Null safety benchmarks for object initialization

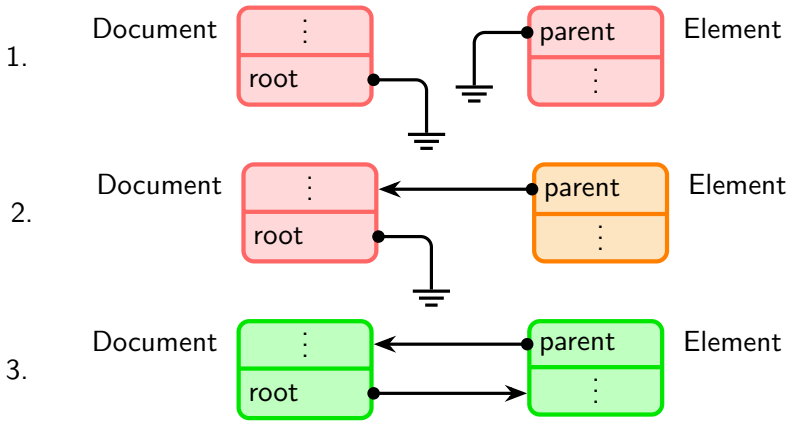
https://bitbucket.org/kwaxer/null-safety-benchmark/src/?at=2017-ispras

Alexander Kogtenkov

2017-11-30

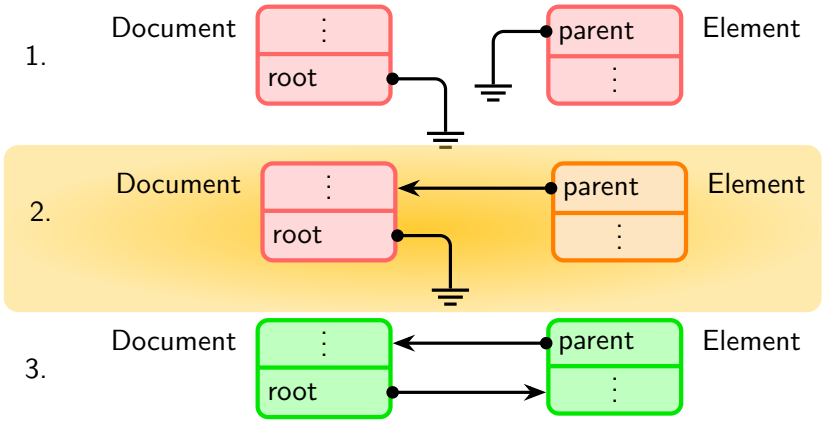
Example: circular references in XML document

```
class Document { var root: Element ... }
class Element { var parent: Document ... }
```

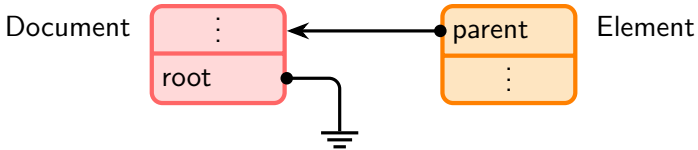


Example: circular references in XML document

```
class Document { var root: Element ... }
class Element { var parent: Document ... }
```



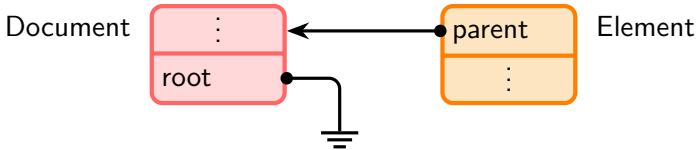
Q: What could go wrong?




```
class Document
{
    var root: Element

    constructor ()
    {
        root = Element (this)
    }
}
```

```
class Element
{
    var parent: Document

    constructor (p: Document)
    {
        parent = p
    }
}
```

Q: What could go wrong?




```
class Document
{
    var root: Element

    constructor ()
    {
        root = Element (this)
    }
}
```

```
class Element
{
    var parent: Document

    constructor (p: Document)
    {
        parent = p
        parent.root.parent
    }
}
```

Soundness vs. expressiveness

Expressive:

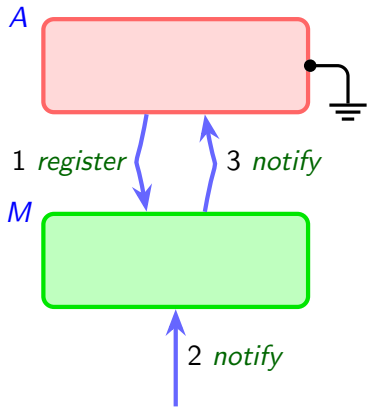
```
class Element
{
    var parent: Document

    constructor (p: Document)
    {
        parent = p
        [redacted] ✓
    }
}
```

Sound:

```
class Element
{
    var parent: Document

    constructor (p: Document)
    {
        parent = p
        parent.root.parent ✗
    }
}
```

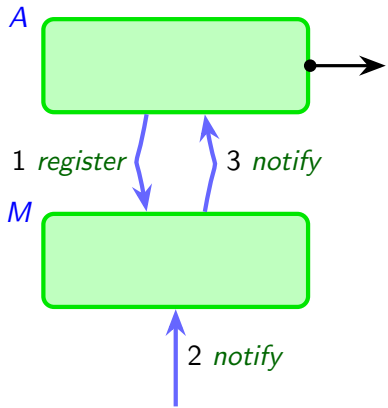
Example: mediator

```
class A {
    var x: X
    constructor (m: M)
    {

        m.register (this)
        x = ...
    }
    fun notify () { x.foo () }
}

class M { ...
    fun register (a: A) { list.add (a) }
    fun notify () { list.forEach { it.notify () } }
}
```



Example: mediator

```
class A {
    var x: X
    constructor (m: M)
    {
        x = ...
        m.register (this)

    }
    fun notify () { x.foo () }
}

class M { ...
    fun register (a: A) { list.add (a) }
    fun notify () { list.forEach { it.notify () } }
}
```



Soundness vs. expressiveness

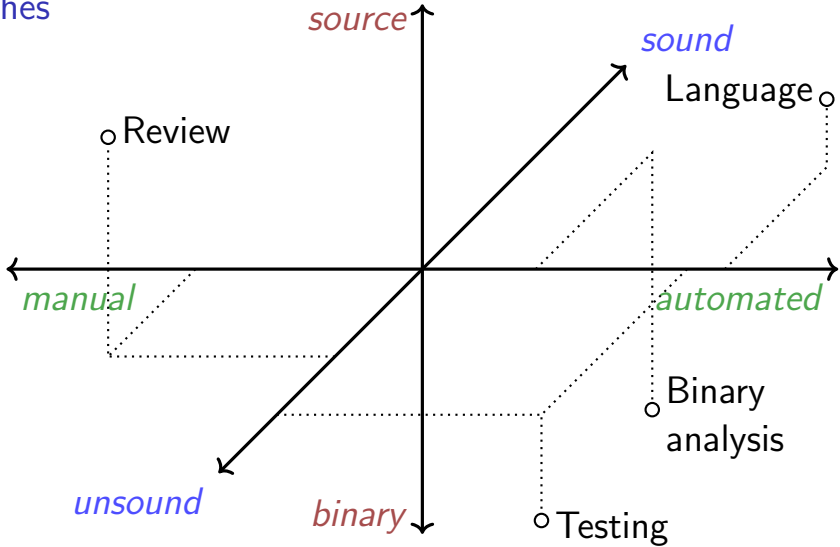
Expressive:

```
class A {
    var x: X
    constructor (m: M)
    {
        x = ... ✓
        m.register (this)

    }
    fun notify () { x.foo () }
}
```

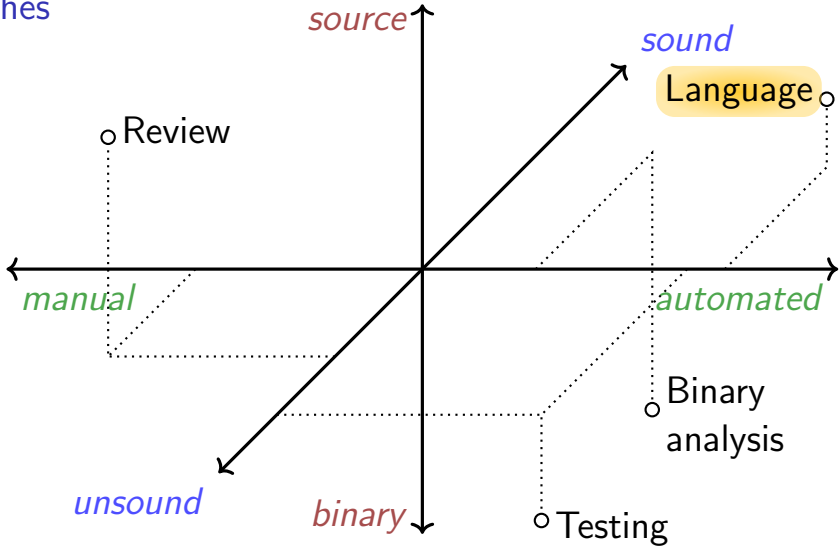
Sound:

```
class A {
    var x: X
    constructor (m: M)
    {

        m.register (this)
        x = ... ✗
    }
    fun notify () { x.foo () }
}
```

Approaches


source
sound
Language
Review
manual
automated
Binary
analysis
unsound
binary
Testing

Approaches


source
sound
Language
Review
manual
automated
Binary
analysis
unsound
binary
Testing

Roots of the object initialization problem

1. Non-atomic initialization

```
class A {
    var x: X
    constructor (m: M)
    {
        x = ... ✓
        m.register (this)
        x = ... ✗
    }
    fun notify () { x.foo () }
}

class M { ...
    fun register (a: A) { list.add (a) }
    fun notify () { list.forEach { it.notify () } }
}
```

Roots of the object initialization problem

1. Non-atomic initialization
2. Aliasing
   - Argument
   - Context (static, global, etc.)
   - Finalizer

```
class A {
    var x: X
    constructor (m: M)
    {
        x = ... ✓
        m.register (this)
        x = ... ✗
    }
    fun notify () { x.foo () }
}

class M { ...
    fun register (a: A) { list.add (a) }
    fun notify () { list.forEach { it.notify () } }
}
```

Roots of the object initialization problem

1. Non-atomic initialization
2. Aliasing
3. Uncontrollable control flow
   - Exception
   - Concurrency
   - Cooperative execution

```
class A {
    var x: X
    constructor (m: M)
    {
        x = ... ✓
        m.register (this)
        x = ... ✗
    }
    fun notify () { x.foo () }
}

class M { ...
    fun register (a: A) { list.add (a) }
    fun notify () { list.forEach { it.notify () } }
}
```

Roots of the object initialization problem

1. Non-atomic initialization
2. Aliasing
3. Uncontrollable control flow
4. Dereferencing

```
class A {
    var x: X
    constructor (m: M)
    {
        x = ... ✓
        m.register (this)
        x = ... ✗
    }
    fun notify () { x.foo () }
}

class M { ...
    fun register (a: A) { list.add (a) }
    fun notify () { list.forEach { it.notify () } }
}
```

Roots of the object initialization problem

1. Non-atomic initialization
2. Aliasing
3. Uncontrollable control flow
4. Dereferencing

```
class A {
    var x: X
    constructor (m: M)
    {
        x = ... ✓
        m.register (this)
        x = ... ✗
    }
    fun notify () { x.foo () }
}

class M { ...
    fun register (a: A) { list.add (a) }
    fun notify () { list.forEach { it.notify () } }
}
```

Results: expressiveness

Tool	Example						Score
	callback	cycle: self	cycle: mutual	uninitialized	reg.: argument	reg.: context	
Java *Checker Framework*	+	+	+	+	⊕	⊕	6*
EiffelStudio compiler	+	+	+	−	+	+	5
Kotlin compiler	+	+	+	+	+	+	6

* Expected: 4

Results: soundness

Tool	Example registration: argument	Example registration: fresh	Example registration: context	Example: reclamation	Example: transfer	Score
Java *Checker Framework*	⊖⊖	⊖⊖	⊖⊖	−	⊖	0*
EiffelStudio compiler	++	++	++	−	+	7
Kotlin compiler	−−	−−	−−	−	−	0

* Expected: 7

The next step

Today:

Theory		Practice
Small subset Formal proof	Programming language Tool	Real PL Compiler

The next step

Today:

Theory		Practice
Small subset Formal proof	Programming language Tool	Real PL Compiler

Next:

Formalized Real programming language Certified Compiler