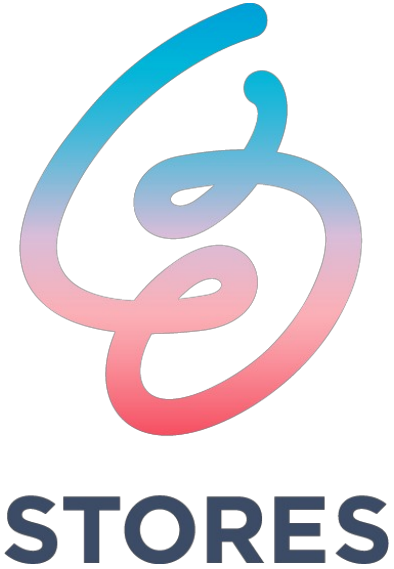
Writing Ruby Scripts with TypeProf

Yusuke Endoh (@mame)

RubyKaigi 2025



Partner with RubyStackNews

Independent Ruby & Rails publication for senior developers

Why RubyStackNews?

- Focused on Ruby and Ruby on Rails
- Long-form articles based on real conference talks
- Audience of senior developers and tech leads
- Readers from the US, Europe, and Asia

RubyStackNews turns conference talks and real-world experience into practical, production-focused technical articles.

Partnerships & Sponsorships

- Article sponsorships
- Inline placements inside articles
- Sidebar visibility

View partnership details

rubystacknews.com

Partnership info

PR: I wrote a book

- 「型システムのしくみ」
  "Type Systems Distilled with TypeScript"
  (A Japanese book)

- It explains how to write a type checker

  … for a subset of TypeScript

PR: I wrote a book

- Inspired by「型システム入門 (TAPL)」
  - "Types and Programming Languages"
    (The most well-known textbook of type systems)
- It explains:
  - Base type system
  - Subtyping
  - Recursive types
  - Generics

... in TypeScript

I am one of the translators of TAPL (Ja version)

PR: I wrote a book

- It uses TypeScript. Why?
  - ~~It sells better than Ruby~~
  - First-class functions make it convenient to explain the traditional type systems
- I want more contributors for Ruby types
  - Interested in Steep or Sorbet? Check it out!
- Available at the bookstore (the 2nd floor)
  - Book signing next break. Get one!

PR: IRB Treasure Hunt! (@ STORES booth)

https://ruby-quiz-2025.storesinc.tech/

by mame, hogelog, and ima1zumi

STORES

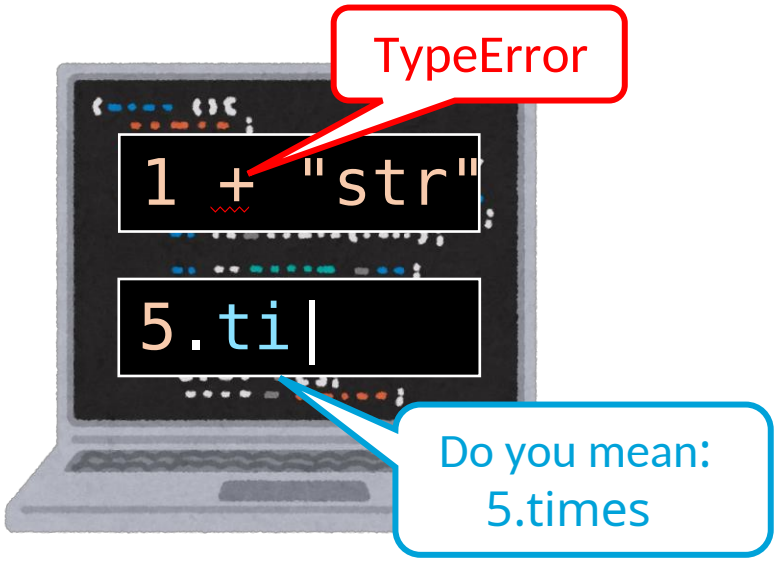
Today's talk

- **What is TypeProf?**
- How to use TypeProf effectively
- Ruby's constant is 
- Conclusion

What is TypeProf?

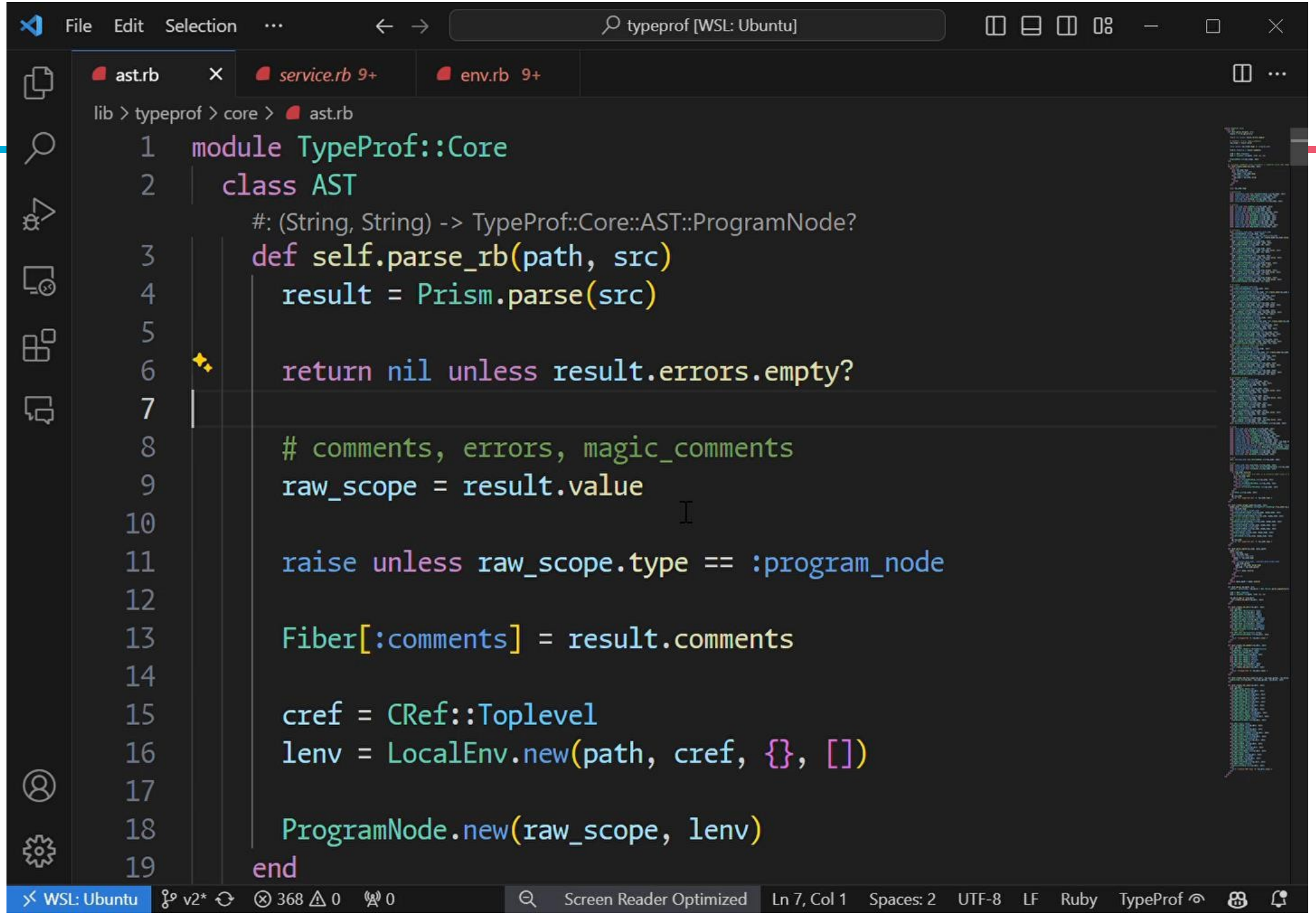
Ruby Editor support

- Error report, go to definition, completion, etc.
- With minimal type annotations!



Features

- Type inference, error/warning report
- Go to definition
- Completion
- Go to references
- Go to type references
- Automatic rename (method, constant)
- Inline RBS (→ rbs-inline)


File Edit Selection ···
typeprof [WSL: Ubuntu]
ast.rb
service.rb 9+
env.rb 9+
lib > typeprof > core > ast.rb
module TypeProf::Core
class AST
#: (String, String) -> TypeProf::Core::AST::ProgramNode?
def self.parse_rb(path, src)
result = Prism.parse(src)
return nil unless result.errors.empty?
# comments, errors, magic_comments
raw_scope = result.value
raise unless raw_scope.type == :program_node
Fiber[:comments] = result.comments
cref = CRef::Toplevel
lenv = LocalEnv.new(path, cref, {}, [])
ProgramNode.new(raw_scope, lenv)
end
WSL: Ubuntu
v2*
368 0
0
Screen Reader Optimized
Ln 7, Col 1
Spaces: 2
UTF-8
LF
Ruby
TypeProf

Progresses since last RubyKaigi

- Called for contributions
  - Got about 100+ PRs (Thanks all contributors!)
- Supported for Ruby's full syntax
- Created TypeProf.wasm (a demo in browser by ruby.wasm)
  - https://mame.github.io/typeprof.wasm/
- **Improved for practical use cases**
- **Fixed a bug of infinite-loop**

Today's main topics

Today's talk

- What is TypeProf?
- **How to use TypeProf (and recent improvements)**
- Ruby's constant is 
- Conclusion

How to use TypeProf for your projects

- Install VSCode "Ruby TypeProf" plugin
- Put typeprof.conf.json (or jsonc) in the top folder

```
{
  "typeprof_version": "experimental",
  "rbs_dir": "sig/",
}
```

- Reopen VSCode, and see if it works 🙏
  - If it doesn't work well, that's a good chance to contribute😌
- For details, check my slide deck for RubyKaigi 2024

New features

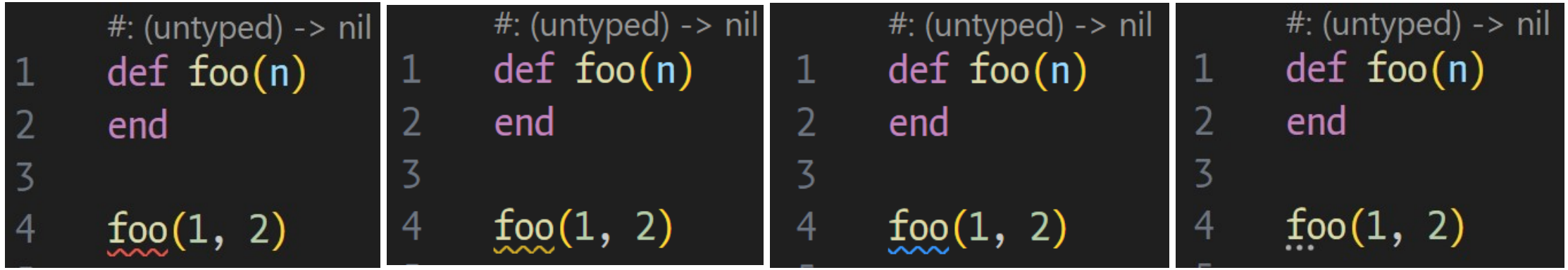
- diagnostic_severity and analysis_unit_dirs

```
{
  "typeprof_version": "experimental",
  "rbs_dir": "sig/",
  "diagnostic_severity": "info",
  "analysis_unit_dirs": [
    "lib/your_project/foo/",
    "lib/your_project/bar/",
  ]
}
```

New features

"diagnostic_severity": change error level

- TypeProf still reports many false positives
  - For a short-term solution, I provided a way to hide errors



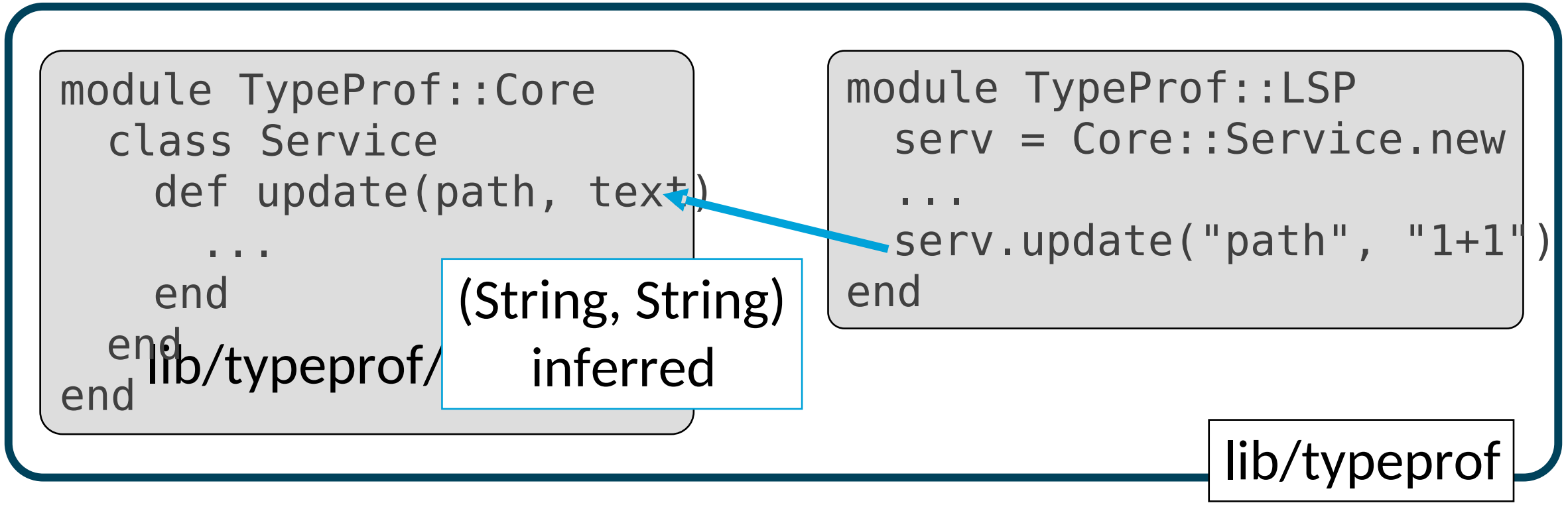
severity: "error" (default) "warning" "info" "hint"

... and "none" are available

"analysis_unit_dirs": separate analysis

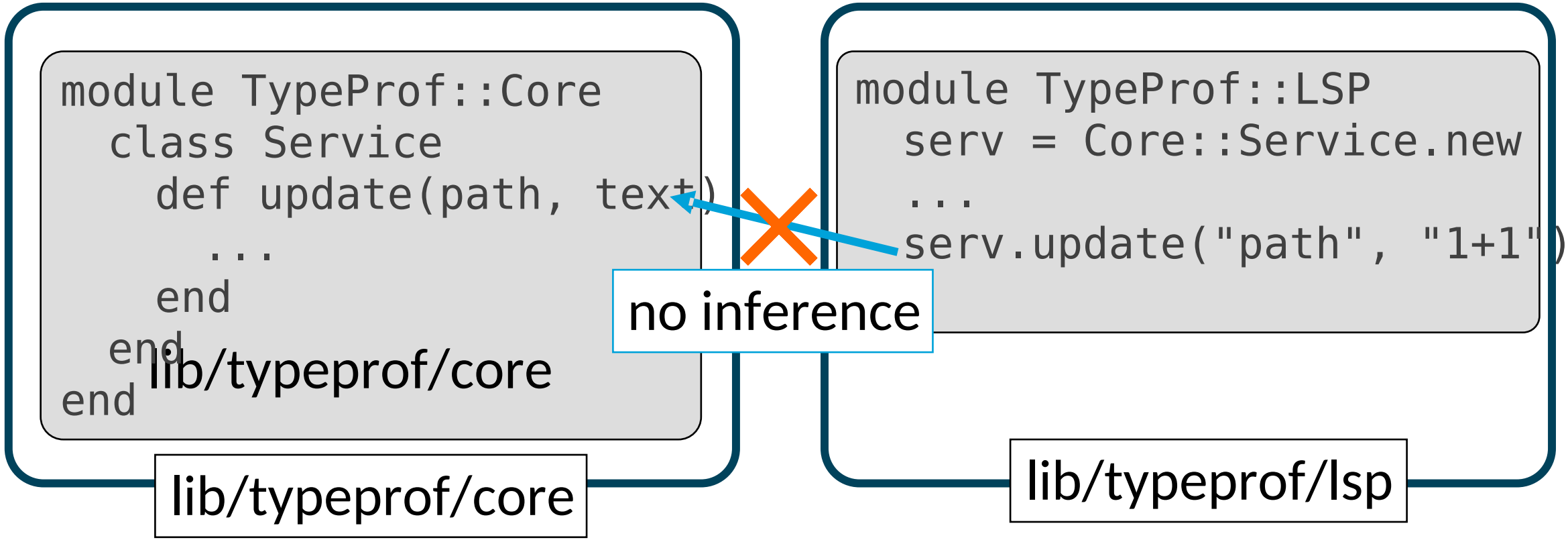
- You can specify directories to be analyzed together
  - For a large project, you need to separate analysis for each directory
  - API calls across file groups should be declared in RBS

TypeProf infers types by default



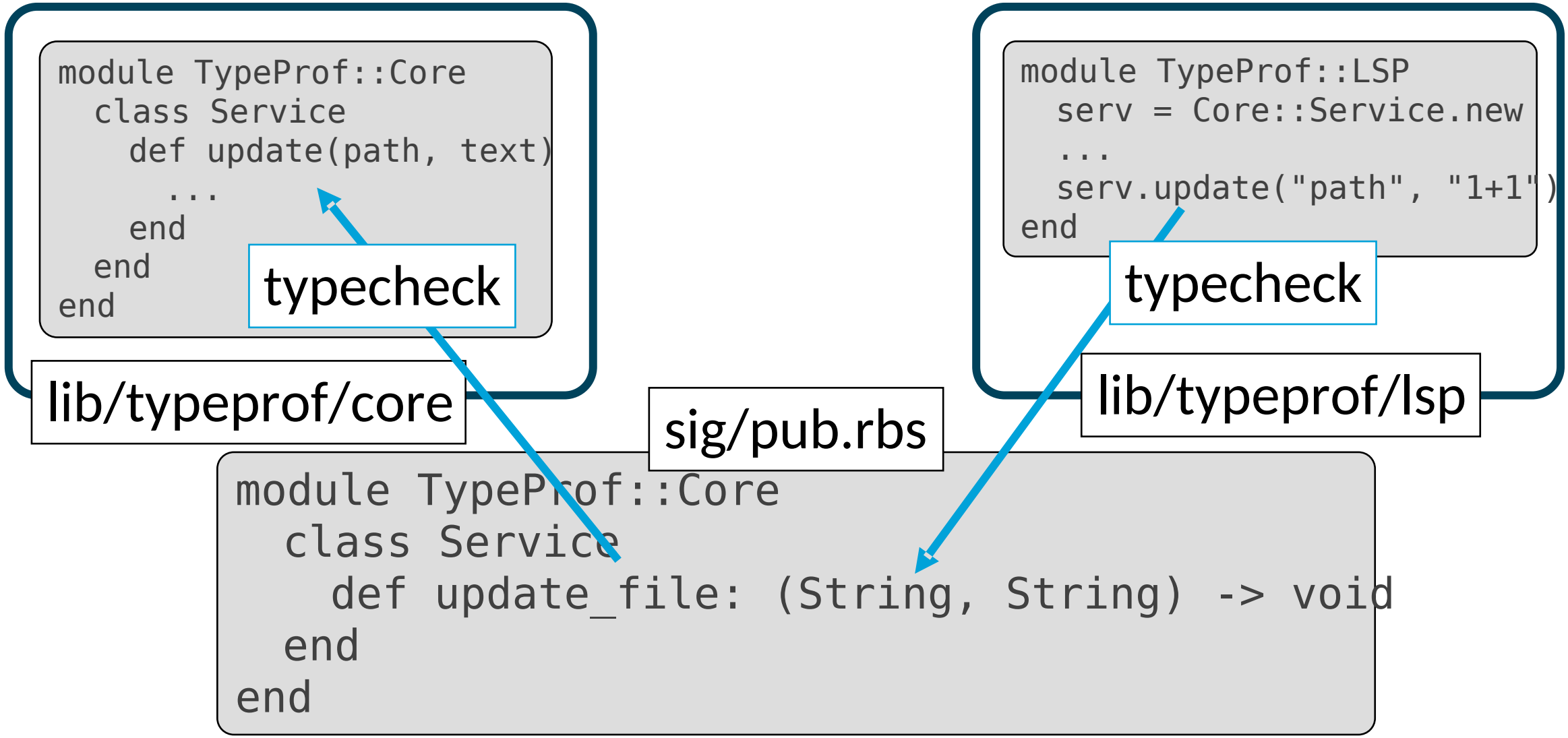
- … but as the project size grows, it will not scale

Separate analysis units



- Stop type inference between units

Write RBS between analysis units



Today's talk

- What is TypeProf?
- How to use TypeProf (and recent improvements)
- **Ruby's constant is** 
- Conclusion

Background: Ruby's constant is too complex

```
class P < Q
end

class A
  class B < Z
    class C < P
      include M

      Foo.new(...)
    end
  end
end
```

What could this `Foo` refer to?

- Ruby's constant resolution
  - `C::Foo`
    - The current context has the priority
  - `M::Foo`, `P::Foo`, `Q::Foo`
    - The inheritance has the next priority
  - `B::Foo`, `A::Foo`, `::Foo`
    - The scope has the last priority
  - (Note: Z::Foo is not searched)

Constant analysis requires whole programs

```
# myapp/main.rb
class MyApp
  String.new(...)
end
```

This `String` should be `::String`.

```
# myapp/string.rb
class MyApp
  class String
  end
end
```

No, that was `MyApp::String`

Constant analysis requires whole programs

```
# myapp/main.rb
class MyApp
  String.new(...)
end
```

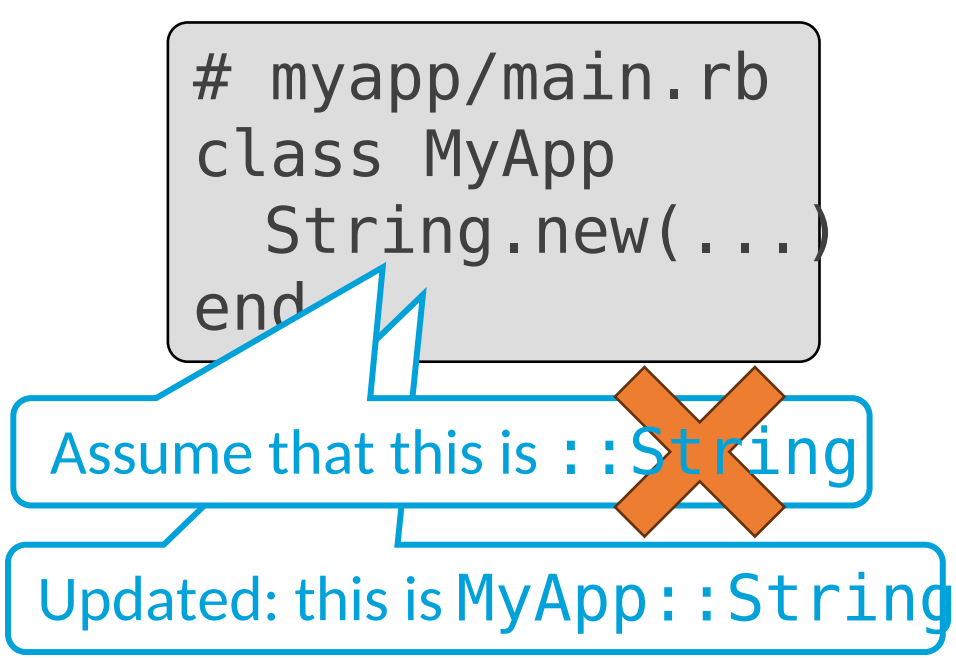
This `String` should be `::String`.

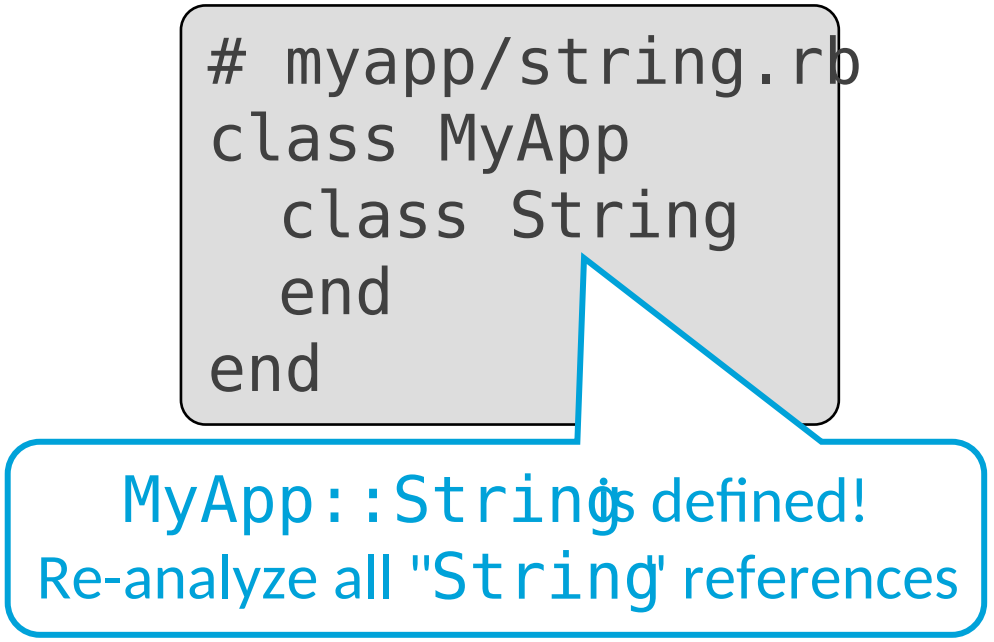
```
# myapp/string.rb
module M
  class String
  end
end
class MyApp
  include M
end
```

Also indirectly makes `MyApp::String` accessible

How to handle constants in TypeProf

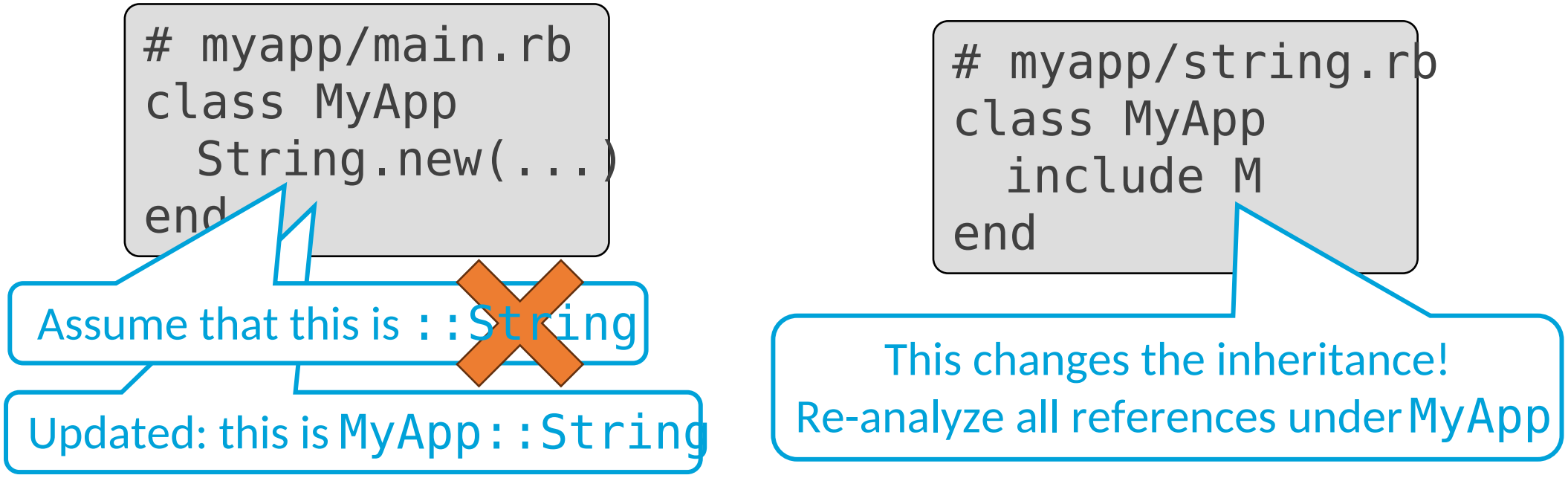
- Re-analyze existing constant resolutions:
  - **when a new constant is defined**
  - when the inheritance hierarchy is changed





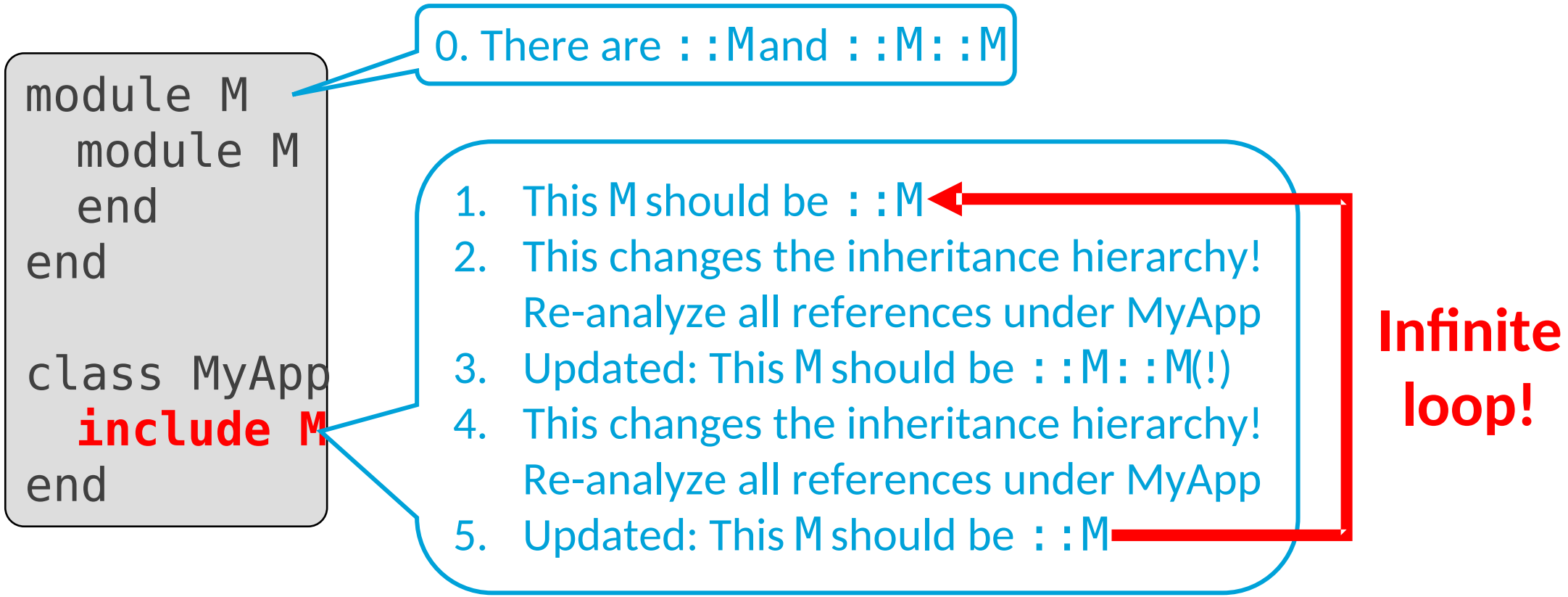
How to handle constants in TypeProf

- Re-analyze existing constant resolutions:
  - when a new constant is defined
  - **when the inheritance hierarchy is changed**



This mechanism caused an infinite-loop bug

- Found by @alpaca-tc

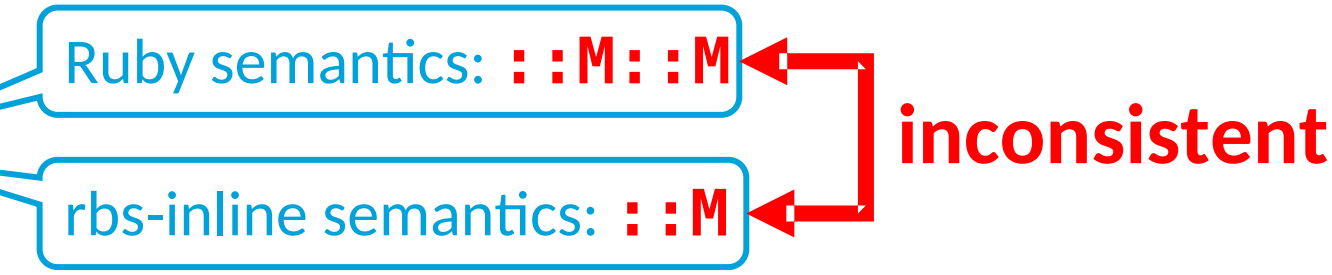


Similar problem was found in rbs-inline

- Constant could be inconsistent between ruby and rbs-inline

```
module M
  module M
  end
end

class MyApp
  include M
  include M
end
```



Solution: Give up the inheritance search

```
class P < Q
end

class A
  class B
    class C < P
      include M

      include Foo
    end
  end
end
```

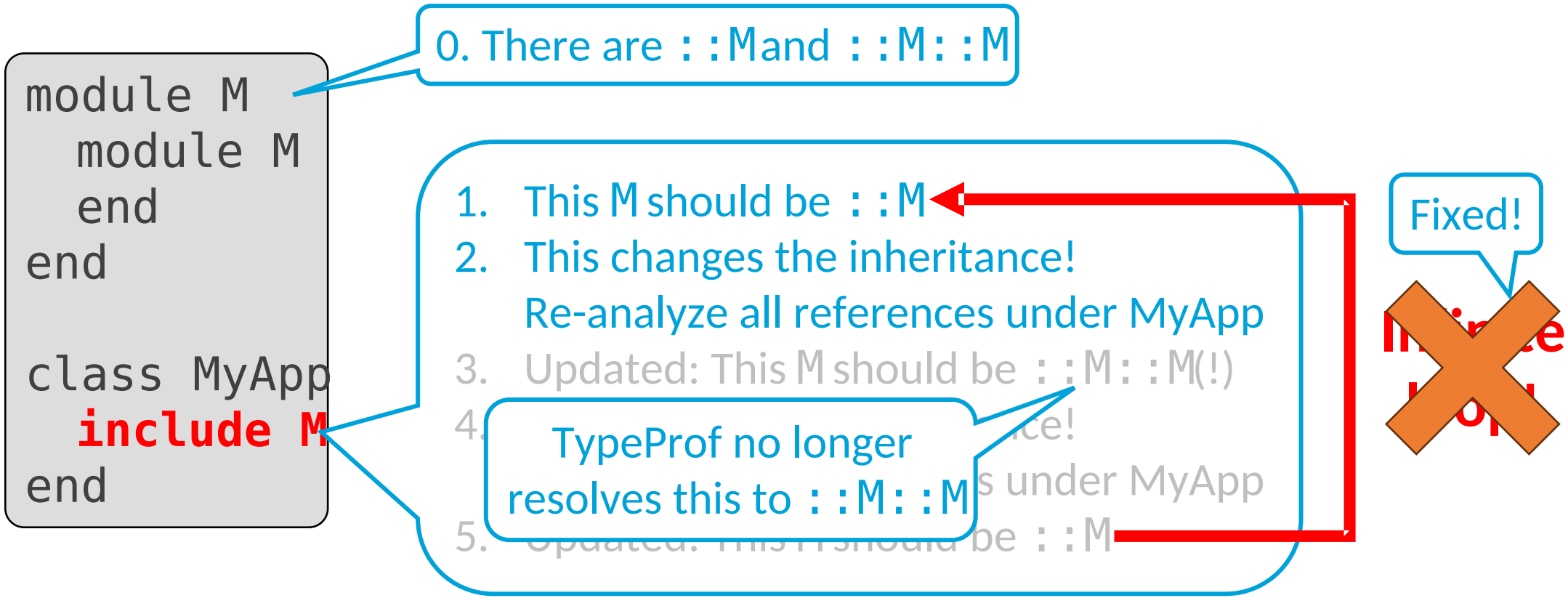
What could this Foo refer to?

- … for the argument of `include`
- Ruby's constant resolution
  - `C::Foo`
    - The current context
  - `M::Foo`, `P::Foo`, `Q::Foo`
    - The inheritance has the next priority
  - `B::Foo`, `A::Foo`, `::Foo`
    - The scope has the last priority

TypeProf no longer searches for inheritance

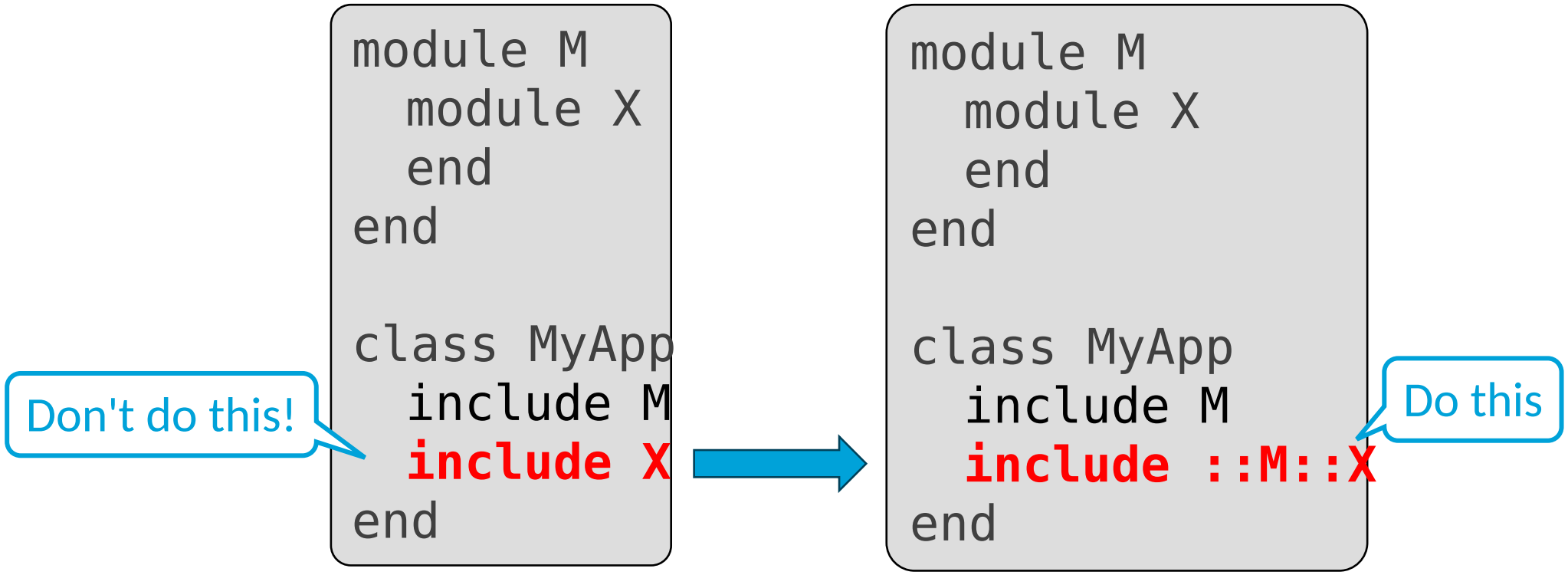
This mechanism caused an infinite-loop bug

- Found by @alpaca-tc



Recap: Rewrite your Ruby Code

- … if you want to use TypeProf or rbs-inline (or Sorbet)
  - Do not depend on the inheritance on constants



Today's talk

- What is TypeProf?
- How to use TypeProf (and recent improvements)
- Ruby's constant is 
- **Conclusion**

Conclusion

- TypeProf is getting production-ready (hopefully)
- Future work
  - Experiment with other than TypeProf itself
  - Still need many improvements
  - Contribution is truly welcome!
- Come meet me at the venue bookstore→