Don’t Waste My Efforts: Pruning Redundant Sanitizer Checks by Developer-Implemented Type Checks

Yizhuo Zhai, Zhiyun Qian, Chengyu Song, Manu Sridharan,

Trent Jaeger, Paul Yu, and Srikanth V. Krishnamurthy

Type Confusion

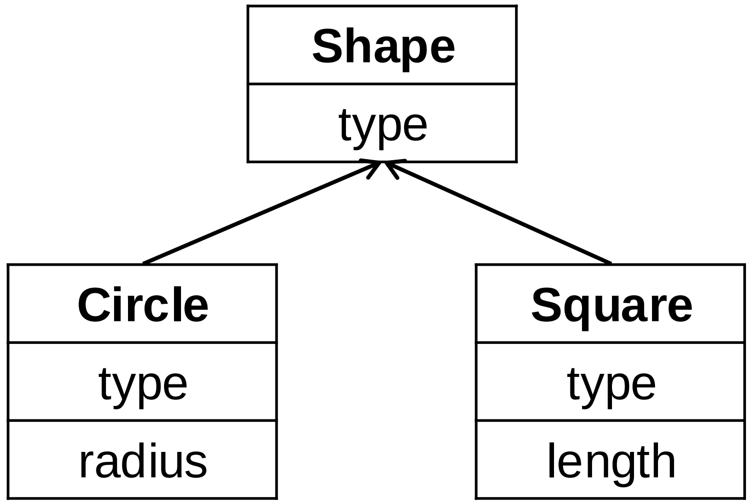
C++ Class Definitions

```
class Shape {RealShape type;};
class Circle : Shape {int radius;};
class Square : Shape {int length;};
```

Type Confusion

C++ Class Definitions

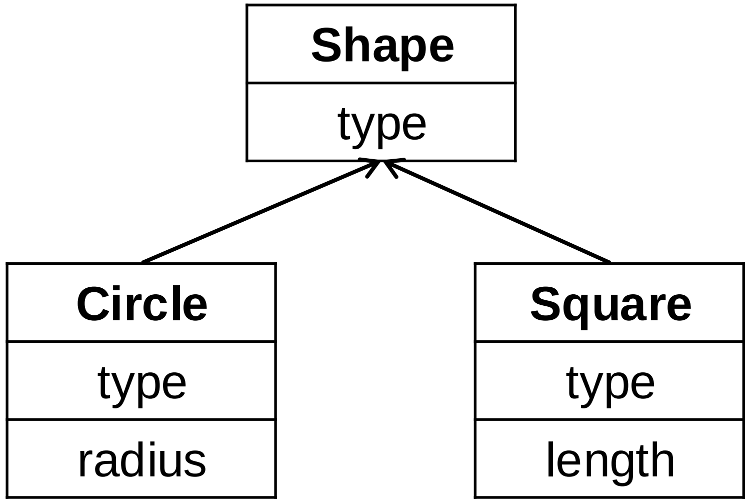
```
class Shape {RealShape type;};
class Circle : Shape {int radius;};
class Square : Shape {int length;};
```



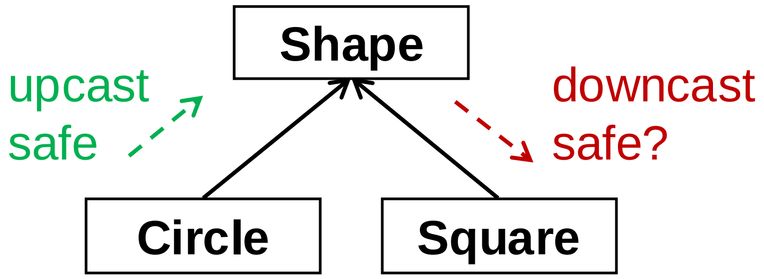
Type Confusion

C++ Class Definitions

```
class Shape {RealShape type;};
class Circle : Shape {int radius;};
class Square : Shape {int length;};
```



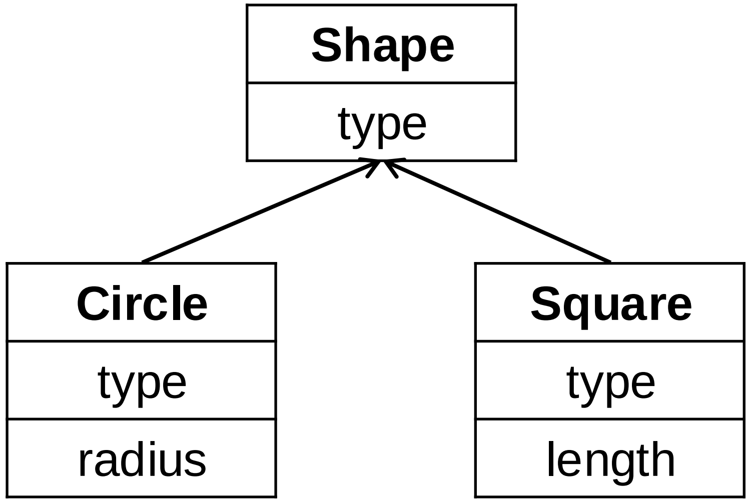
Type Casting



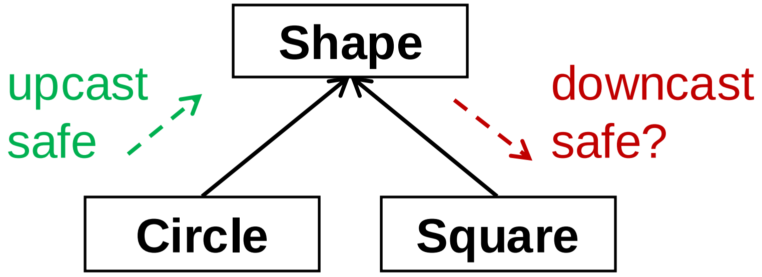
Type Confusion

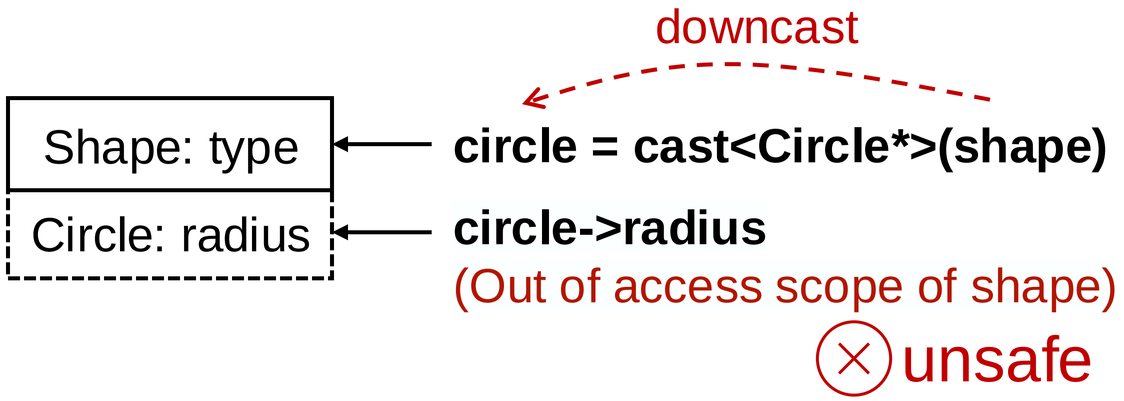
C++ Class Definitions

```
class Shape {RealShape type;};
class Circle : Shape {int radius;};
class Square : Shape {int length;};
```



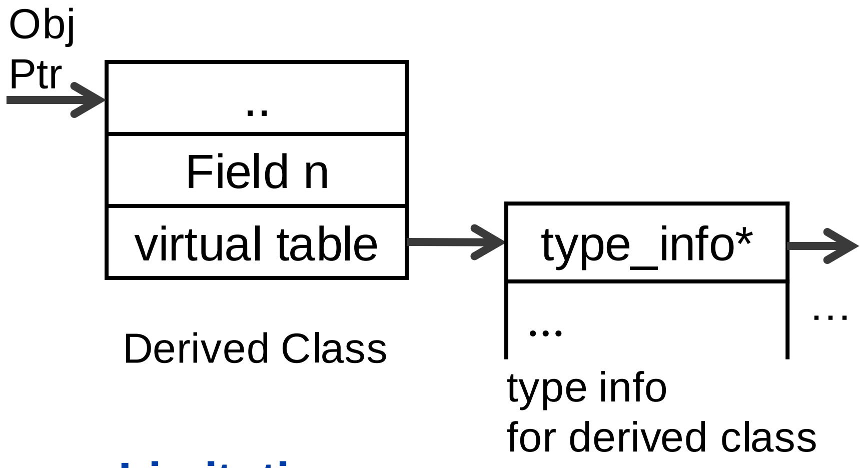
Type Casting





Dynamic Mitigations

1. dynamic_cast<>

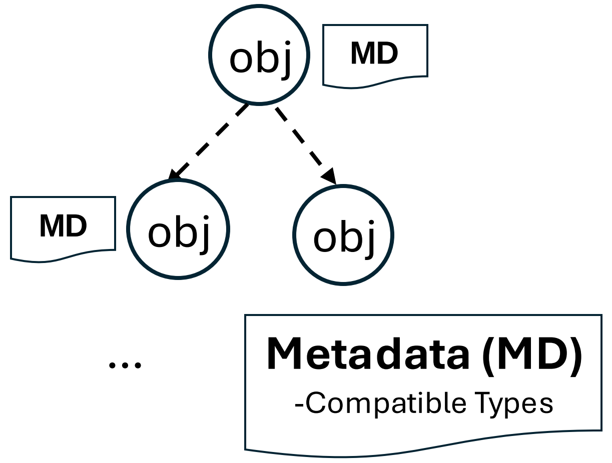


Limitations:

- Virtual Table Required
- High Overhead

☹ Prohibit for performance critical software

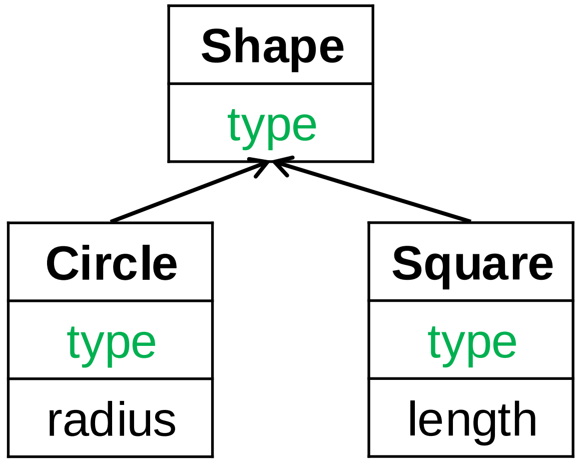
2. Sanitizer Approach



```
sanitizer_check()
cast<>
sanitizer_check()
cast<>
```

☺ Full Protection
☹ High Overhead

3. Custom Run Time Type Information (RTTI)

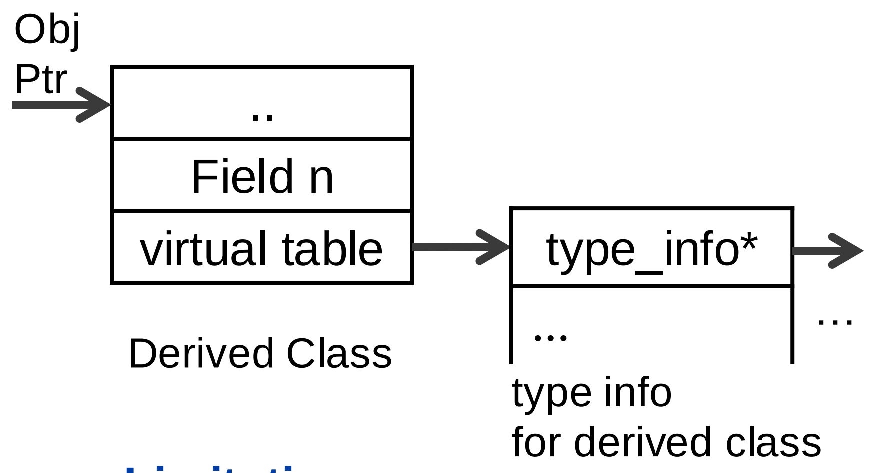


```
if (shape->type == CIRCLE) {
   circle = cast<circle*>(shape)
}
```

☹ No Full Protection
☺ Low Overhead

Dynamic Mitigations

1. dynamic_cast<>

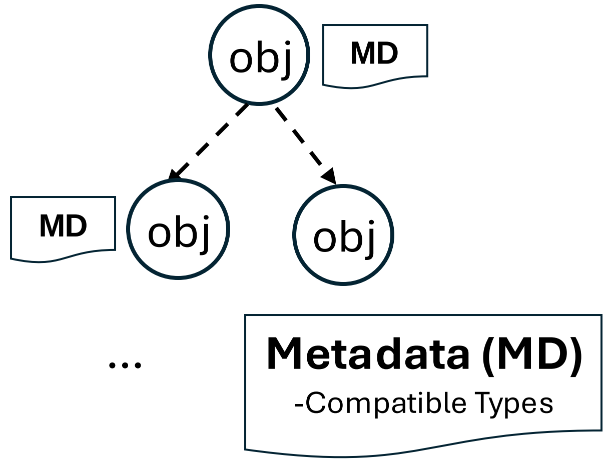


Limitations:

- Virtual Table Required
- High Overhead

☹ Prohibit for performance critical software

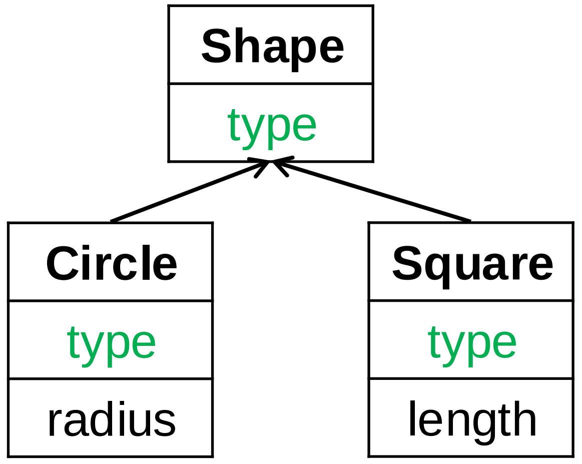
2. Sanitizer Approach



```
sanitizer_check()
cast<>
sanitizer_check()
cast<>
```

☺ Full Protection
☹ High Overhead

3. Custom Run Time Type Information (RTTI)

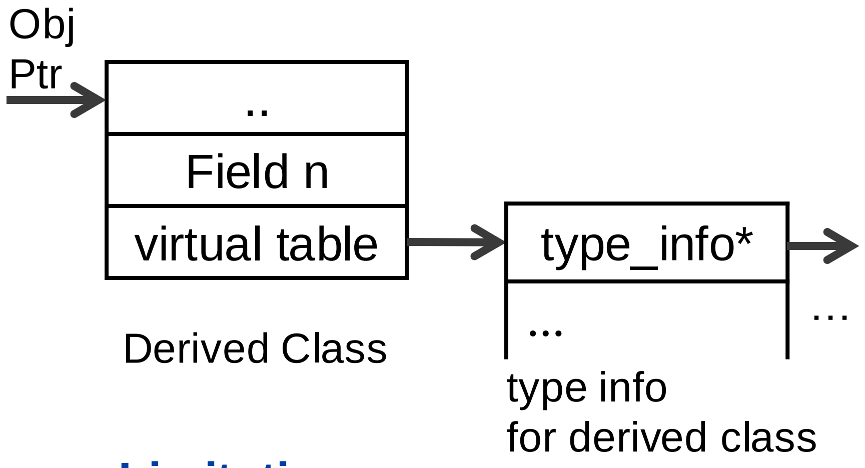


```
if (shape->type == CIRCLE) {
  circle = cast<circle*>(shape)
}
```

☹ No Full Protection
☺ Low Overhead

Dynamic Mitigations

1. dynamic_cast<>

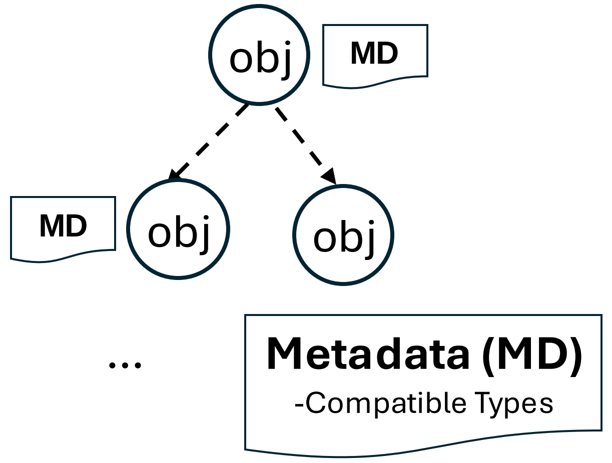


Limitations:

- Virtual Table Required
- High Overhead

☹ Prohibit for performance critical software

2. Sanitizer Approach

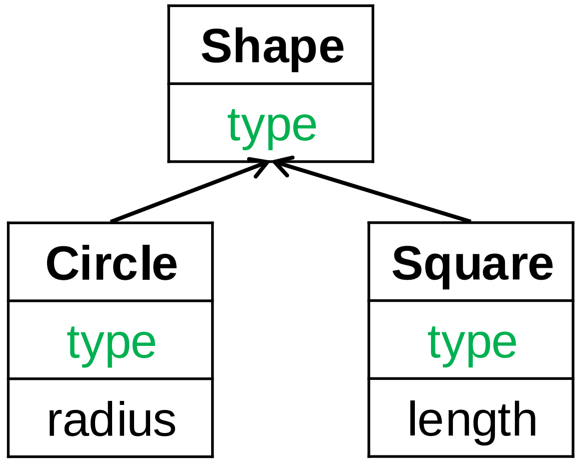


```
sanitizer_check()
cast<>
sanitizer_check()
cast<>
```

☺ Full Protection

~~☹ High Overhead~~

3. Custom Run Time Type Information (RTTI)

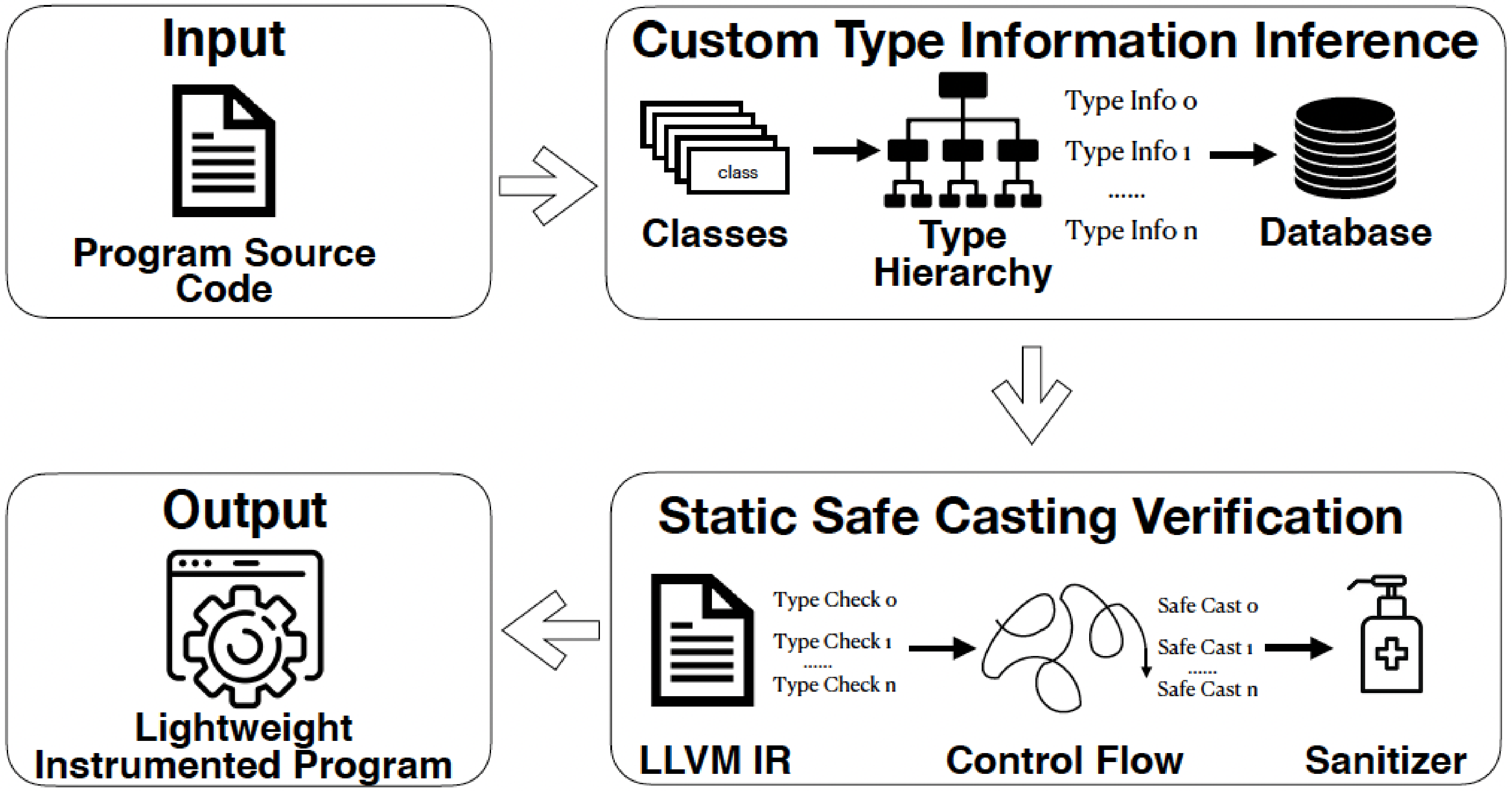


```
if (shape->type == CIRCLE) {
  circle = cast<circle*>(shape)
}
```

~~☹ No Full Protection~~

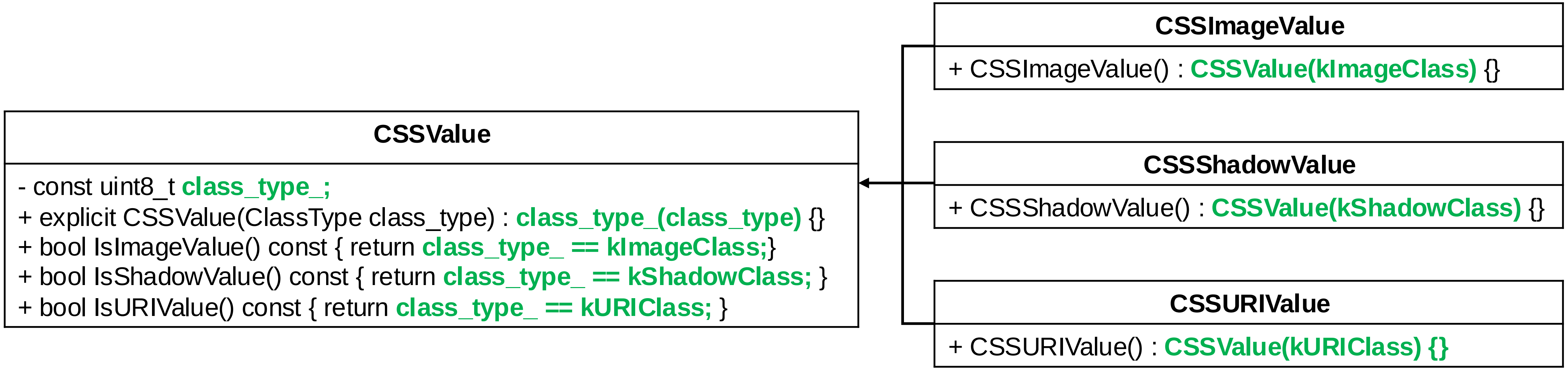
☺ Low Overhead

TPRunify Approach



Custom RTTI Inference

Category 1: Encoded in a Base Class Field



```
CSSValue
- const uint8_t class_type_;
+ explicit CSSValue(ClassType class_type) : class_type_(class_type) {}
+ bool IsImageValue() const { return class_type_ == kImageClass;}
+ bool IsShadowValue() const { return class_type_ == kShadowClass; }
+ bool IsURIValue() const { return class_type_ == kURIClass; }
```

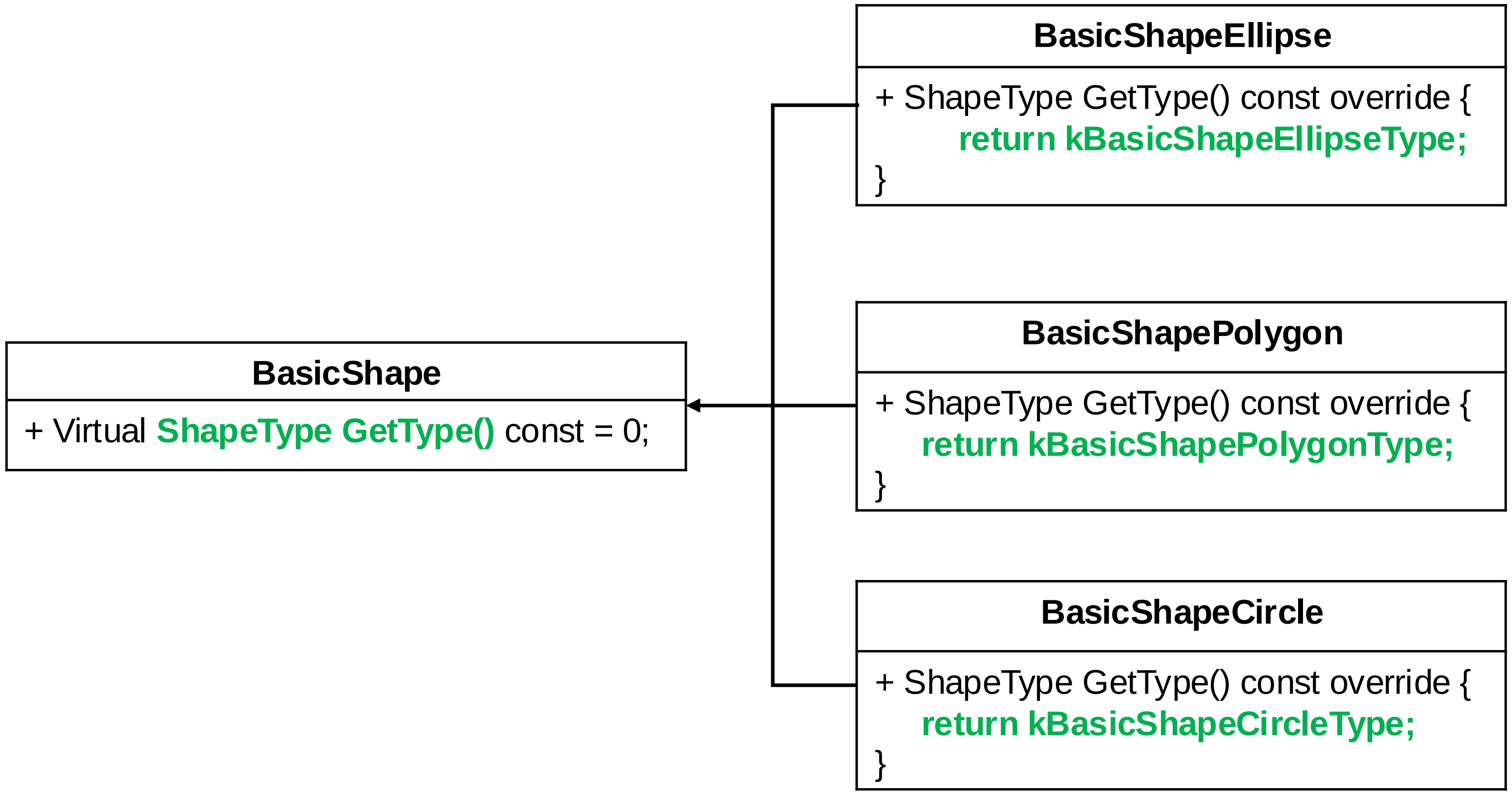
```
CSSImageValue
+ CSSImageValue() : CSSValue(kImageClass) {}
```

```
CSSShadowValue
+ CSSShadowValue() : CSSValue(kShadowClass) {}
```

```
CSSURIValue
+ CSSURIValue() : CSSValue(kURIClass) {}
```

Custom RTTI Inference

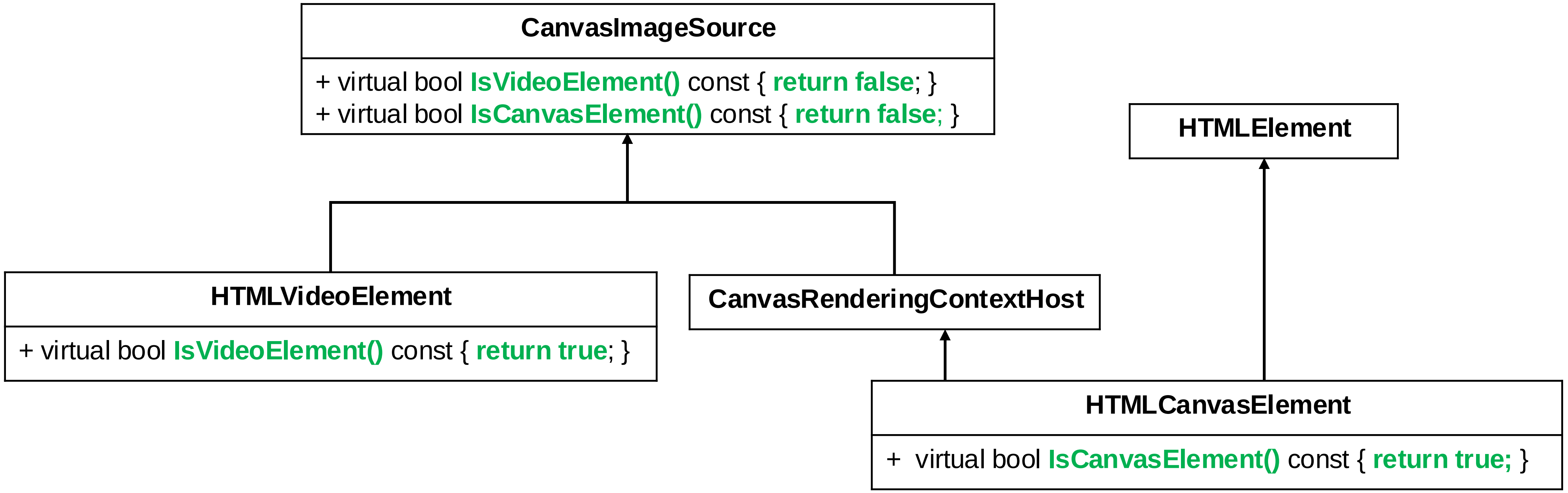
Category 2: As a Constant.



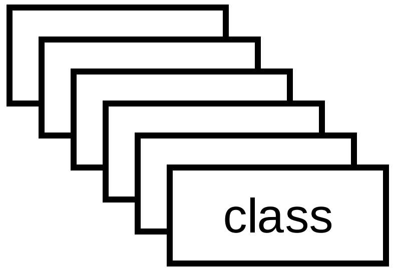
BasicShape
+ Virtual ShapeType GetType() const = 0;
BasicShapeEllipse
+ ShapeType GetType() const override {
return kBasicShapeEllipseType;
}
BasicShapePolygon
+ ShapeType GetType() const override {
return kBasicShapePolygonType;
}
BasicShapeCircle
+ ShapeType GetType() const override {
return kBasicShapeCircleType;
}

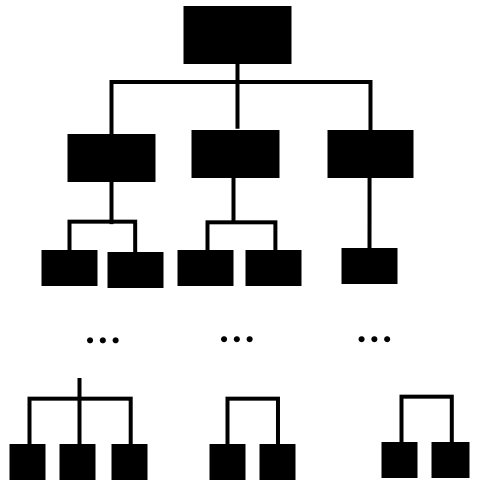
Custom RTTI Inference

Category 3: As a Type Check Function.


CanvasImageSource
+ virtual bool IsVideoElement() const { return false; }
+ virtual bool IsCanvasElement() const { return false; }
HTMLElement
HTMLVideoElement
+ virtual bool IsVideoElement() const { return true; }
CanvasRenderingContextHost
HTMLCanvasElement
+ virtual bool IsCanvasElement() const { return true; }

Custom RTTI Inference





Type Hierarchies

C1: Encoded in a Base Class Field

C2: As a Constant.

- Enumeration Constants.
- Assigned to the field in constructor
  or
  Returned by a virtual method
- Not be changed once initialized in constructor.

C3: As a Type Check Function.

- Override the return value of the virtual function in base class.
- The overwritten return value is unique across the hierarchy.

Static Safe Casting Verification

TPrunify Approach:

```
if (shape->type == CIRCLE) {   Circle

   circle = cast<circle>(shape); ✓ safe

}
```

Sanitizer Approach:

```
if (shape->type == CIRCLE) {
    sanitizer_check(); <- redundant
   circle = cast<Circle*>(shape);
}
```

Implementation

- Libclang : Custom RTTI Identification
- LLVM v14.0.5
- 9,652 lines of code in total
- Update HexType to LLVM v14.0.5 for comparison

Evaluation

- RQ1: Prevalence of custom RTTI
- RQ2: Safe casts identified by TPRunify
- RQ3: Runtime overhead reduction

Prevalence of custom RTTI

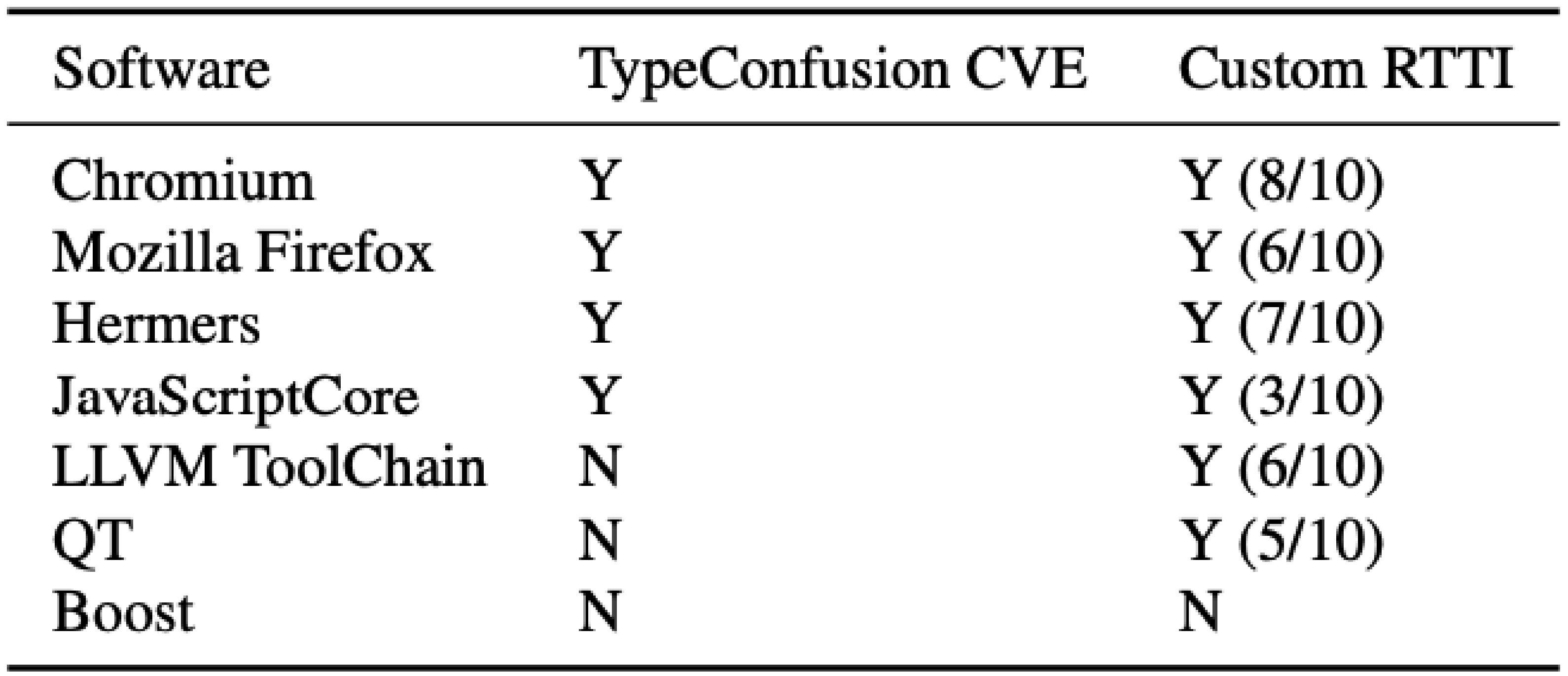
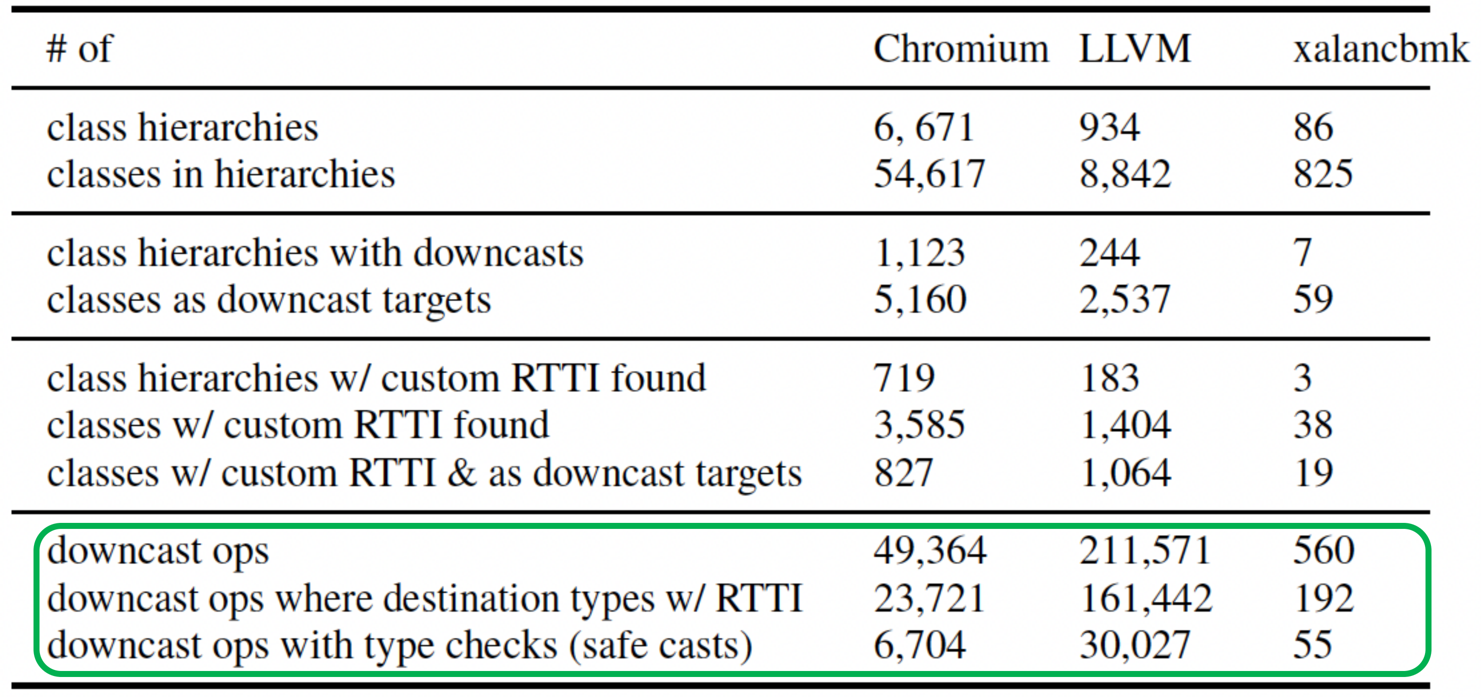


Table 1: Prevalence of the custom RTTI in large scale C/C++ software.

Software	TypeConfusion CVE	Custom RTTI
Chromium	Y	Y (8/10)
Mozilla Firefox	Y	Y (6/10)
Hermers	Y	Y (7/10)
JavaScriptCore	Y	Y (3/10)
LLVM ToolChain	N	Y (6/10)
QT	N	Y (5/10)
Boost	N	N

Safe Casts -- Statically

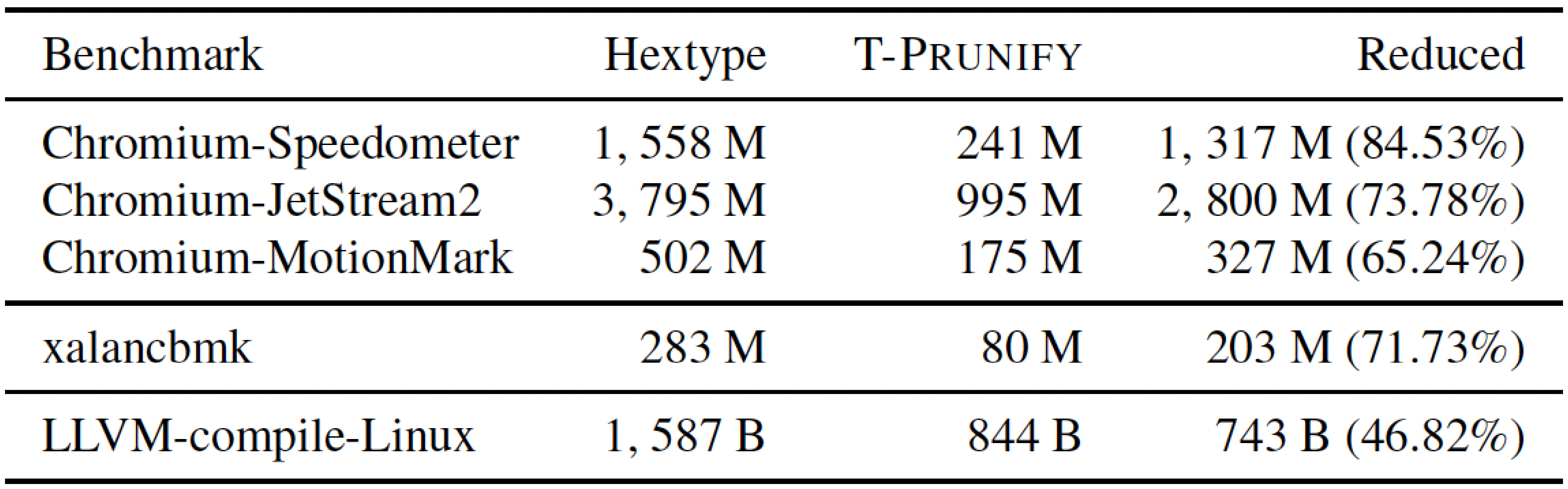
Table 3: Overall statistics of the results.



# of	Chromium	LLVM	xalancbmk
class hierarchies	6, 671	934	86
classes in hierarchies	54,617	8,842	825
class hierarchies with downcasts	1,123	244	7
classes as downcast targets	5,160	2,537	59
class hierarchies w/ custom RTTI found	719	183	3
classes w/ custom RTTI found	3,585	1,404	38
classes w/ custom RTTI & as downcast targets	827	1,064	19
downcast ops	49,364	211,571	560
downcast ops where destination types w/ RTTI	23,721	161,442	192
downcast ops with type checks (safe casts)	6,704	30,027	55

Safe Casts -- Dynamically

Table 6: Number of dynamic cast verification performed by HexType versus TPRunify.



Benchmark	Hextype	T-PRUNIFY	Reduced
Chromium-Speedometer	1, 558 M	241 M	1, 317 M (84.53%)
Chromium-JetStream2	3, 795 M	995 M	2, 800 M (73.78%)
Chromium-MotionMark	502 M	175 M	327 M (65.24%)
xalancbmk	283 M	80 M	203 M (71.73%)
LLVM-compile-Linux	1, 587 B	844 B	743 B (46.82%)

Overhead Reduction

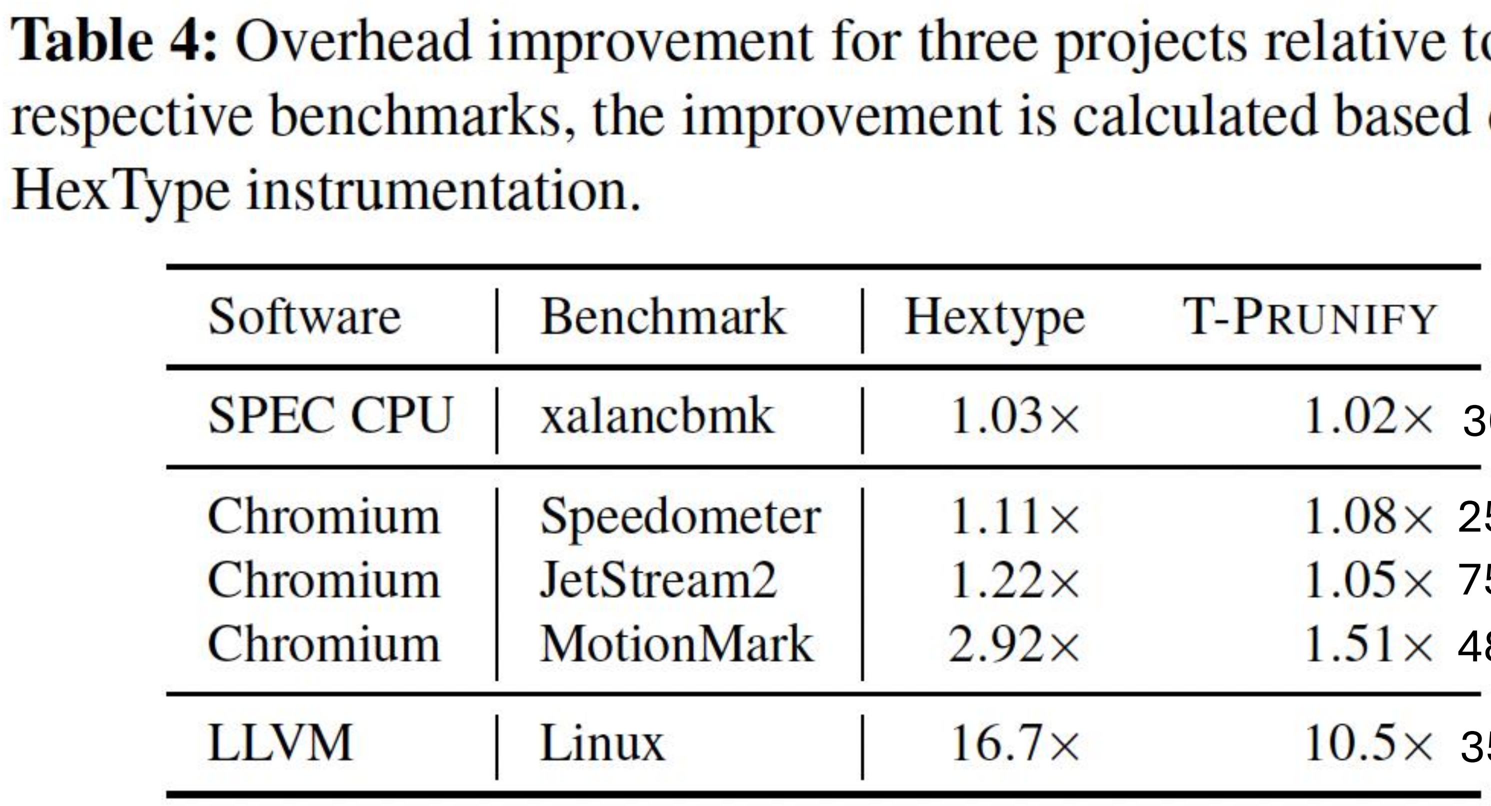


Table 4: Overhead improvement for three projects relative to their respective benchmarks, the improvement is calculated based on the HexType instrumentation.

Software	Benchmark	Hextype	T-PRUNIFY	
SPEC CPU	xalancbmk	1.03×	1.02×	30%↓
Chromium	Speedometer	1.11×	1.08×	25%↓
Chromium	JetStream2	1.22×	1.05×	75%↓
Chromium	MotionMark	2.92×	1.51×	48%↓
LLVM	Linux	16.7×	10.5×	35%↓

Conclusion

- Custom RTTI is widely used in preventing type confusion
- **TPRunify:** An automatic tool to combine the sanitizers' approach and custom RTTI to ensures full protection with minimal overhead to mitigate type confusion.
- **Evaluation:** Reduce the sanitizer's overhead by *25%* to *75%* for large scale C++ software.
- **Open Source:** https://github.com/seclab-ucr/TPrunify.git

Q & A

In job market this year!