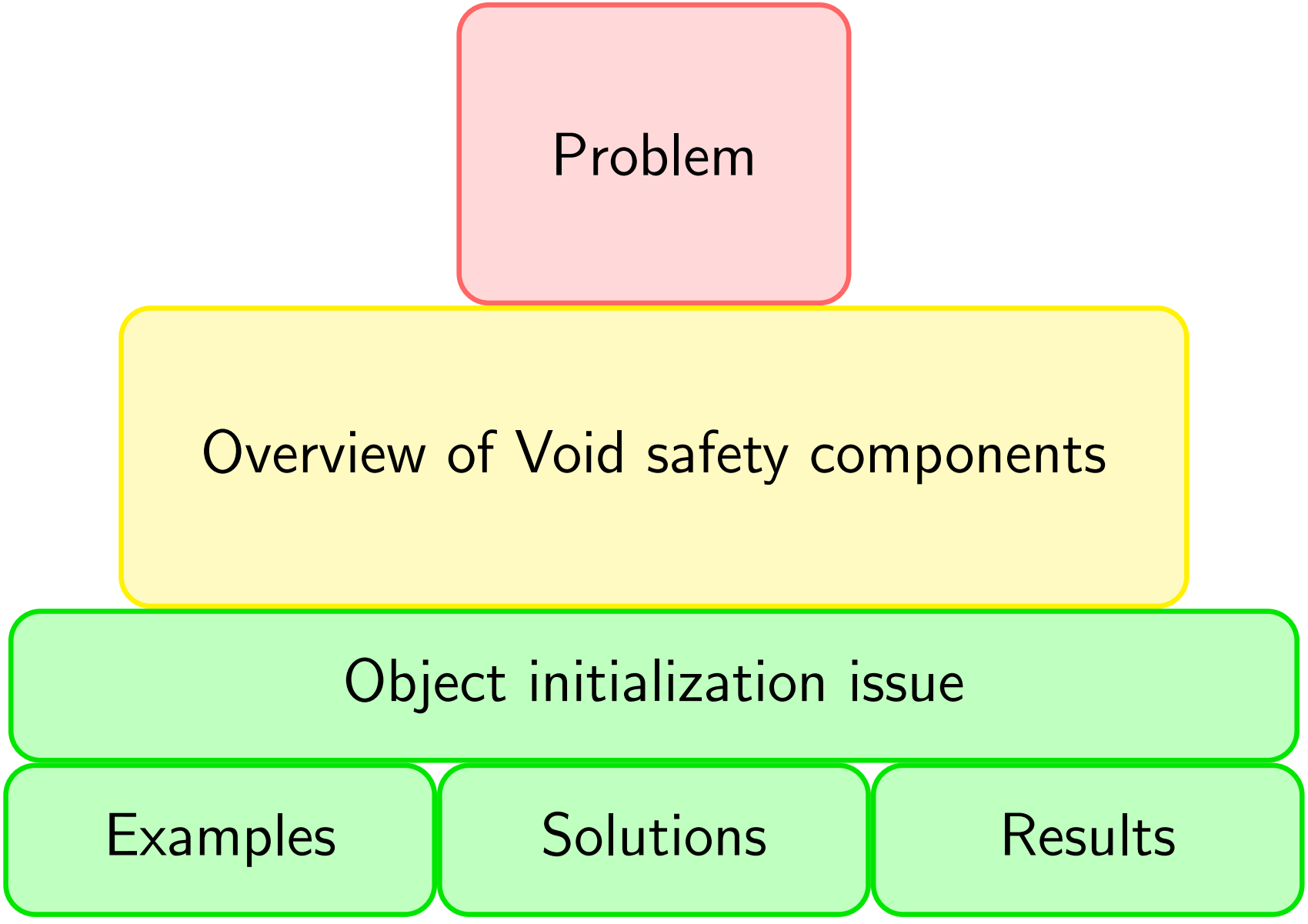
Making Void Safety Practical

Alexander Kogtenkov

2017-06-26


Problem
Overview of Void safety components
Object initialization issue
Examples
Solutions
Results

Problem

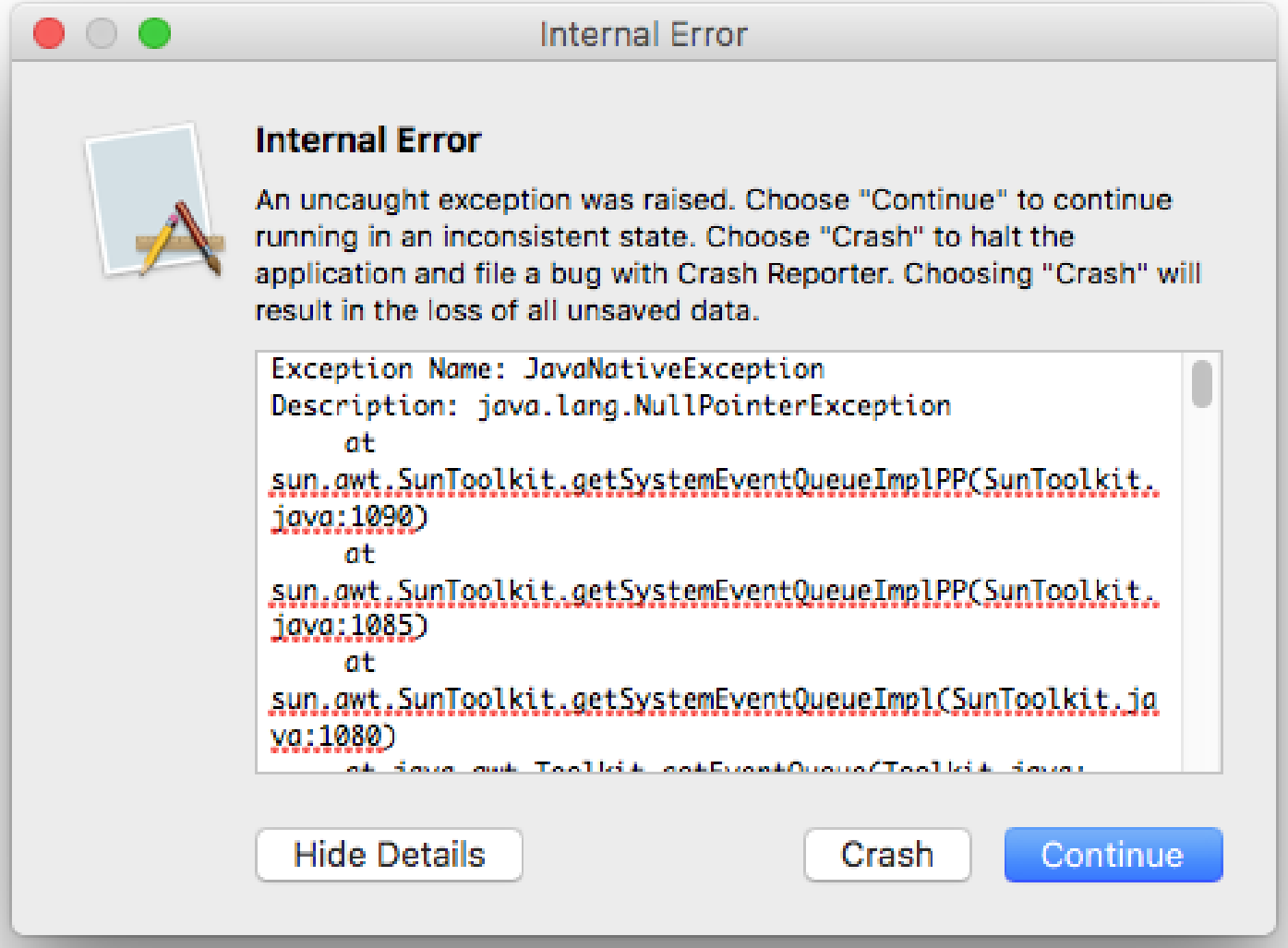
expr.method (args);

Problem

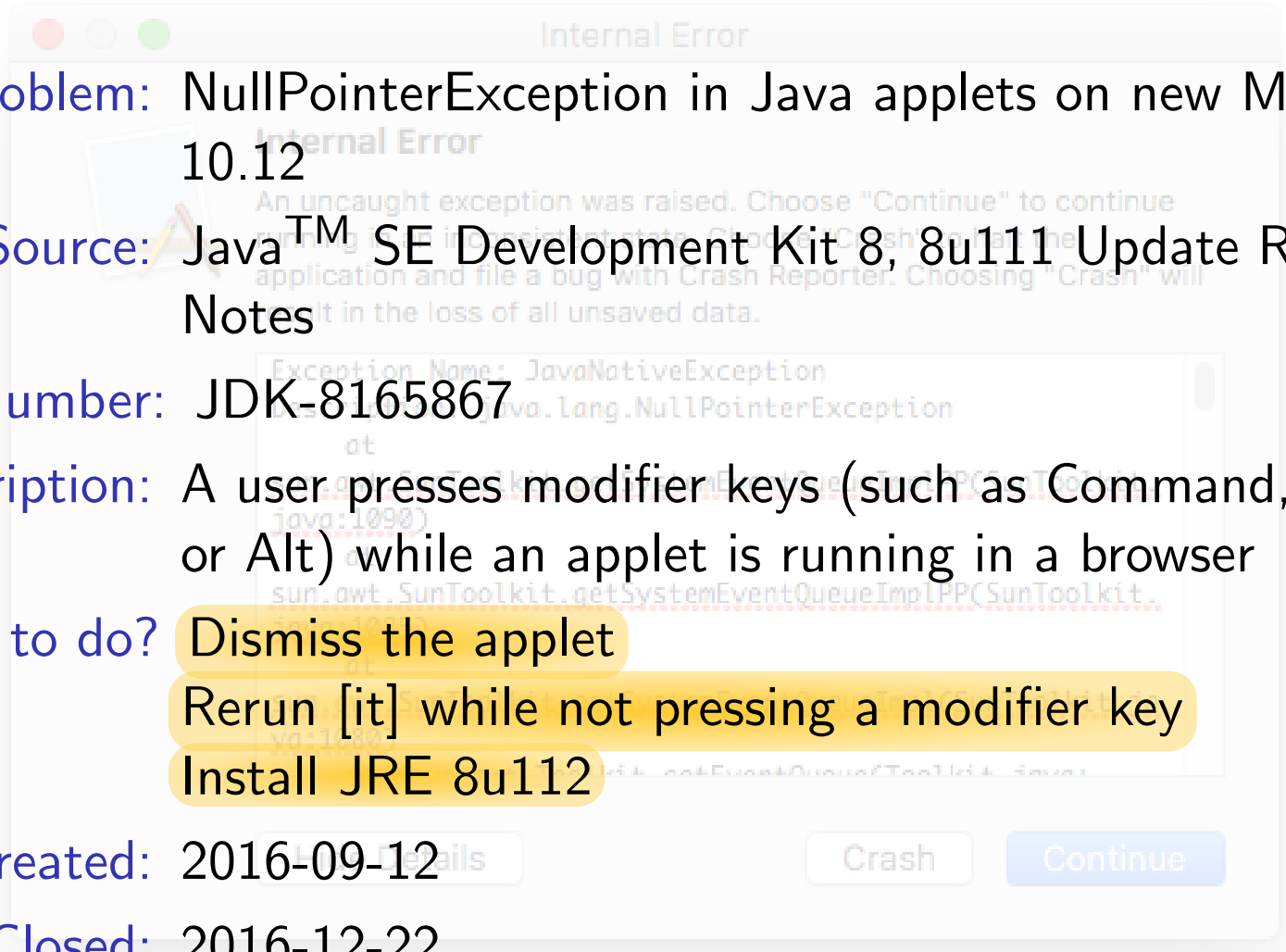
expr.method (args);

```
tmp = expr;
if (tmp == null)
        throw new NullPointerException ();
else
        tmp.method (args);
```

NullPointerException in Java applets on new MacOS 10.12



NullPointerException in Java applets on new MacOS 10.12



Problem: NullPointerException in Java applets on new MacOS 10.12

Source: Java™ SE Development Kit 8, 8u111 Update Release Notes

Bug number: JDK-8165867

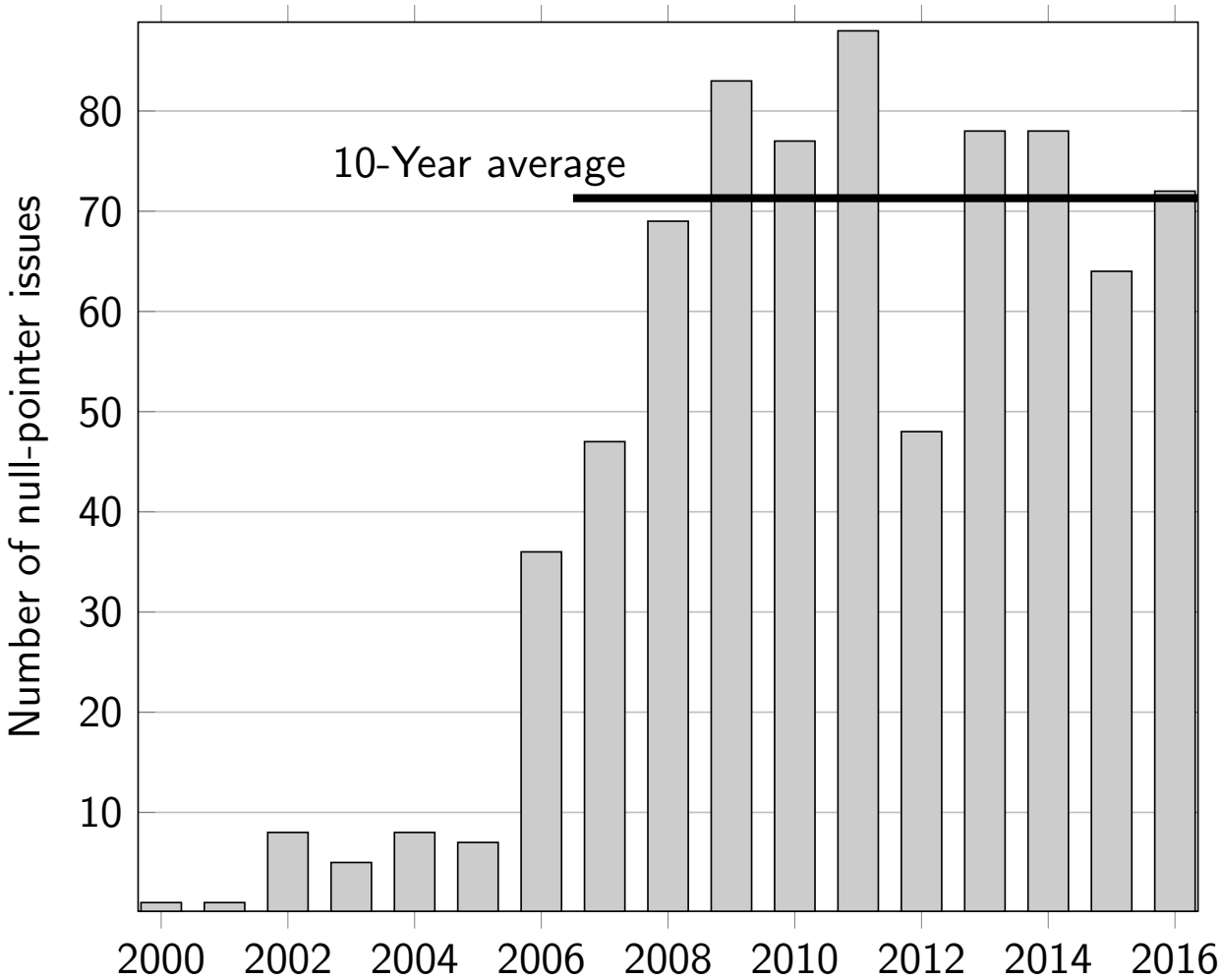
Description: A user presses modifier keys (such as Command, Shift, or Alt) while an applet is running in a browser

What to do? Dismiss the applet
Rerun [it] while not pressing a modifier key
Install JRE 8u112

Created: 2016-09-12

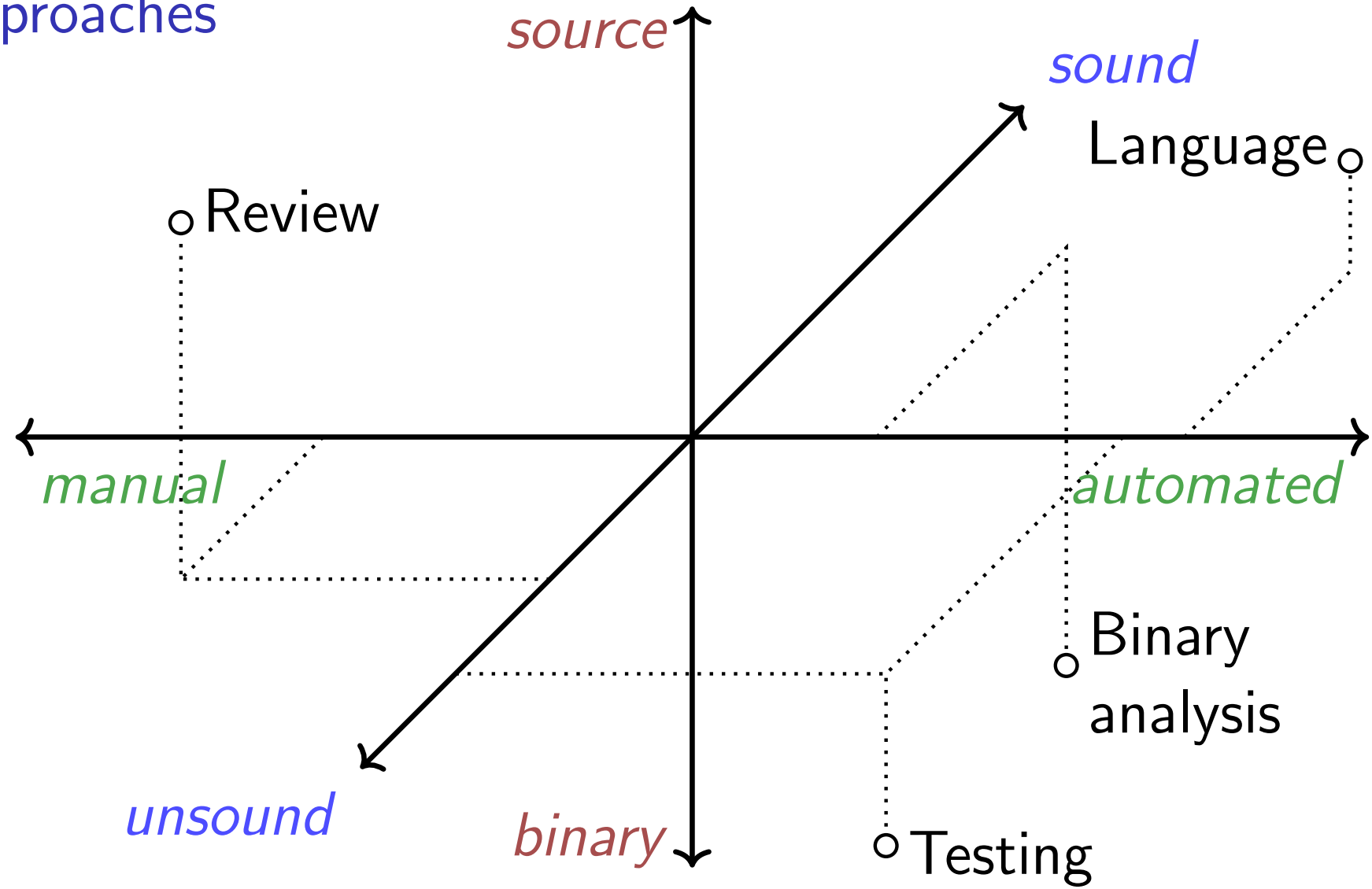
Closed: 2016-12-22

Statistics

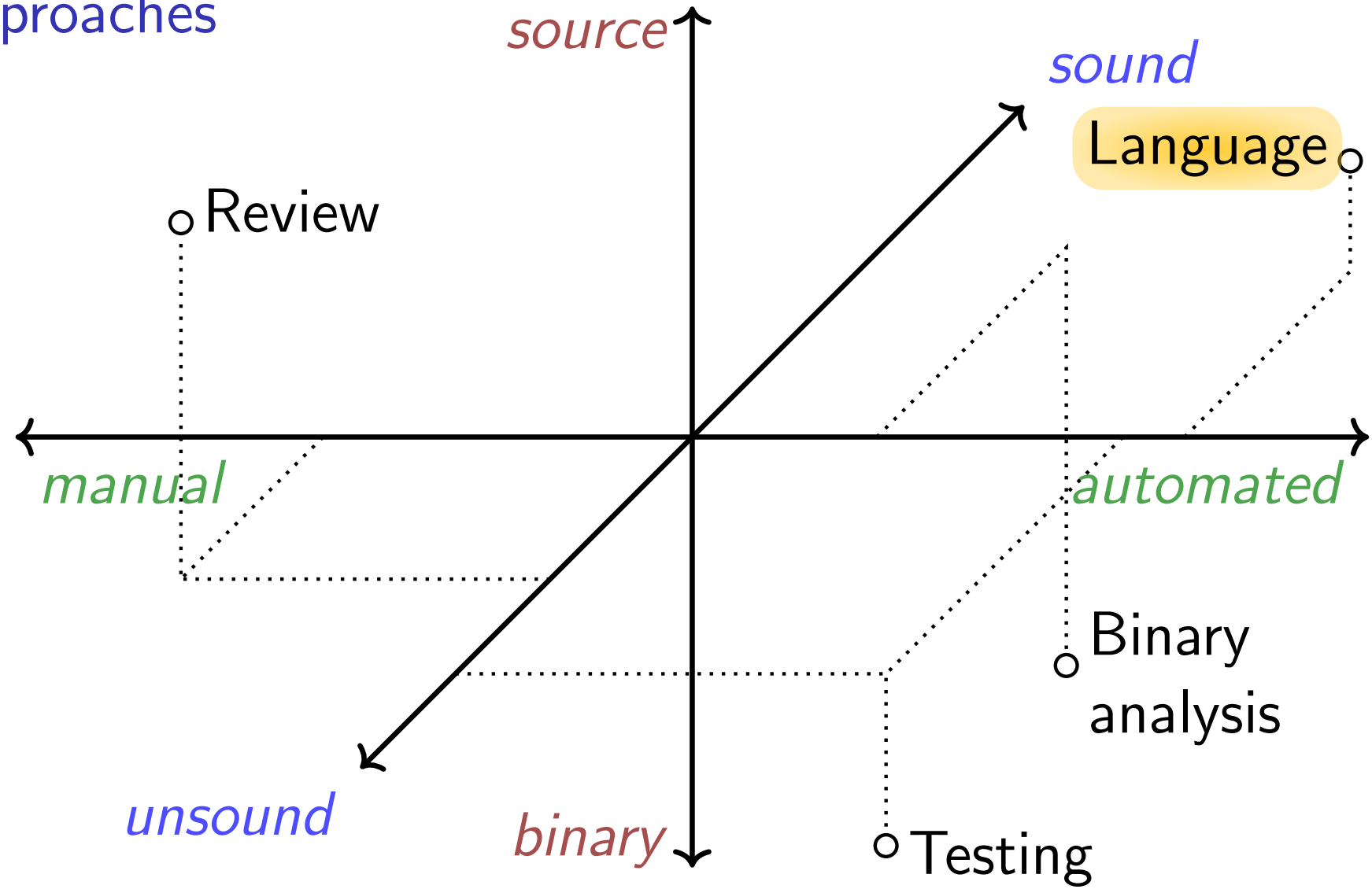


Null pointer issues (such as null pointer dereferencing) in Common Vulnerabilities and Exposures database (funded by CERT)

Approaches


source
sound
Language
Review
manual
automated
Binary
analysis
unsound
binary
Testing

Approaches


source
sound
Language
Review
manual
automated
Binary
analysis
unsound
binary
Testing

Void safety

Run-time ✓

Void safety

Compile-time ✓ $\Longrightarrow$ Run-time ✓

Components of void safety

- Type system
- Object initialization
- Certified attachment patterns (CAP)

Components of void safety

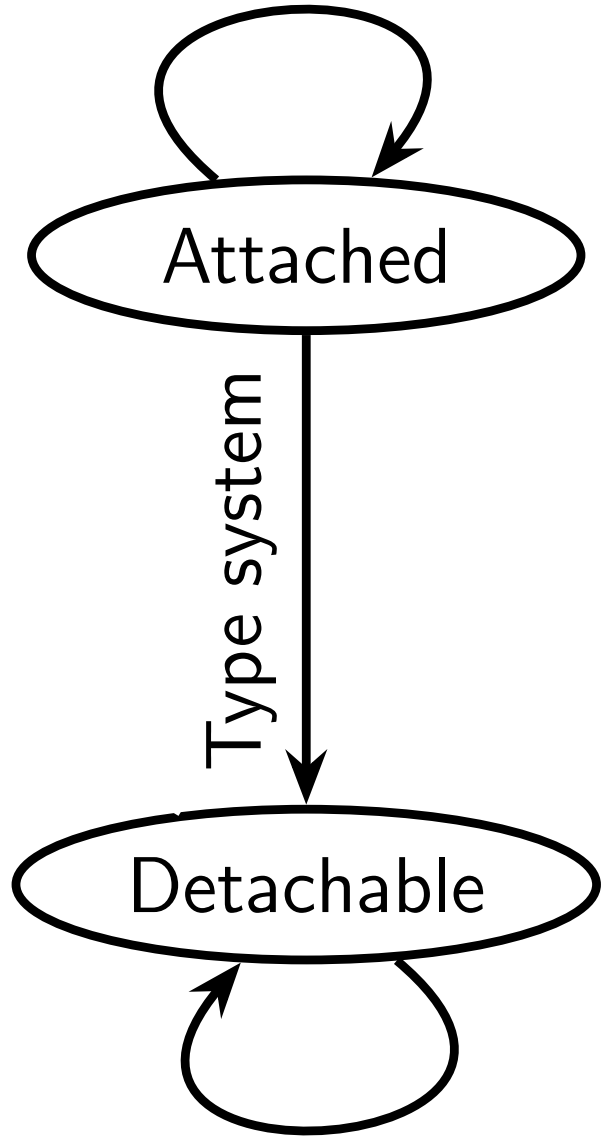
- Type system

target := *source*

		source	
		detachable	**attached**
target	**detachable**	✓	✓
	attached	✗	✓

Components of void safety

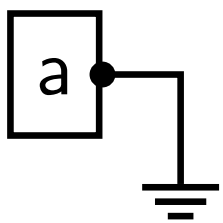
- Type system



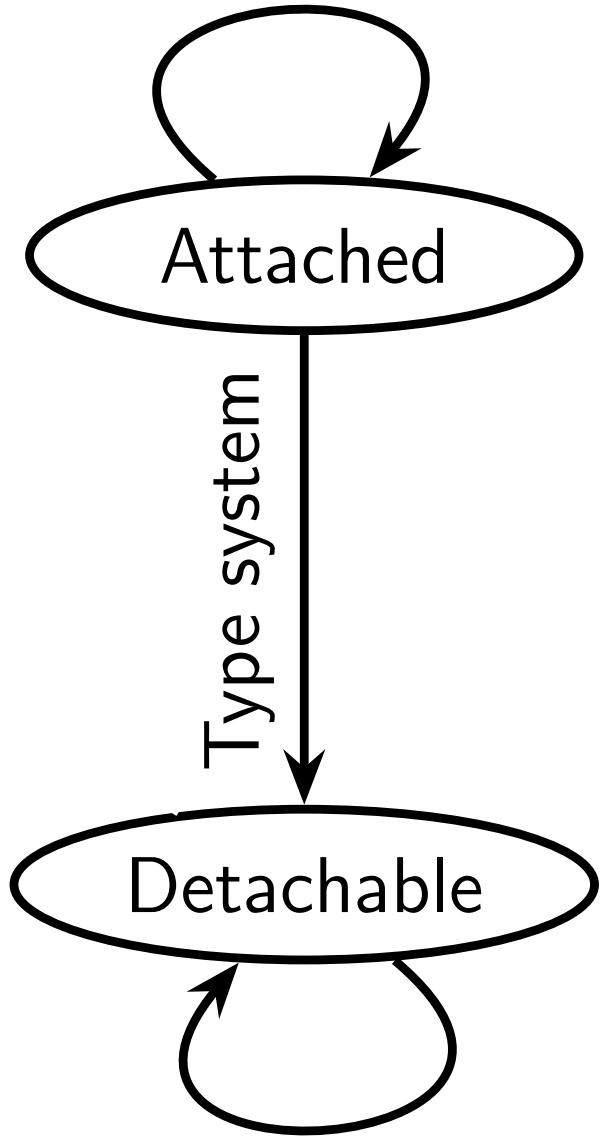
Conformance: $\longrightarrow$

Components of void safety

- Type system
- Object initialization

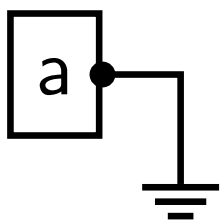


✗

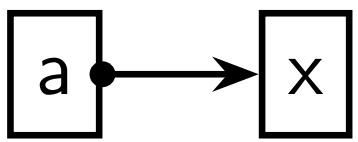


Components of void safety

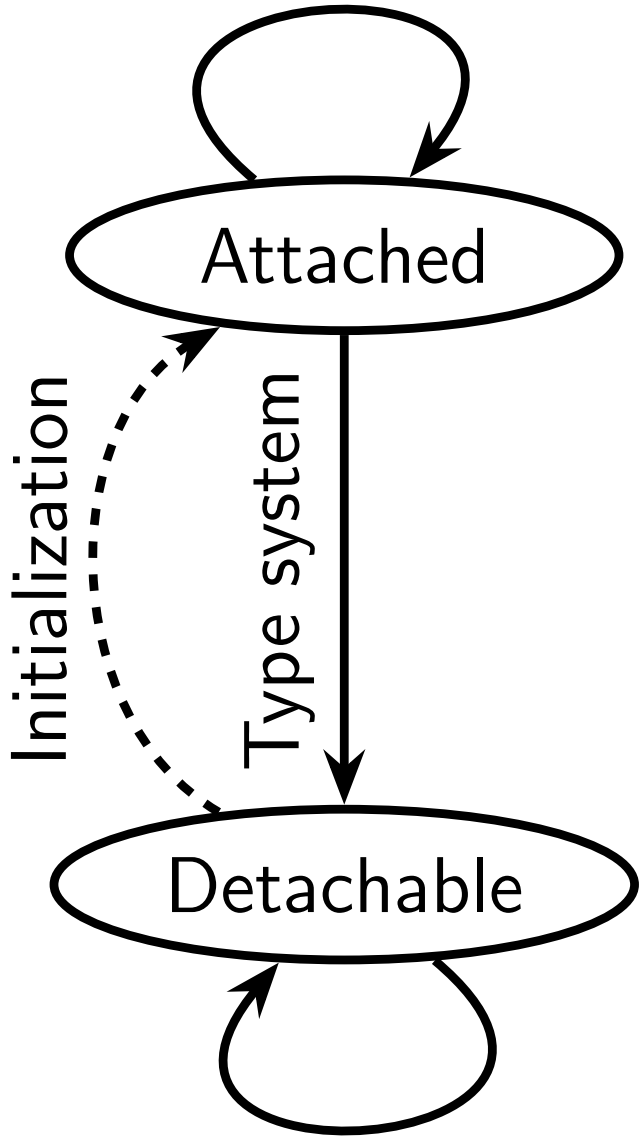
- Type system
- Object initialization



✗



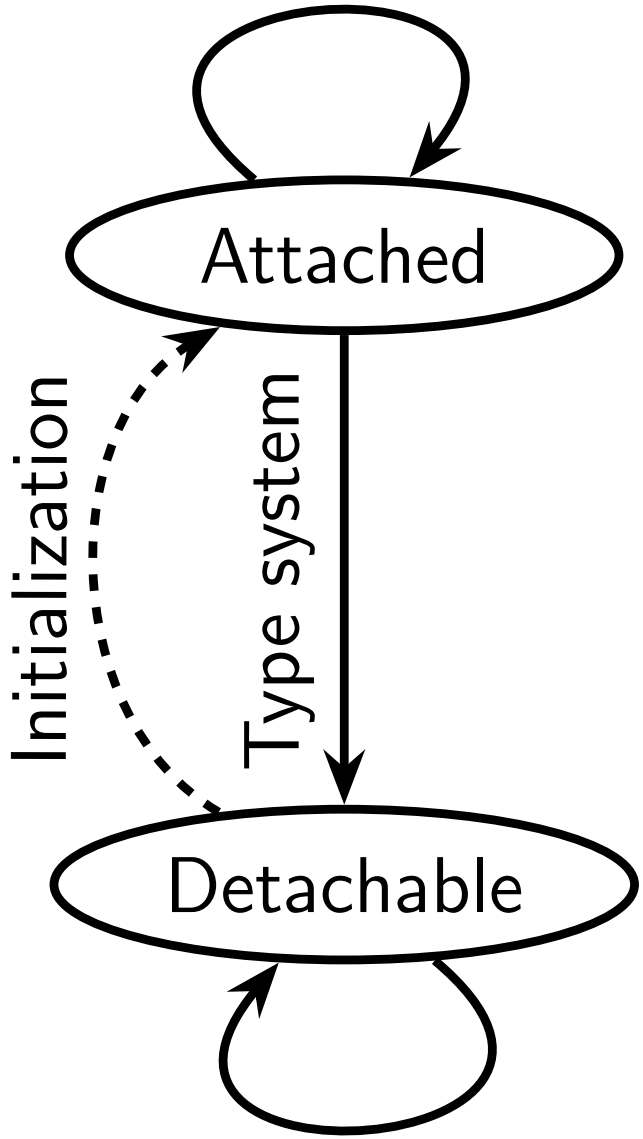
✓



Components of void safety

- Type system
- Object initialization
- Certified attachment patterns (CAP)

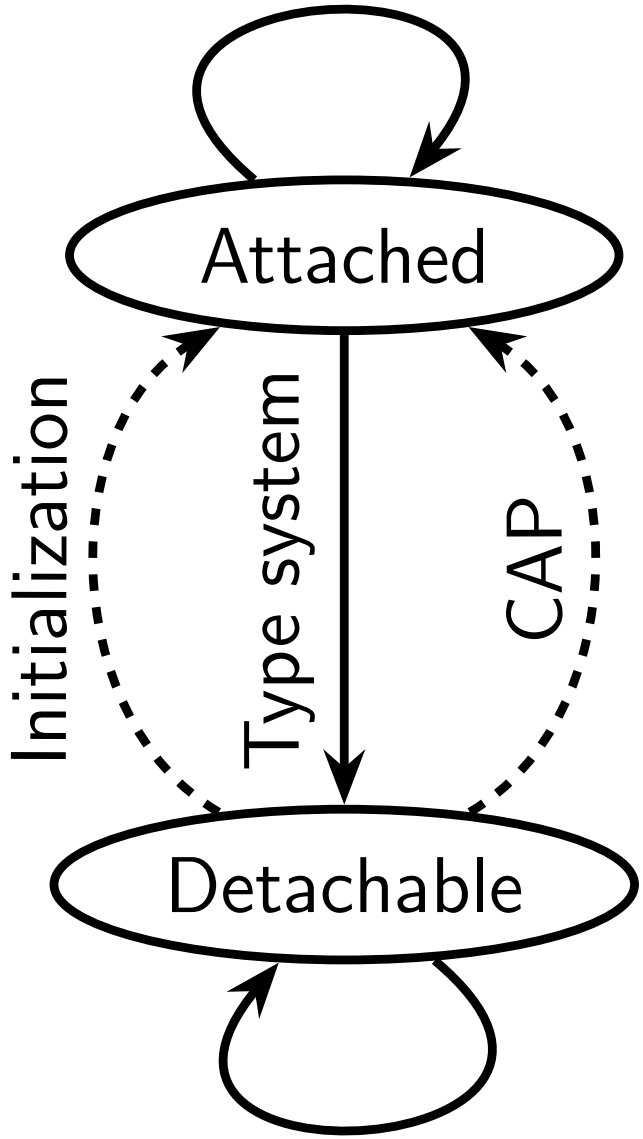
$a := x$ ✗



Components of void safety

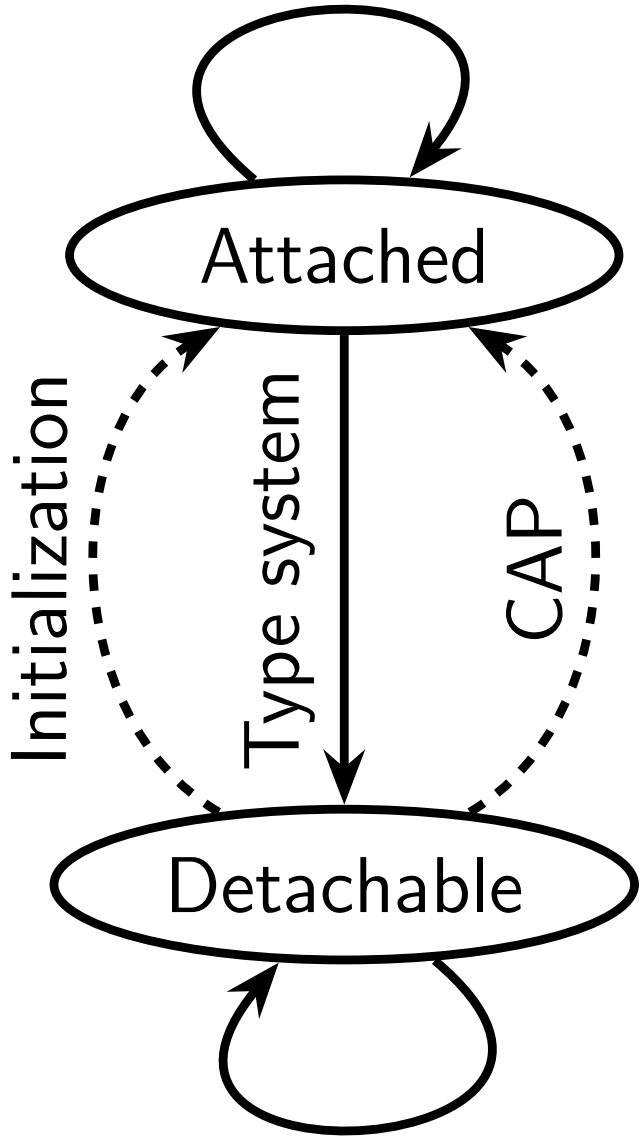
- Type system
- Object initialization
- Certified attachment patterns (CAP)

if $x /= $ **Void then**
    $a := x$ ✓
end



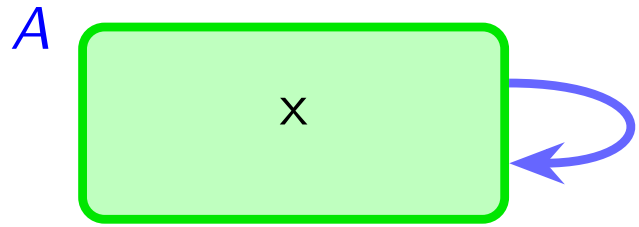
Components of void safety

- Type system
- Object initialization
- Certified attachment patterns (CAP)



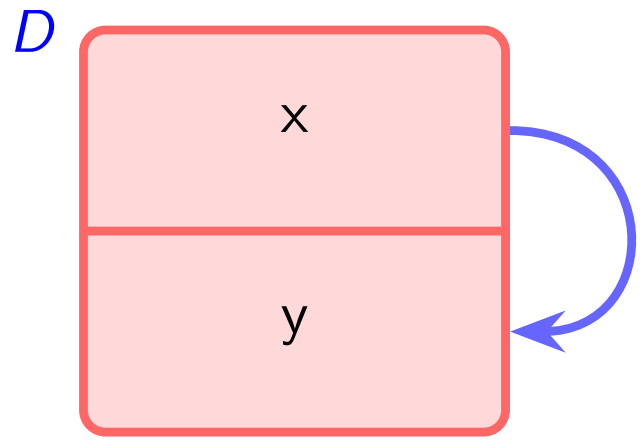
Example 1: polymorphic call

```
class A create make feature
    make do
            x := ...
            f
        end
    x: X
    init do ... end
    f do ... end
end
```



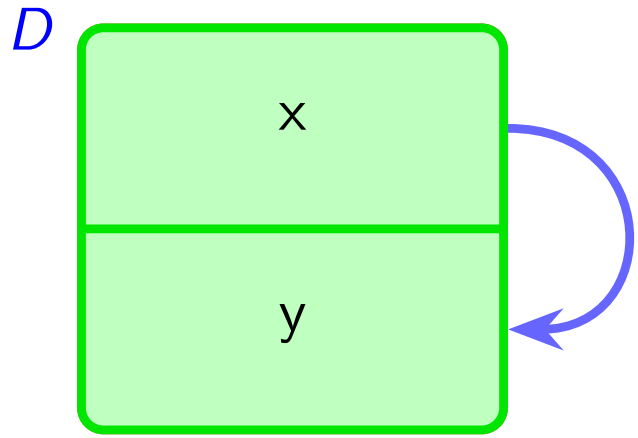
Example 1: polymorphic call

```
class A create make feature
    make do
            x := ...
            f
        end
    x: X
    init do ... end
    f do ... end
end

class D inherit A redefine init, f end
create make feature
    y: Y
    init do y := ... end
    f    do y.something end
end
```



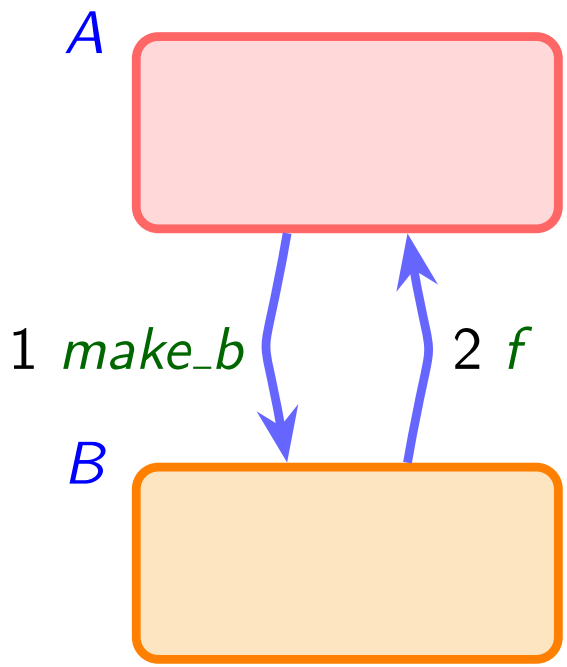
Example 1: polymorphic call

```
class A create make feature
    make do
            x := ...
            init; f
        end
    x: X
    init do ... end
    f do ... end
end

class D inherit A redefine init, f end
create make feature
    y: Y
    init do y := ... end
    f    do y.something end
end
```



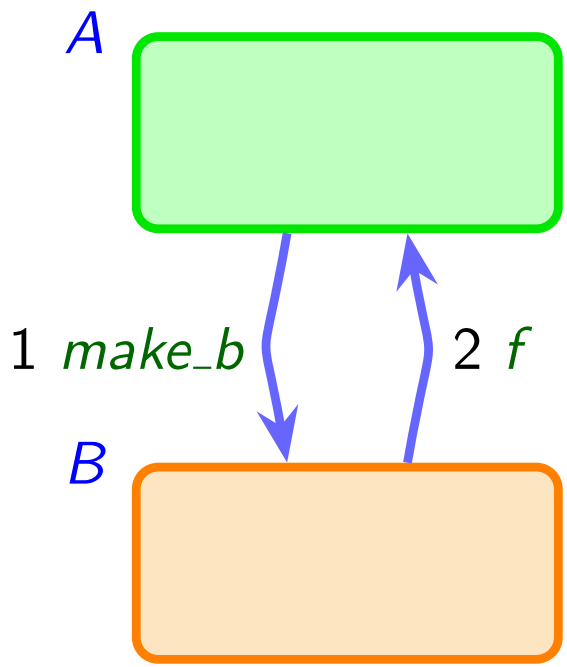
Example 2: callback

```
class A create make feature
    make_a local b: B
        do

            create b.make_b (Current)
            x := ...
        end
    x: X
    f do x.something end
end

class B create make_b feature
    make_b (a: A) do ... a.f ... end
end
```



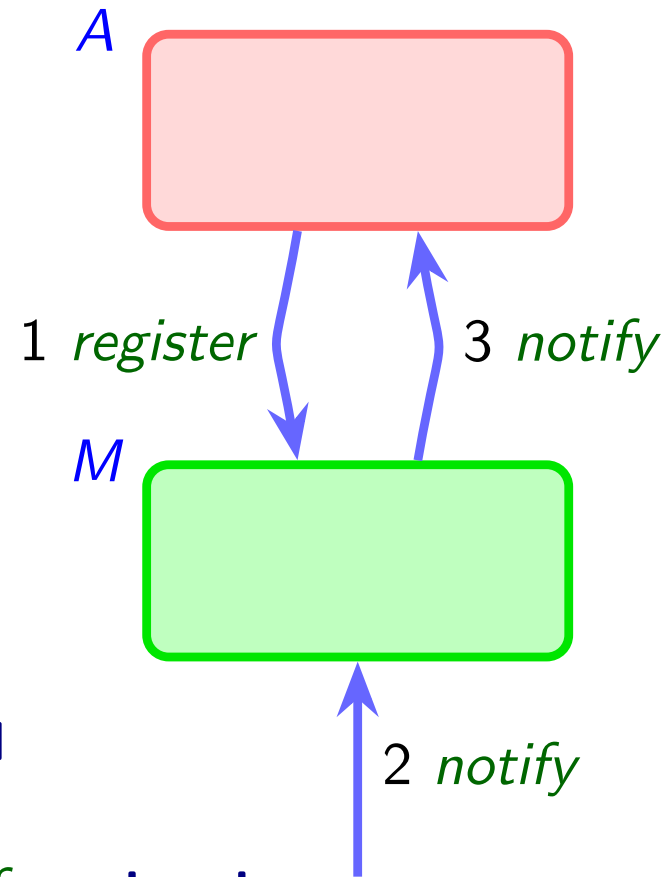
Example 2: callback

```
class A create make feature
    make_a local b: B
        do
            x := ...
            create b.make_b (Current)

        end
    x: X
    f do x.something end
end

class B create make_b feature
    make_b (a: A) do ... a.f ... end
end
```



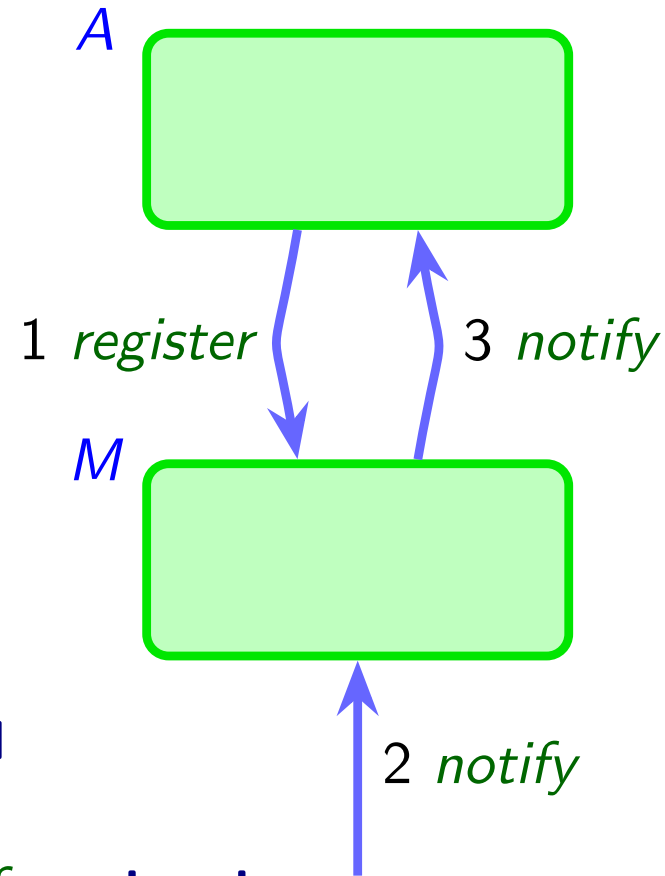
Example 3: mediator

```
class A create make feature
    make (m: M) do

            m.register (Current)
            x := ...
        end
    x: X
    notify do x.something end
end

class M ... feature
    register (a: A) do list.extend (a) end
    notify
        do across list as c do c.item.notify end end
end
```



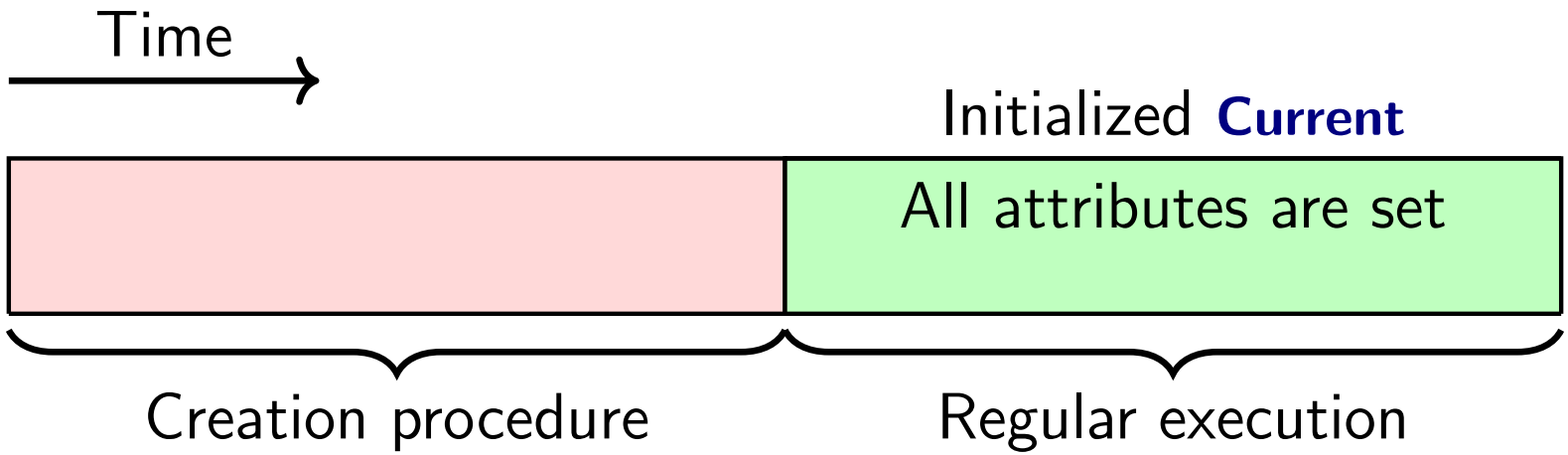
Example 3: mediator

```
class A create make feature
    make (m: M) do
            x := ...
            m.register (Current)

        end
    x: X
    notify do x.something end
end

class M ... feature
    register (a: A) do list.extend (a) end
    notify
        do across list as c do c.item.notify end end
end
```



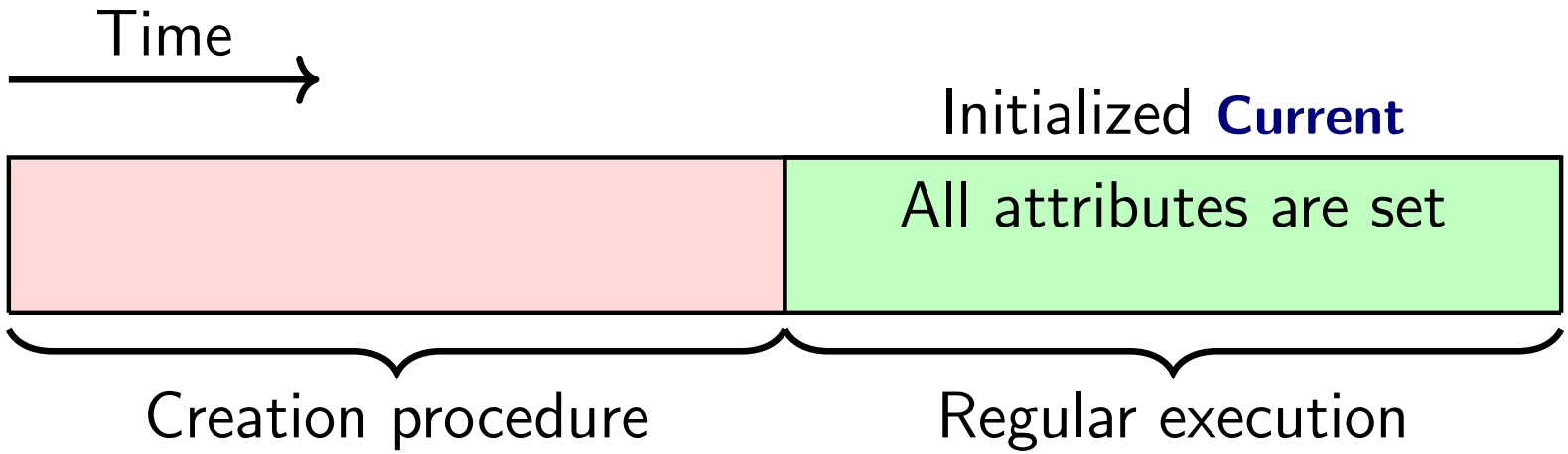
How can it be done?

Alexander J. Summers and Peter Müller
Freedom Before Commitment



How can it be done?

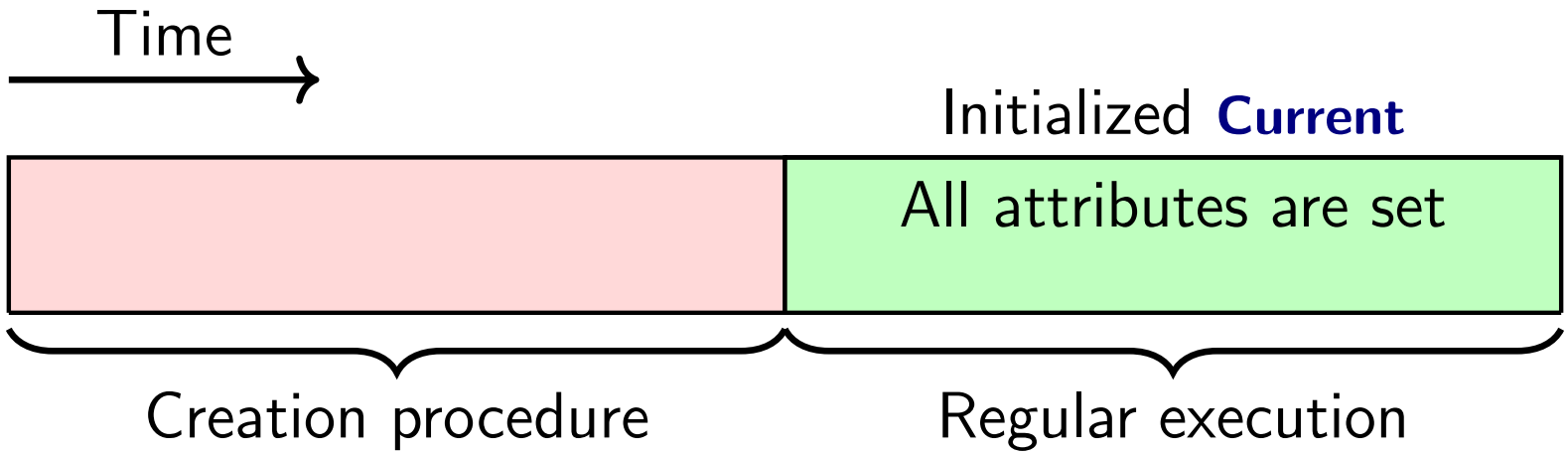
Alexander J. Summers and Peter Müller
Freedom Before Commitment



Polymorphic call	✓
Callback	?
Mediator	✗

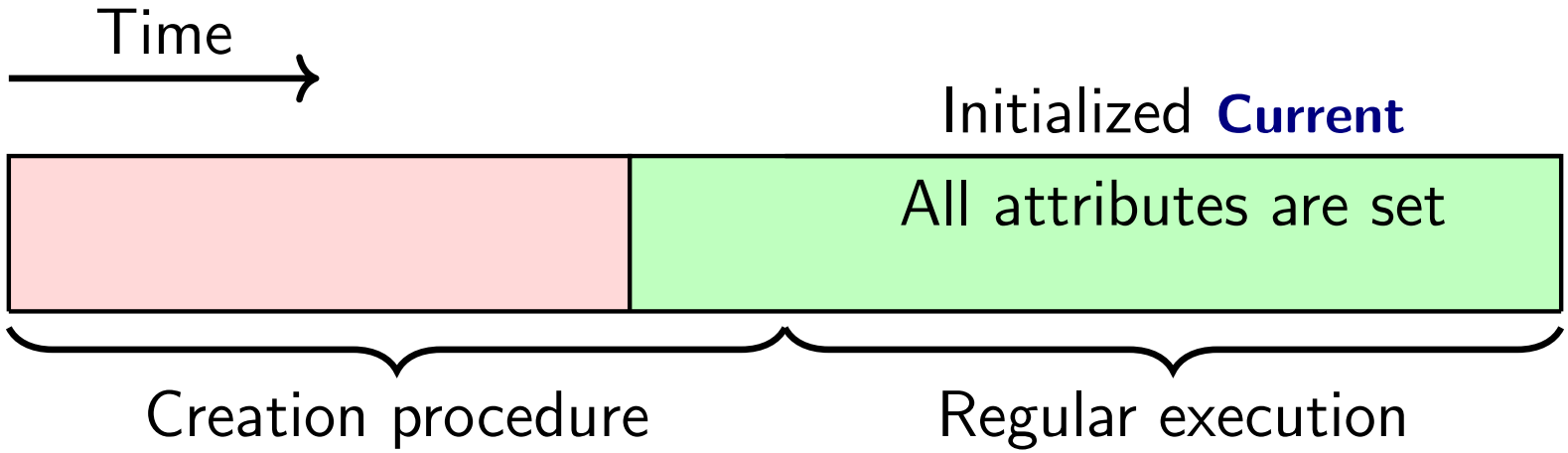
How can it be done?

Alexander J. Summers and Peter Müller
Freedom Before Commitment



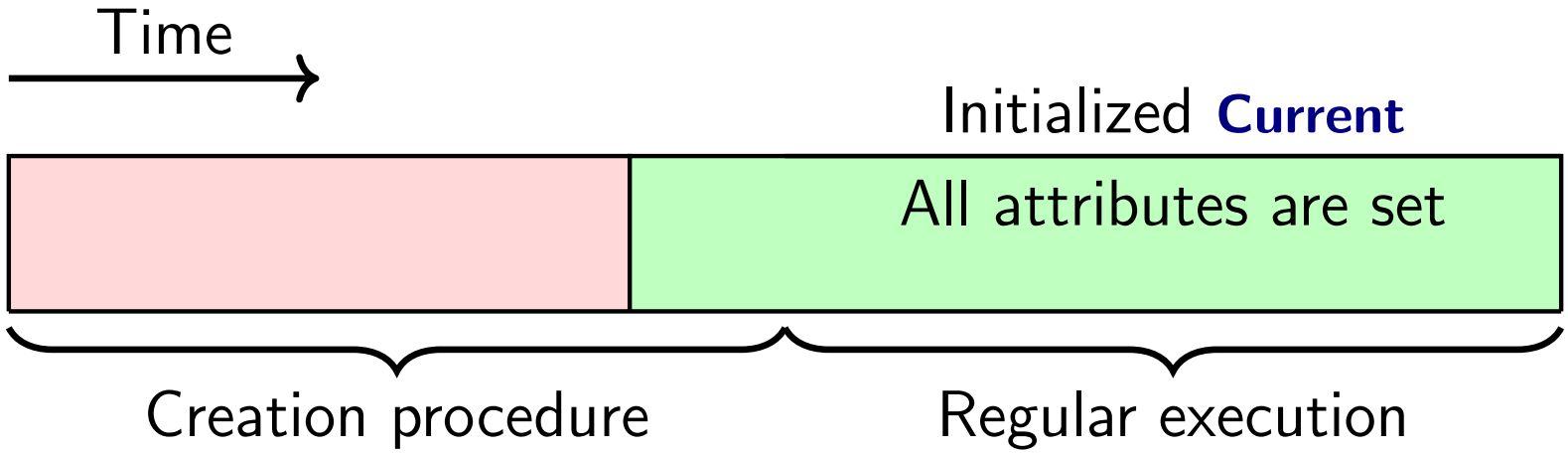
- Additional annotations
- Cannot register new objects in existing structures

Static analysis: simple rule



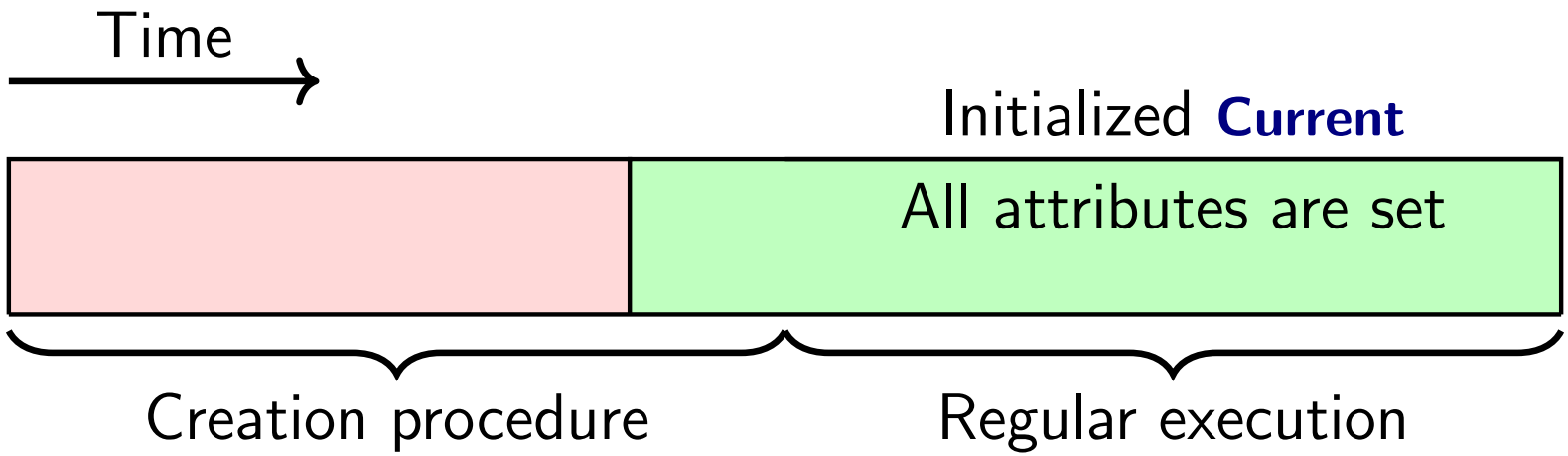
$$\frac{V = \varnothing}{V \vdash \mathit{Current}\ \surd_c'}\ \textsc{Current}$$

Static analysis: simple rule



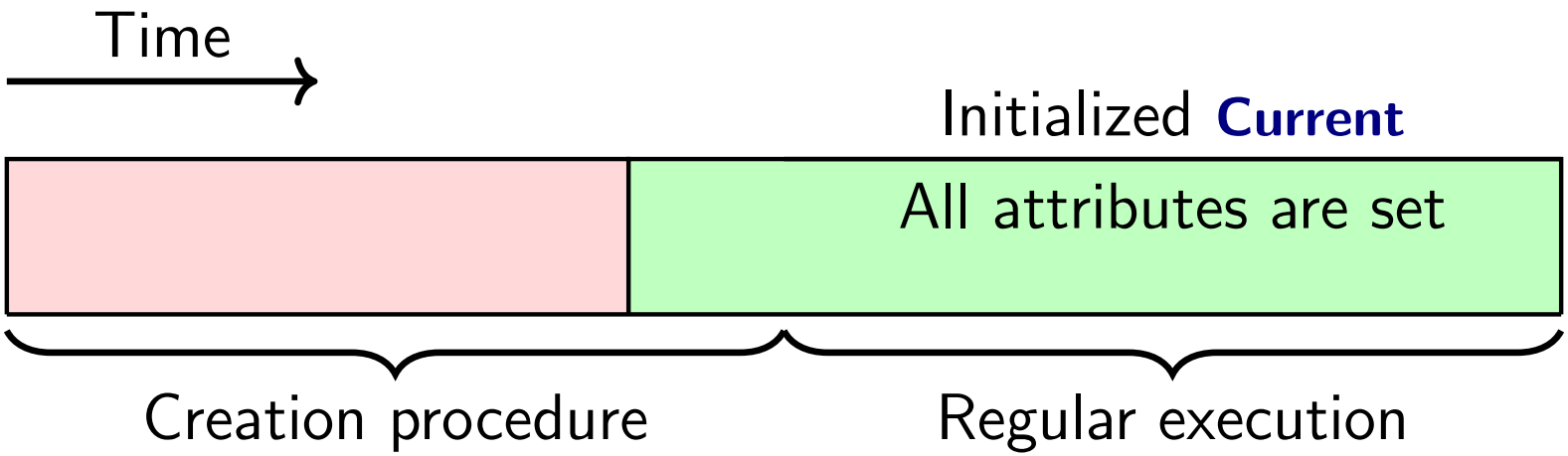
Polymorphic call ✓
Callback ✓
Mediator ✓

Static analysis: simple rule



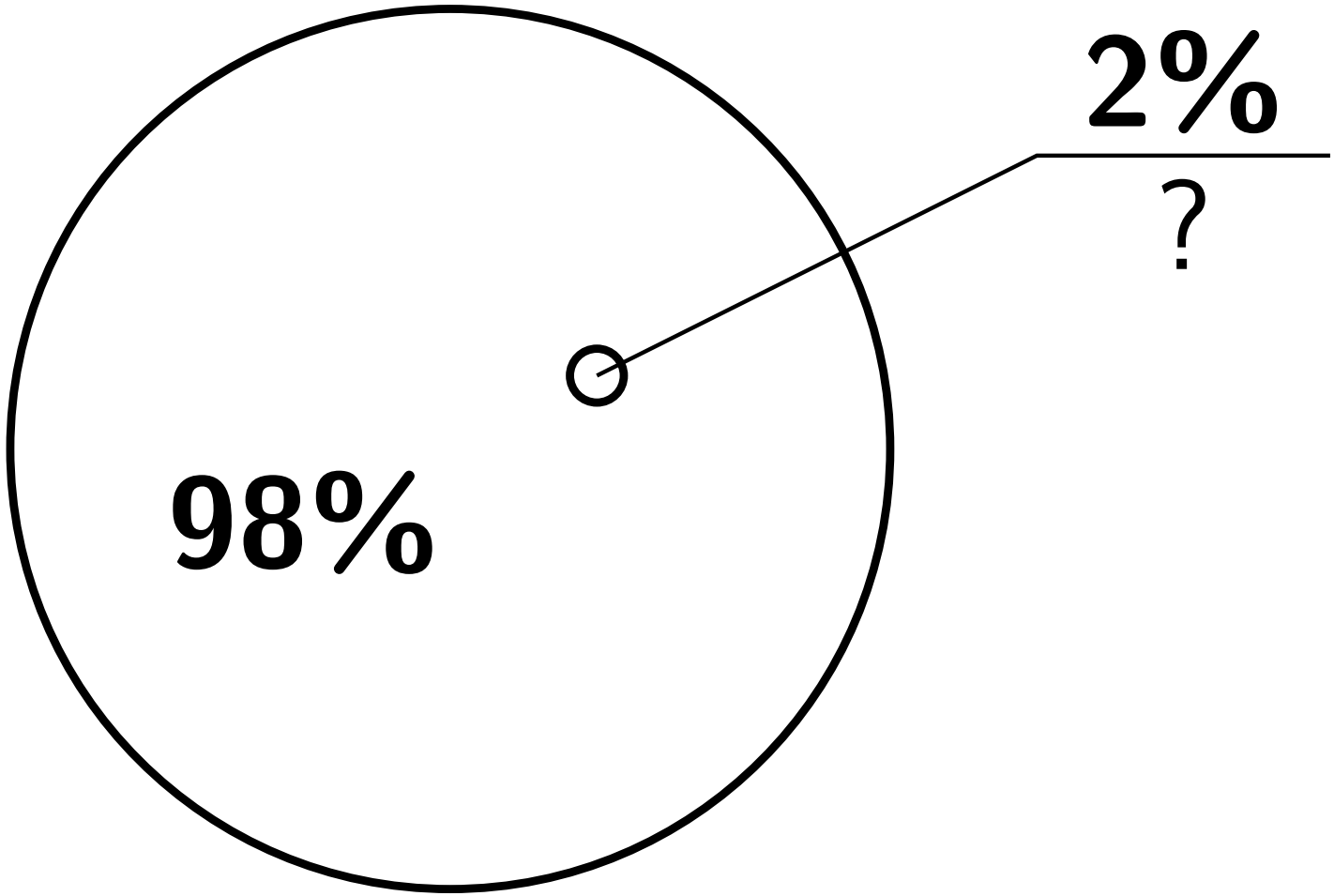
- No annotations
- All scenarios

Static analysis: simple rule

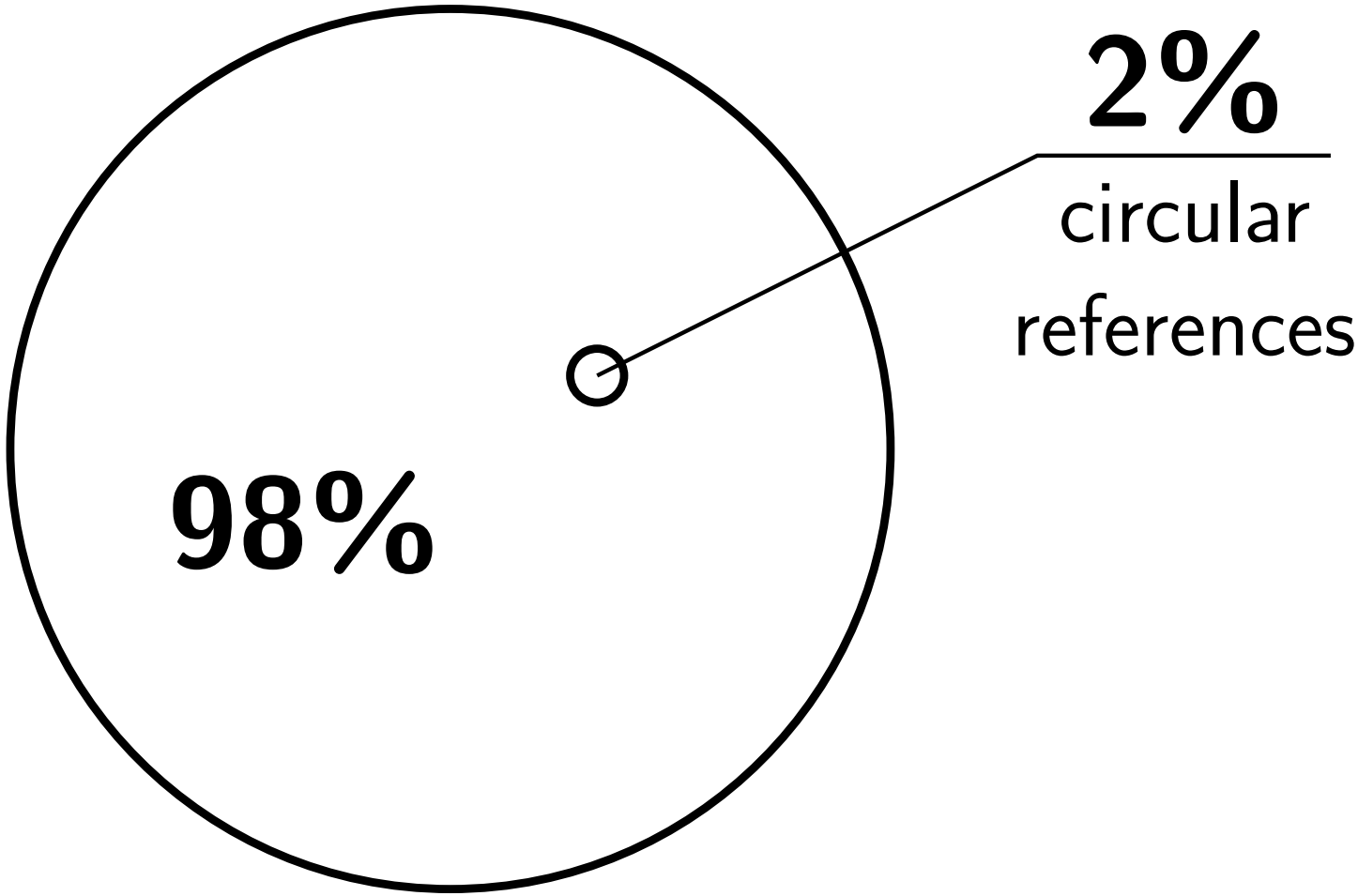


- No annotations
- All scenarios so far

Simple rule: practice

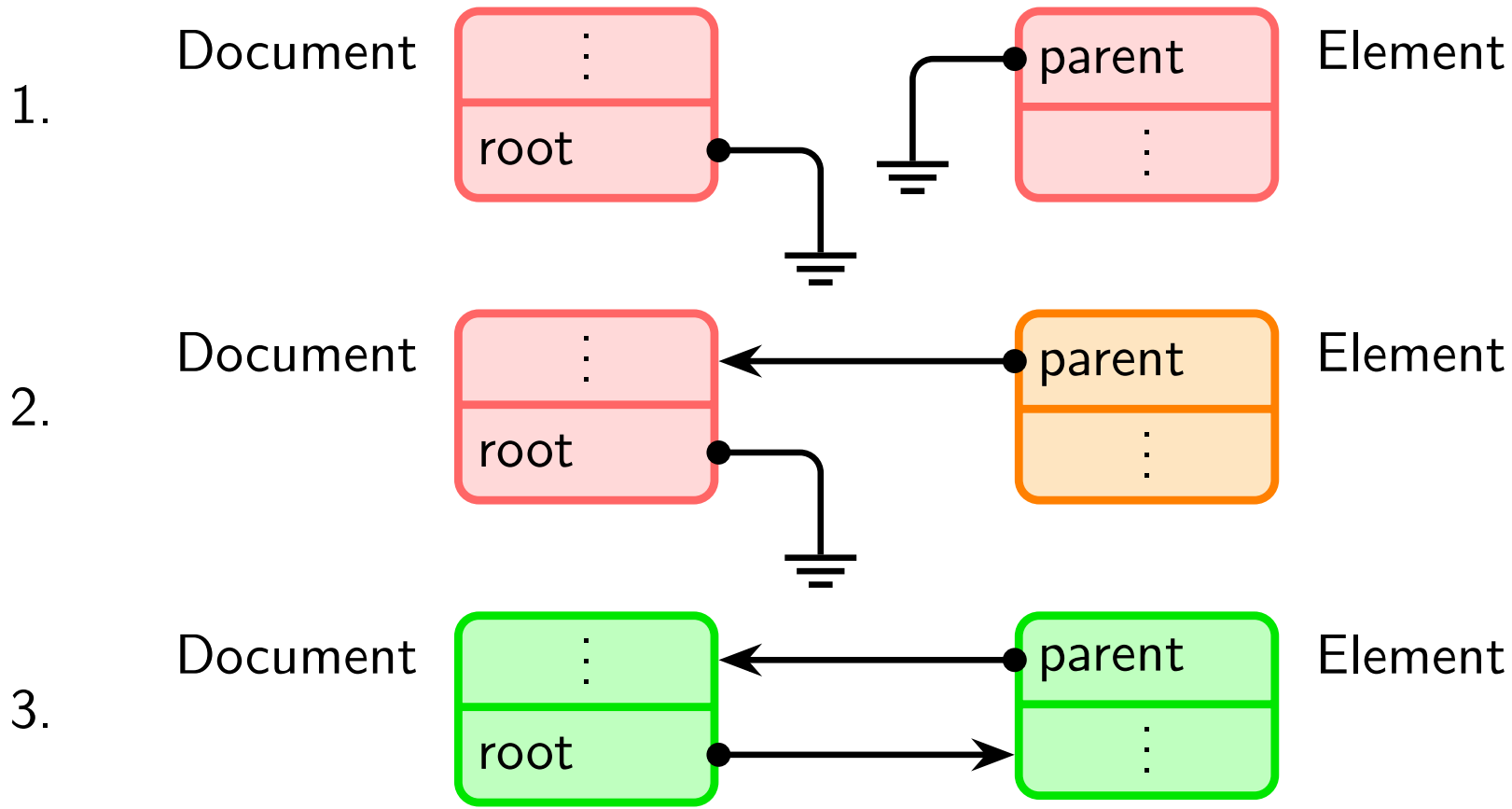


Simple rule: practice

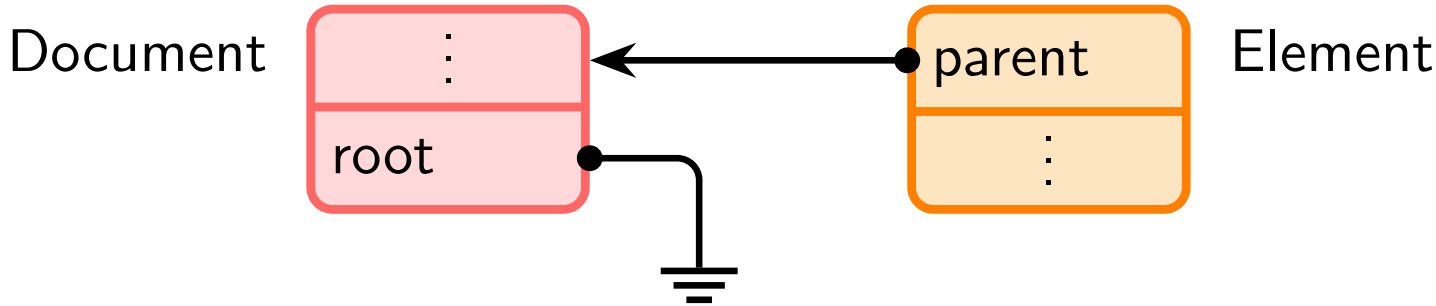


Circular references

class *DOCUMENT* **feature** *root*: *ELEMENT* ... **end**
class *ELEMENT* **feature** *parent*: *DOCUMENT* ... **end**



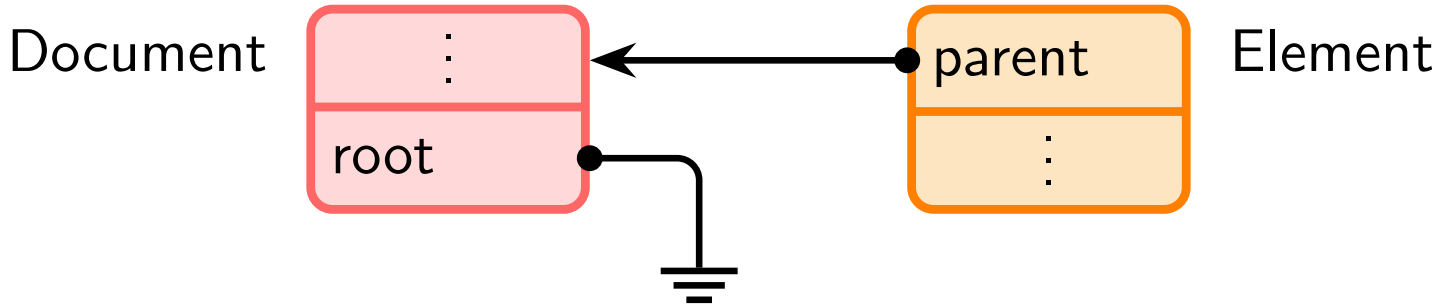
Q: What could go wrong?




```
class DOCUMENT
create make feature
    root: ELEMENT
    make
        do
            create
                root.make (Current)
        end
end
```

```
class ELEMENT
create make feature
    parent: DOCUMENT
    make (p: DOCUMENT)
        do
            parent := p
        end
end
```

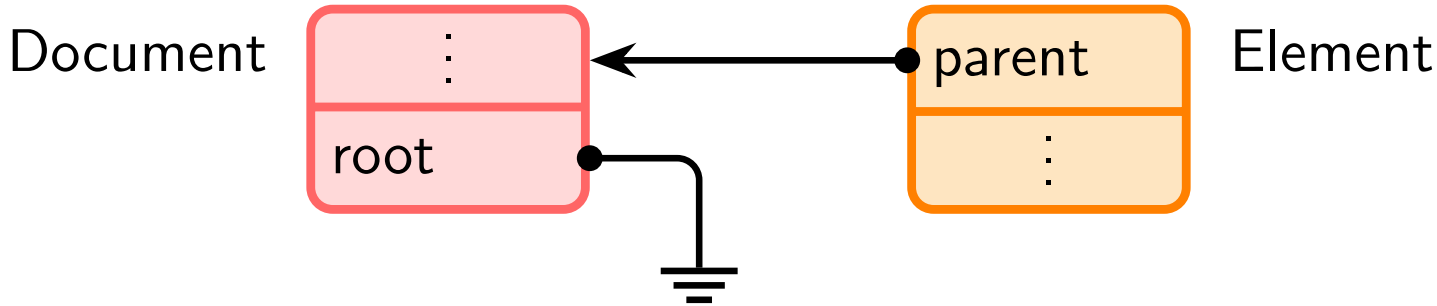
Q: What could go wrong?




```
class DOCUMENT
create make feature
    root: ELEMENT
    make
        do
            create
                root.make (Current)
        end
end
```

```
class ELEMENT
create make feature
    parent: DOCUMENT
    make (p: DOCUMENT)
        do
            parent := p
            parent.root.something
        end
end
```

Q: What could go wrong?

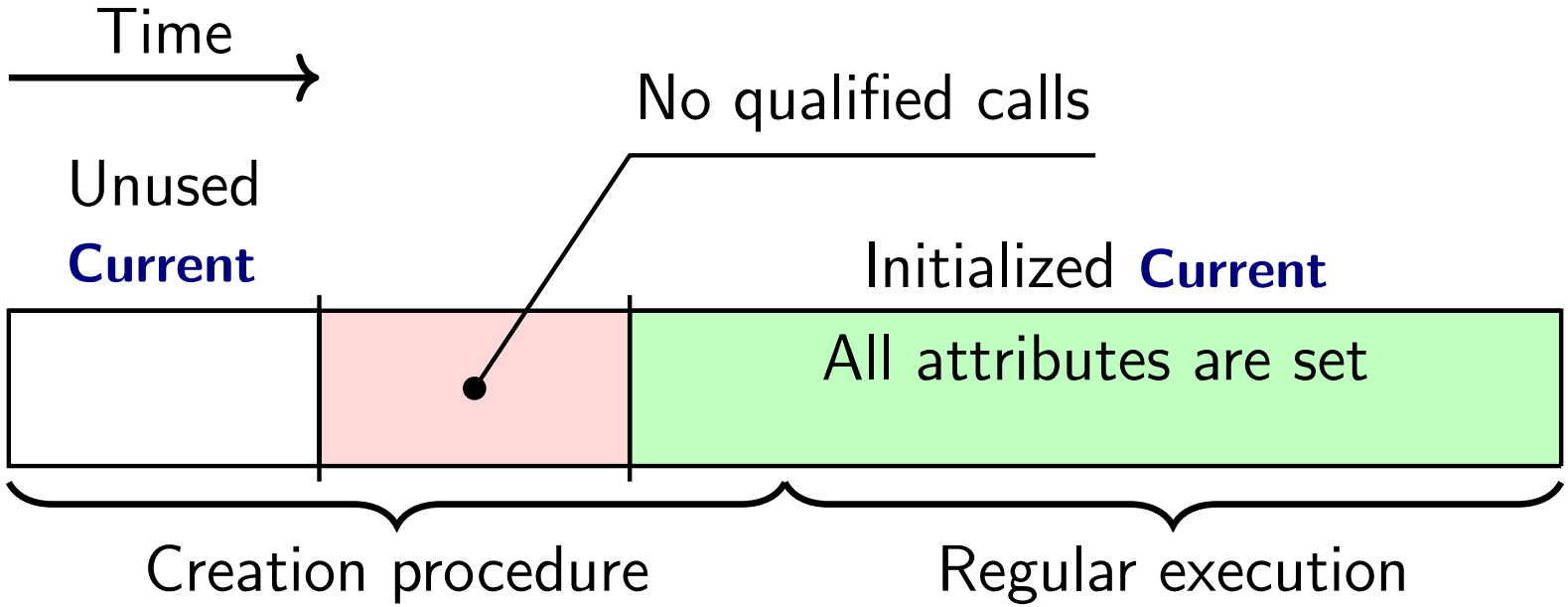



```
class DOCUMENT
create make feature
    root: ELEMENT
    make
        do
            create
                root.make (Current)
        end
end
```

```
class ELEMENT
create make feature
    parent: DOCUMENT
    make (p: DOCUMENT)
        do
            parent := p
            parent.root.something
        end
end
```

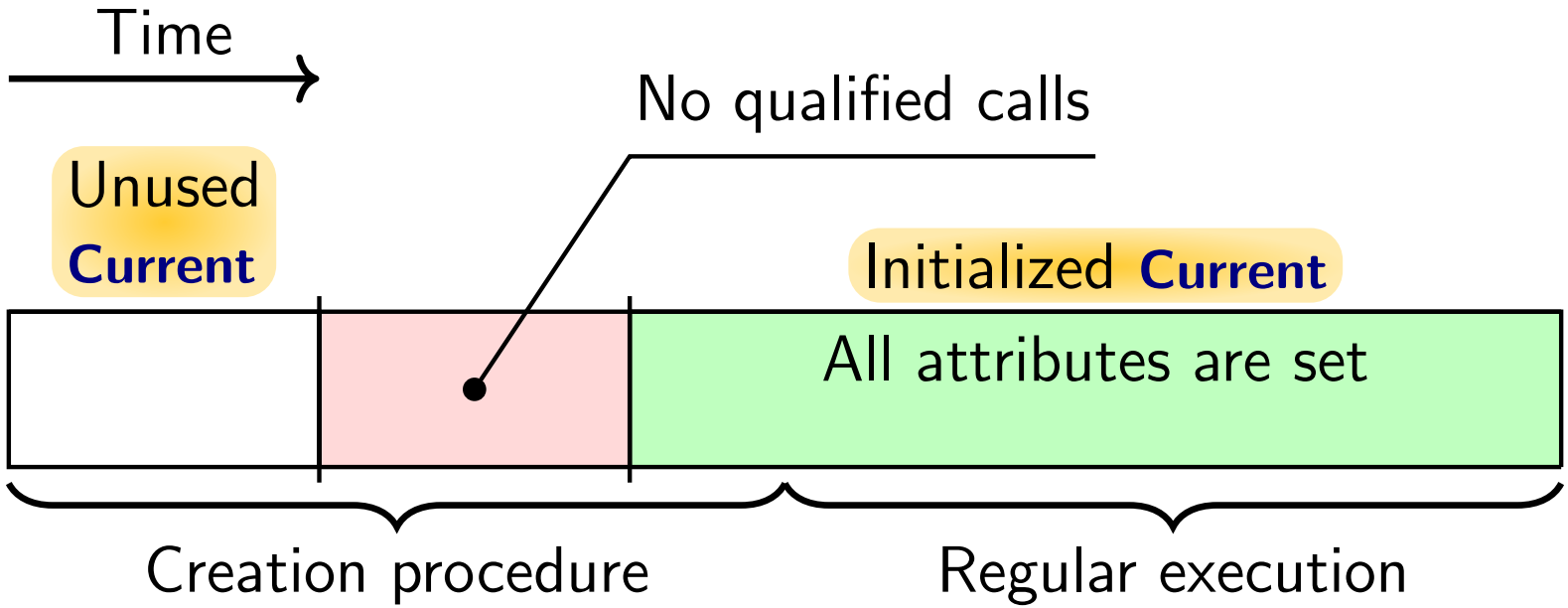
A: Qualified call

Solution



$$\frac{}{S,\ V \vdash \mathit{Current}\ \surd_c}\ \textsc{Current}$$

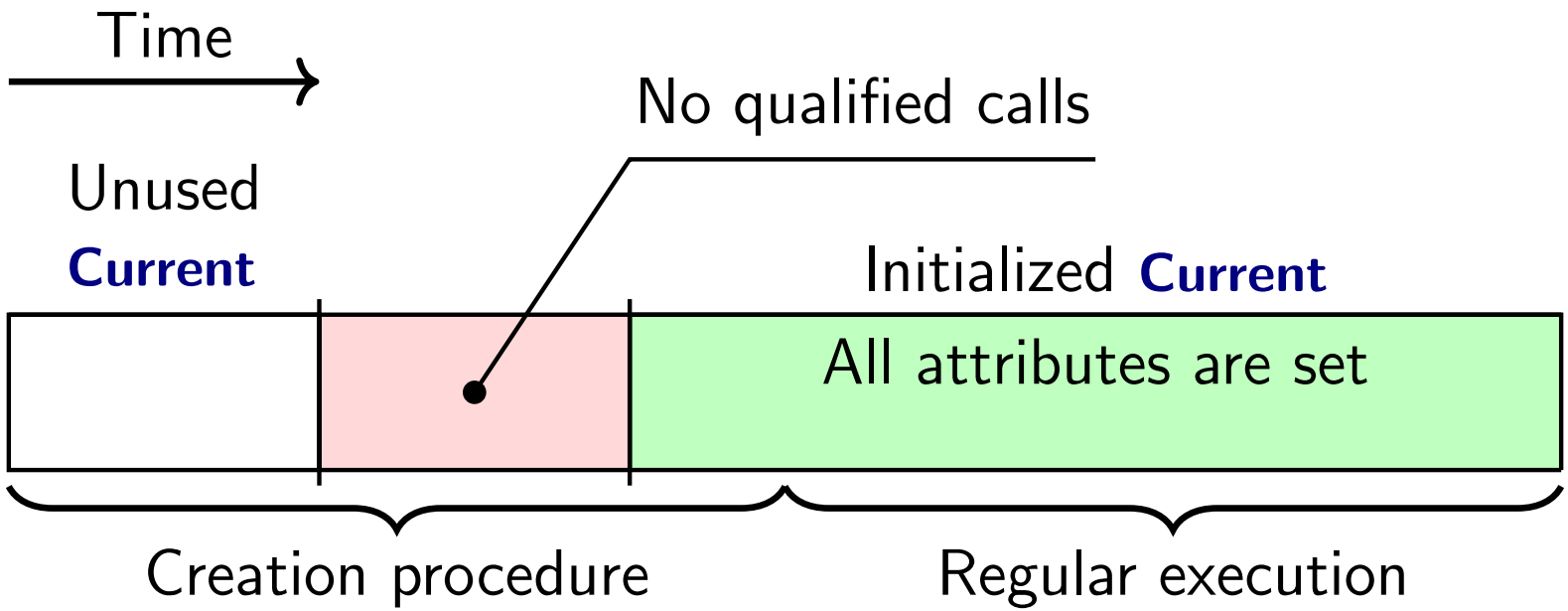
Solution



$$\frac{}{S,\ V \vdash \mathit{Current}\ \sqrt{_c}}\ \textsc{Current}$$

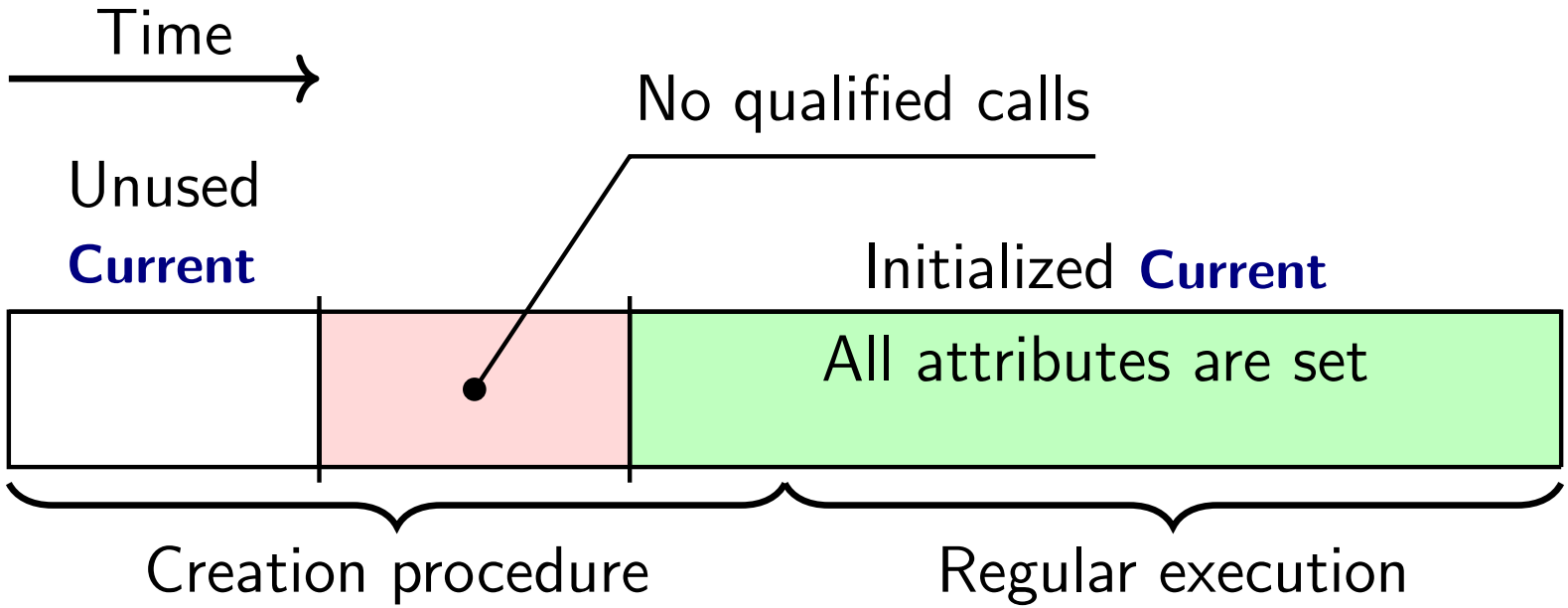
$$\frac{S,\ V \vdash e\ \sqrt{_c} \wedge S,\ V \gg e \vdash es\ [\sqrt{_c}] \wedge \mathit{safe}\ (e \cdot es)\ V}{S,\ V \vdash e\ .\ n\ (es)\ \sqrt{_c}}\ \textsc{Call}$$

Solution



Polymorphic call	✓
Callback	✓
Mediator	✓
Circular references	✓

Solution



- No additional annotations
- All existing code
- **1%** compilation time increase

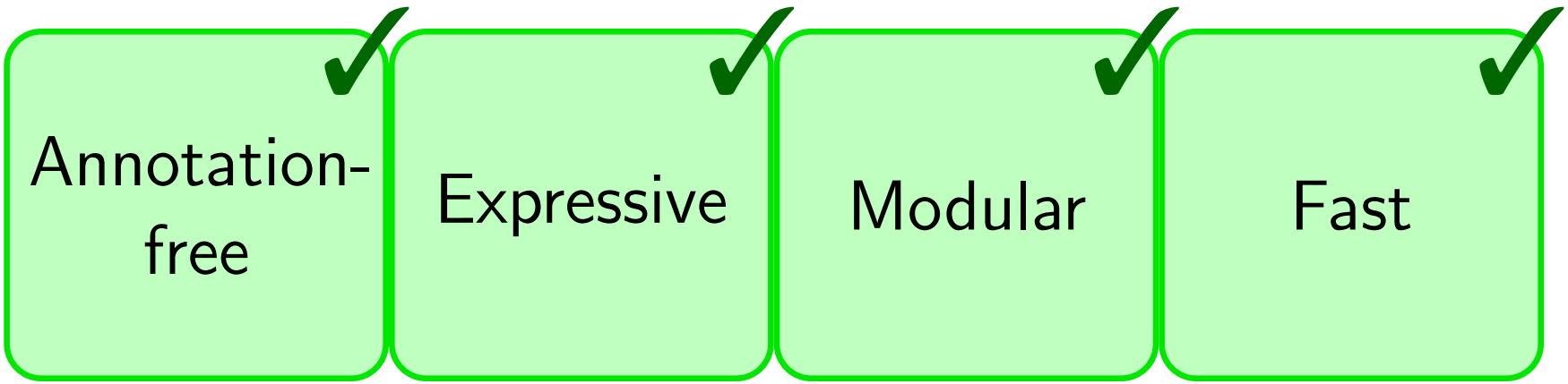
Comparison with existing approaches

Approach	Examples: polymorphic call	callback	mediator	circular references	Annotations
Raw types		✗	✗	✗	2+
Masked types					**many**
Free/committed types		?	✗		1+
Targeted expressions					0
Abstract interpretation					0
Practical void safety					0

Comparison with existing approaches

Approach	Examples: polymorphic call	callback	mediator	circular references	Annotations	Modularity
Raw types		✗	✗	✗	2+	class
Masked types					**many**	**?**
Free/committed types		?	✗		1+	class
Targeted expressions					0	**system**
Abstract interpretation					0	**system**
Practical void safety					0	library

Key features



expr.method (args);