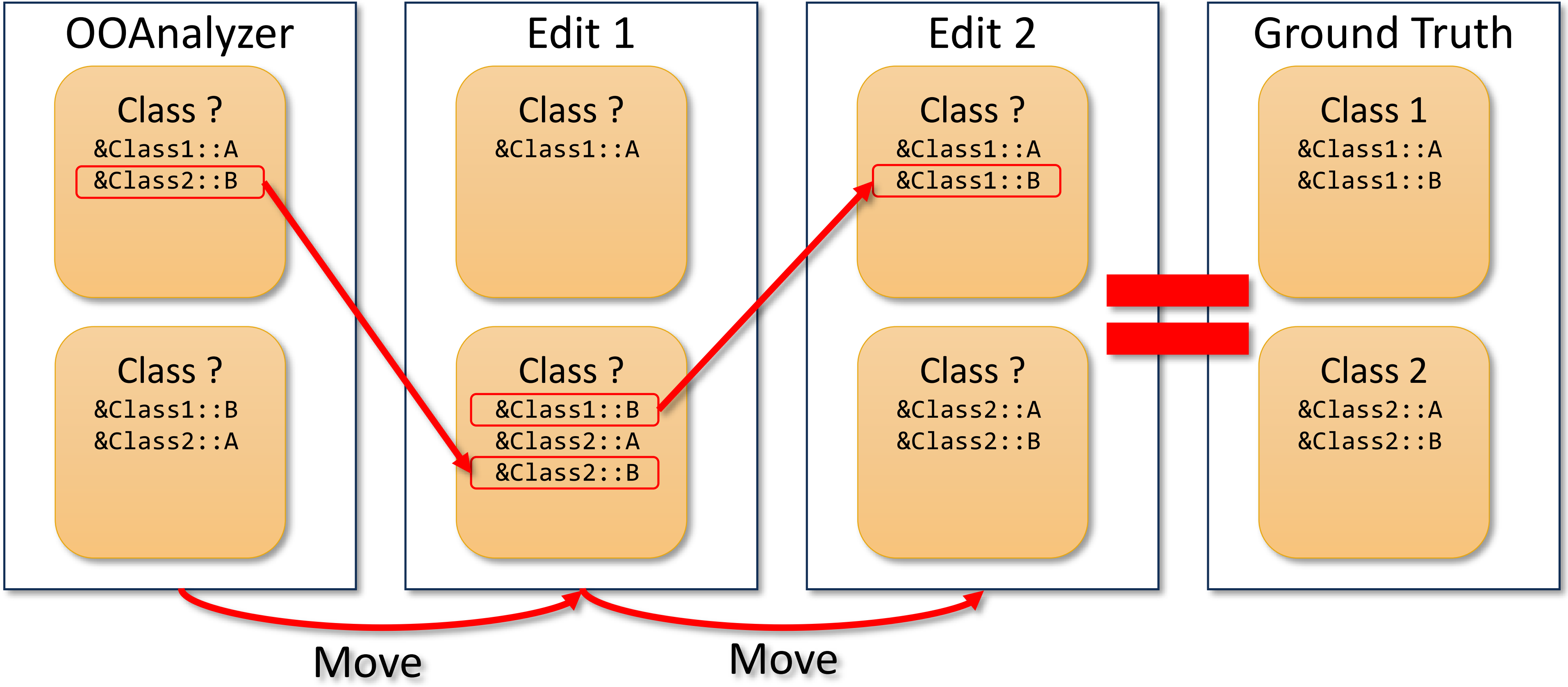

```
int main() {
  Square sq;
  sq.draw ();
}
```

Research Questions & Evaluation

Research Questions

- How well does OOAnalyzer identify C++ classes and their constituent methods?
  - OOAnalyzer classes may not naturally correspond to ground truth classes
  - New metric: <u>edit distance</u>

Edit Distance Example



ObjDigger vs OOAnalyzer Edit Distances on ObjDigger Programs

ObjDigger made 121 mistakes

ObjDigger recovered about 45% of methods

OOAnalyzer recovered about 90% of methods

Program	# Class	# Method
CImg	29	220
Light-pop3-smtp	44	295
Optionparser	11	56
PicoHttpD	95	656
X3c	6	28

ObjDigger vs OOAnalyzer Edit Distances on Cleanware

Program	# Class	# Method	ObjDigger Edits	ObjDigger Edits (%)	OOAnalyzer Edits	OOAnalyzer Edits (%)
Firefox.exe	141	638	507	79.5%	212	33.2%
Log4cpp Debug	139	893	829	92.8%	239	26.8%
Log4cpp Release	76	378	272	72.0%	75	19.8%
muParser Debug	180	1437	1361	94.7%	483	33.6%
muParser Release	94	598	369	61.7%	183	30.6%
MySQL cfg_editor.dll	190	1266	∞	∞	391	30.9%
MySQL mysql.exe	202	1395	∞	∞	439	31.5%
TinyXML Debug	35	415	268	64.6%	69	16.6%
TinyXML Release	33	283	174	61.5%	55	19.4%

ObjDigger vs OOAnalyzer Edit Distances on Cleanware

Program	# Class	# Method	ObjDigger Edits	ObjDigger Edits (%)	OOAnalyzer Edits	OOAnalyzer Edits (%)
Firefox.exe	141	638	507	79.5%	**212**	**33.2%**
Log4cpp Debug	139	893	829	92.8%	239	26.8%
Log4cpp Release	76	378	272	72.0%	75	19.8%
muParser Debug	180	1437	1361	94.7%	483	33.6%
muParser Release	94	598	369	61.7%	183	30.6%
MySQL cfg_editor.dll	190	1266	∞	∞	391	30.9%
MySQL mysql.exe	202	1395	∞	∞	**439**	**31.5%**
TinyXML Debug	35	415	268	64.6%	69	16.6%
TinyXML Release	33	283	174	61.5%	55	19.4%

OOAnalyzer recovers between 67 and 84% of methods on medium to large programs

ObjDigger vs OOAnalyzer Edit Distances on Malware

Program	# Class	# Method	ObjDigger Edits	ObjDigger Edits (%)	OOAnalyzer Edits	OOAnalyzer Edits (%)
Malware 0faaa3d3	21	135	121	89.6%	21	15.6%
Malware 29be5a33	19	130	91	70.0%	15	11.5%
Malware 6098cb7c	55	339	131	38.6%	29	8.6%
Malware 628053dc	207	1920	1245	64.8%	378	19.7%
Malware 67b9be3c	400	2072	1299	62.7%	670	32.3%
Malware cfa69fff	39	184	125	67.9%	37	20.1%
Malware d597bee8	19	133	68	51.1%	17	12.8%
Malware deb6a7a1	283	2712	1900	70.1%	639	23.6%
Malware f101c05e	169	1601	987	61.6%	329	20.5%

Research Questions

- How well does OOAnalyzer identify C++ classes and their constituent methods?
  - OOAnalyzer classes may not naturally correspond to ground truth classes
  - New metric: edit distance

- **How well does OOAnalyzer classify methods as constructors, destructors, and virtual methods?**

OOAnalyzer Method Classification on ObjDigger Programs

Program	Constructor		
	Recall	Prec.	F
CImg	44/51	44/53	0.85
Light-pop3-smtp	41/52	41/44	0.85
Optionparser	10/11	10/10	0.95
PicoHttpD	117/142	117/126	0.87
X3c	6/7	6/6	0.92

6 of the 7 constructors were identified

Of the 6 constructors identified, 6 were correct

$$2 * \frac{recall * precision}{recall + precision}$$

OOAnalyzer Method Classification on Cleanware

Program	Constructor			Destructor			VFTables			Virtual Methods		
	Recall	Prec.	F	Recall	Prec.	F	Recall	Prec.	F	Recall	Prec.	F
Firefox.exe	40/51	40/54	0.76	1/39	1/1	0.05	18/33	18/18	0.71	85/101	85/98	0.85
Log4cpp Debug	192/209	192/197	0.95	40/118	40/40	0.51	18/18	18/18	1.00	84/101	84/86	0.92
Log4cpp Release	135/165	135/170	0.81	24/73	24/36	0.44	18/21	18/18	0.92	84/101	84/86	0.90
muParser Debug	293/325	293/314	0.92	28/156	28/30	0.30	12/12	12/13	0.96	35/47	35/43	0.78
muParser Release	197/252	197/269	0.76	15/91	15/21	0.27	12/14	12/13	0.89	35/47	35/37	0.83
MySQL cfg_editor.dll	260/290	260/311	0.87	107/281	107/111	0.55	69/69	69/69	1.00	321/427	321/325	0.85
MySQL mysql.exe	282/314	282/341	0.86	115/300	115/121	0.55	75/75	75/75	1.00	341/453	341/345	0.85
TinyXML Debug	53/60	53/57	0.91	0/39	0/3	0.00	24/24	24/24	1.00	101/119	101/102	0.91
TinyXML Release	49/60	49/53	0.87	27/39	27/36	0.72	24/24	24/24	1.00	101/119	101/103	0.91

Conclusion: OOAnalyzer

- Can scalably and statically recover abstractions about all classes and all methods from C++ executables
- Combines a light-weight symbolic analysis with a Prolog-based reasoning system that enables hypothetical reasoning about ambiguous scenarios
- Recovers 70% of classes/methods or more on most cleanware and malware
- Classifies constructors, virtual methods, and virtual function tables with high accuracy

Open source tool in the Pharos binary analysis framework:
https://github.com/cmu-sei/pharos

Questions?

Presenter

Edward J. Schwartz

Research Scientist

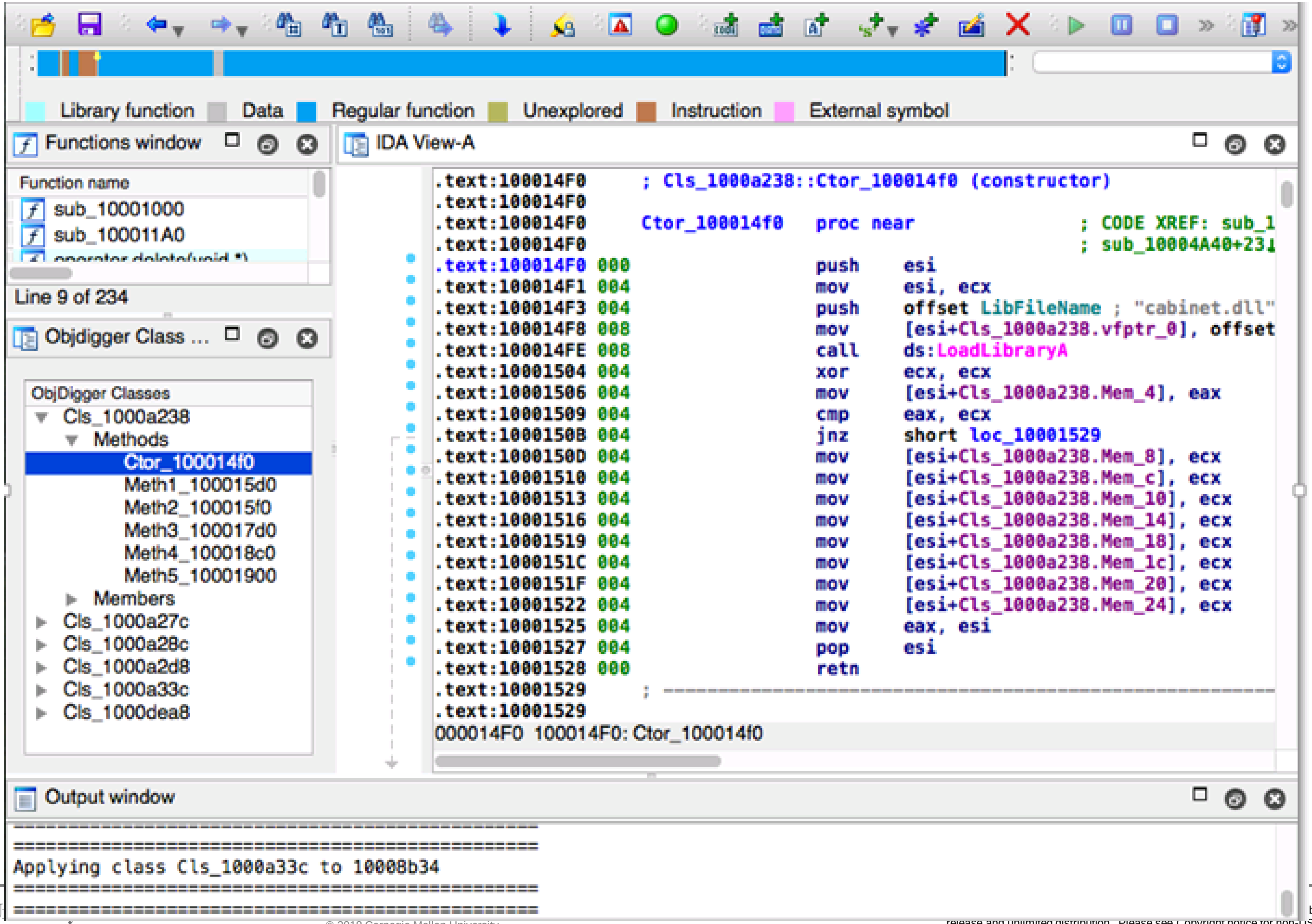
Carnegie Mellon University

SEI CERT/CC

Contact

Email: eschwartz@cert.org

Open source tool in the Pharos binary analysis framework:
https://github.com/cmu-sei/pharos


Library function
Data
Regular function
Unexplored
Instruction
External symbol
Functions window
IDA View-A
Function name
sub_10001000
sub_100011A0
Line 9 of 234
Objdigger Class ...
ObjDigger Classes
Cls_1000a238
Methods
Ctor_100014f0
Meth1_100015d0
Meth2_100015f0
Meth3_100017d0
Meth4_100018c0
Meth5_10001900
Members
Cls_1000a27c
Cls_1000a28c
Cls_1000a2d8
Cls_1000a33c
Cls_1000dea8
.text:100014F0 ; Cls_1000a238::Ctor_100014f0 (constructor)
.text:100014F0
.text:100014F0 Ctor_100014f0 proc near ; CODE XREF: sub_1
.text:100014F0 ; sub_10004A40+23↓
.text:100014F0 000 push esi
.text:100014F1 004 mov esi, ecx
.text:100014F3 004 push offset LibFileName ; "cabinet.dll"
.text:100014F8 008 mov [esi+Cls_1000a238.vfptr_0], offset
.text:100014FE 008 call ds:LoadLibraryA
.text:10001504 004 xor ecx, ecx
.text:10001506 004 mov [esi+Cls_1000a238.Mem_4], eax
.text:10001509 004 cmp eax, ecx
.text:1000150B 004 jnz short loc_10001529
.text:1000150D 004 mov [esi+Cls_1000a238.Mem_8], ecx
.text:10001510 004 mov [esi+Cls_1000a238.Mem_c], ecx
.text:10001513 004 mov [esi+Cls_1000a238.Mem_10], ecx
.text:10001516 004 mov [esi+Cls_1000a238.Mem_14], ecx
.text:10001519 004 mov [esi+Cls_1000a238.Mem_18], ecx
.text:1000151C 004 mov [esi+Cls_1000a238.Mem_1c], ecx
.text:1000151F 004 mov [esi+Cls_1000a238.Mem_20], ecx
.text:10001522 004 mov [esi+Cls_1000a238.Mem_24], ecx
.text:10001525 004 mov eax, esi
.text:10001527 004 pop esi
.text:10001528 000 retn
.text:10001529 ; ---
.text:10001529
000014F0 100014F0: Ctor_100014f0
Output window
Applying class Cls_1000a33c to 10008b34

```
finalClass(ClassID, VFTable, MinSize, MaxSize, RealDestructor, MethodList).
finalClass(0x412190, 0, 0x5, 0x5, 0, [0x412190, 0x4122f0, 0x4123c0,
0x4124c0]).
finalClass(0x412400, 0, 0x4, 0x4, 0, [0x412400]).
finalClass(0x417824, 0x417824, 0x54, 0x54, 0, [0x411860, 0x411970, 0x4119f0]).
finalClass(0x417860, 0x417860, 0xc, 0xc, 0, [0x4117e0, 0x411a70, 0x411ac0]).
finalClass(0x41787c, 0x41787c, 0x4, 0x4, 0, [0x4118f0, 0x411b40, 0x411b90]).
finalClass(0x41789c, 0x41789c, 0x4, 0x4, 0, [0x412800, 0x412830]).
finalClass(0x41b2f8, 0, 0, 0, 0, [0x41b2f8, 0x41b2fc, 0x41b324, 0x41b330]).
finalClass(0x41b304, 0, 0, 0, 0, [0x41b304, 0x41b310, 0x41b348]).
finalClass(0x41b308, 0, 0, 0, 0, [0x41b308, 0x41b320, 0x41b33c, 0x41b340]).
finalClass(0x41b30c, 0, 0, 0, 0, [0x41b30c, 0x41b31c, 0x41b334, 0x41b338]).
finalClass(0x41b328, 0, 0, 0, 0, [0x41b328, 0x41b32c, 0x41b344]).
```

```
finalInheritance(DerivedClassID, BaseClassID, ObjectOffset, VFTable, Virtual).
finalInheritance(0x412190, 0x412400, 0, 0, false).
finalInheritance(0x417824, 0x417860, 0, 0x417824, false).
finalInheritance(0x417824, 0x41787c, 0xc, 0x417814, false).
```

```
finalMember(Class, Offset, Sizes, Certainty).
finalMember(0x412190, 0x4, [0x1], certain).
finalMember(0x412400, 0, [0x4], certain).
finalMember(0x417824, 0x50, [0x4], certain).
finalMember(0x417860, 0, [0x4], certain).
finalMember(0x41787c, 0, [0x4], certain).
finalMember(0x41789c, 0, [0x4], certain).
```

```
finalMethodProperty(Method, Type, Certainty).
finalMethodProperty(0x411860, constructor, certain).
finalMethodProperty(0x4118f0, virtual, certain).
finalMethodProperty(0x411970, virtual, certain).
finalMethodProperty(0x4119f0, virtual, certain).
finalMethodProperty(0x411a70, constructor, certain).
finalMethodProperty(0x411ac0, virtual, certain).
finalMethodProperty(0x411b40, constructor, certain).
finalMethodProperty(0x411b90, virtual, certain).
finalMethodProperty(0x412190, constructor, certain).
finalMethodProperty(0x412400, constructor, certain).
finalMethodProperty(0x412800, constructor, certain).
finalMethodProperty(0x412830, virtual, certain).
```