Even Lighter Than Lightweight

Augmenting Type Inference with Primitive Heuristics and AI-support

Larisa Safina, Jan Blizničenko

IWST 25 – Gdańsk, 2025

Type Inference? In Dynamic Languages? Why?

Improves correctness, maintainability, performance

Hard in dynamic languages like Smalltalk

- Polymorphism
- Trade-off: precision vs. speed
- Computational resources

And no one did it before?

A bit of history...

Impressive timeline (from 1981 by Suzuki)

From less to more formal

Compiler-oriented vs IDE-oriented

Pluggable Types

Gradual Typing

Heuristics

…

What about the present

- IDE: RoelTyper
- IDE: JInferer
- TypeInfoTools
- VM-level type feedback
- …

However, IDE-integrated type inference is not widely used by Pharo developers on a daily basis

Why this insufficient adoption?

- Problems of early solutions
- Human factor
  - end of funding
  - lost of interest
  - phd/postdoc ends
- Not user-friendly

So what do we want ?

- Balance precision and speed
- Make inference interactive and IDE-friendly

And how ?

Step 1: Lightweight heuristics to pre-calculate types and speed up stronger tools

Simple, fast, surprisingly effective

Going back to heuristics

Heuristics on method titles (127k methods)

- Implicit self returns (~57k methods)
- Type by name (~7,8k methods)
- Single return methods (~10k methods)
- Collection-specific methods
- Test methods (~22,5k methods)

Going back to heuristics

Heuristics on method titles

- Type by name (~7,8k methods)
  - Boolean (isEmpty, hasElements, includes, ..)
  - String (asString, …)
  - Numerical (size, …)

Going back to heuristics

Heuristics on method titles

- Single return methods (~10k methods)
  - AST LiteralNode as String, Number (8745 methods)
  - class reference (982 methods)
  - class new (394 methods)
  - self
  - self new (34 methods)
  - nil (278 methods)

Going back to heuristics

Heuristics on method titles

- Collection-specific methods
  - Numeric e.g indexOf
  - String e.g. concatenation
  - Boolean e.g. anySatisfy:
  - Transformation e.g asCollection
  - Collection-preserving e.g addAll:, copy

Going back to heuristics

Heuristics on method titles (127k methods)

- Implicit self returns (~57k methods)
- Type by name (~7,8k methods)
- Single return methods (~10k methods)
- Collection-specific methods
- Test methods (~22,5k methods)

Evaluation

Inference takes ~1 minute on Mac M2 with 24 GB of RAM

In Pharo 13 image heuristics cover 97,3k methods of 127k

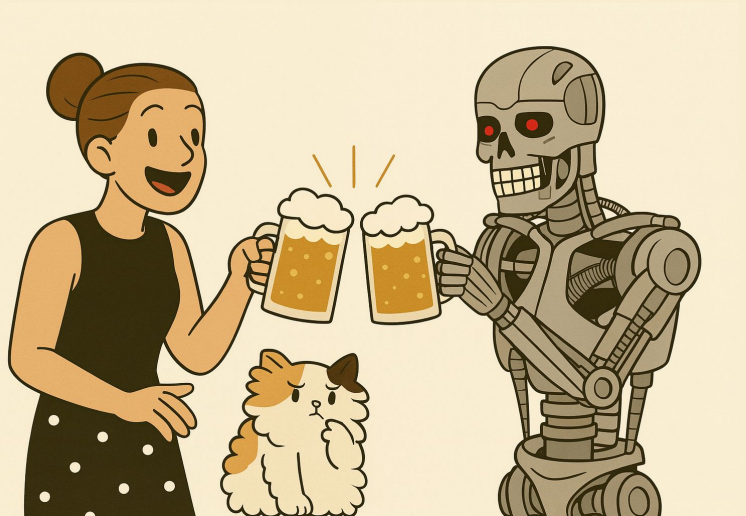
Validation

- Sanity Check
- Comparison with TypeInfoTools

AI-Assisted Inference (Experimental)

- Queries to OpenAI/Mistral for type prediction
- Promising, but slow and sometimes wrong
- Can complement heuristics

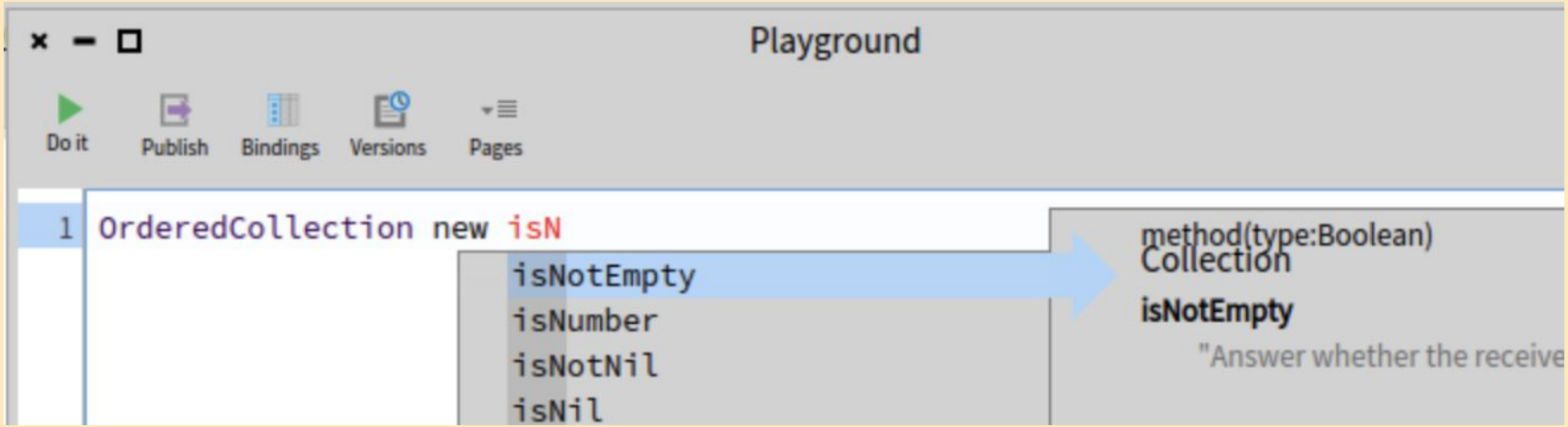
- Need to learn lessons from Python



IDE Integration

- Type results shown on demand
- CSV-based or loaded into Pharo (negligible overhead)
- AI-specific settings

Type results shown on demand



What's Next?

Combine everything! (Heuristics, other TI tools, AI, CI, ADHD, …)

- Interactive Type Refinement
- Create a Smalltalk “typeshed”
- Allow devs to refine suggestions
- Improve AI part
- …

Thanks / Q&A

github.com/lsafina/typeMe