UNIVERSITÄT BERN

TypePlug -- Practical, Pluggable Types

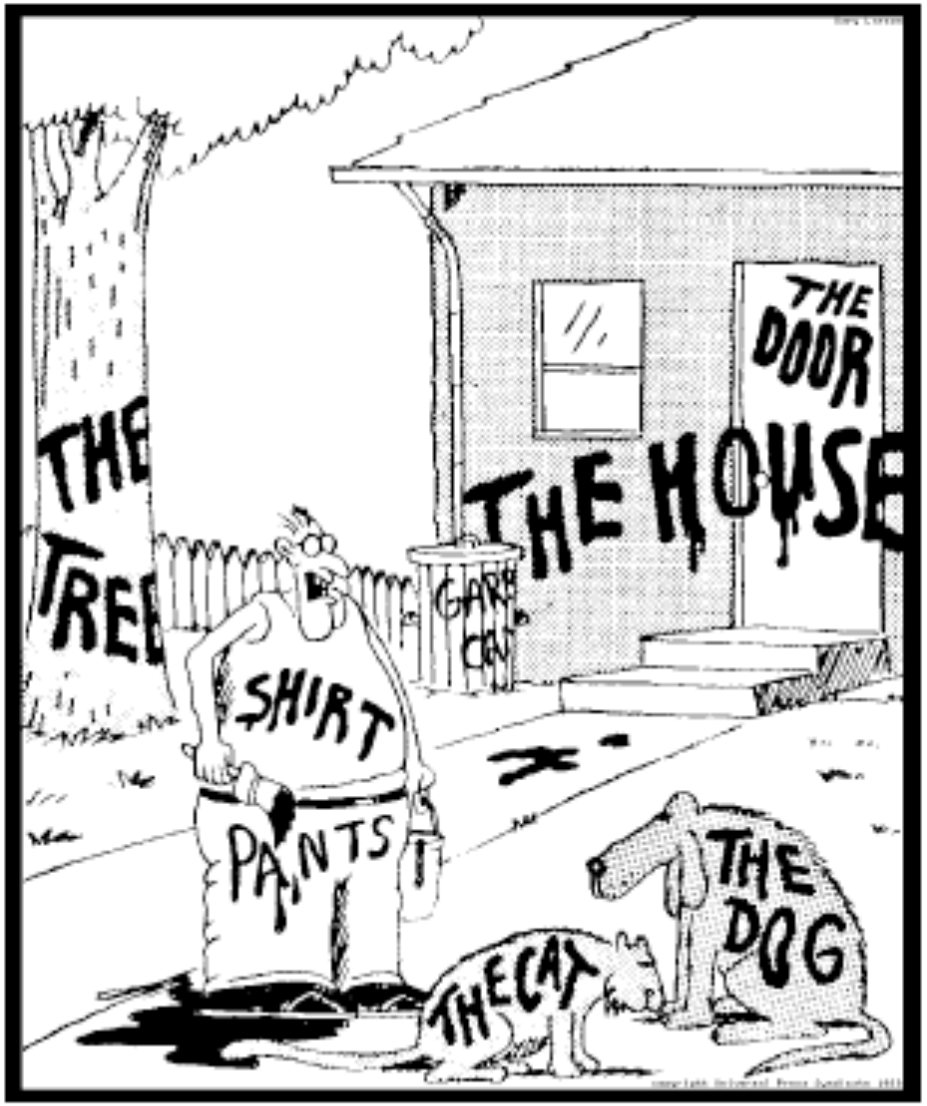
Marcus Denker
with:
Nik Haldiman
Oscar Nierstrasz

University of Bern

Types?

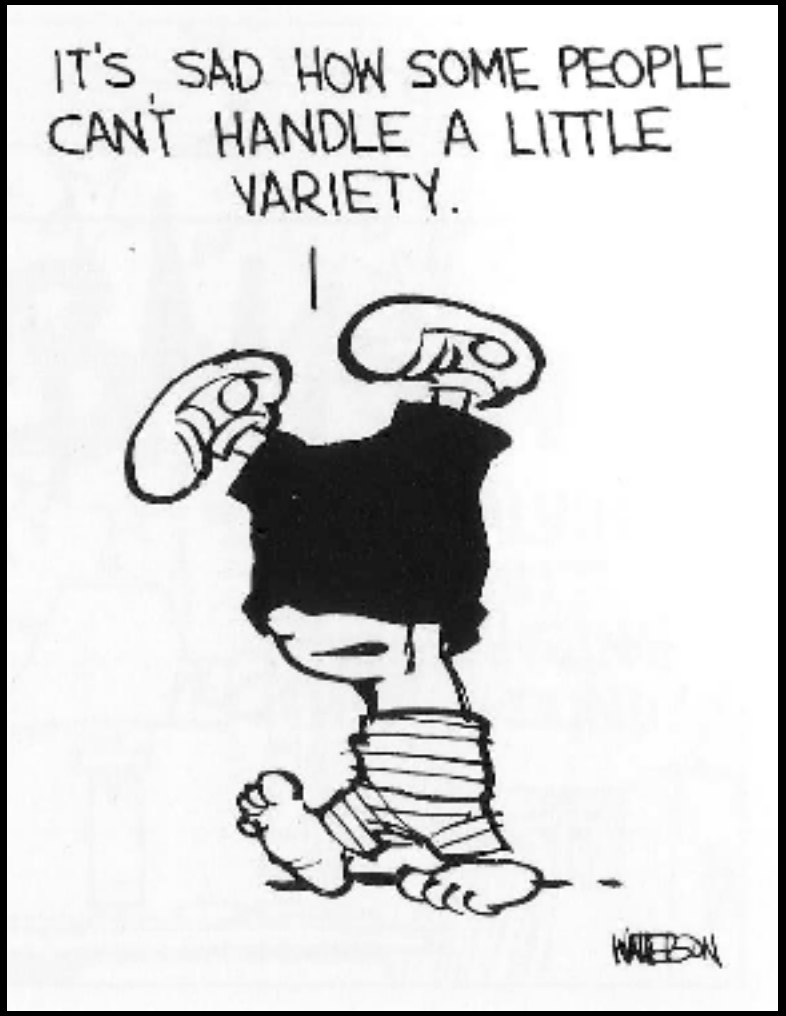
Caveat....

I am not a Type Person



"Now! *That* should clear up
a few things around here!"

Static types are Evil



Static types are Evil?

Static is Evil!

The Future....

...change

...evolution

...dynamic

...biological

Static typing is Good!

- Programs with failures are rejected
  - Reduces errors detected at runtime
- Documentation
- Minor inconvenience, major payoff

Static typing is Evil!

- Exactly all cool programs are rejected
  - Reflection?!
- Inconvenience is not at all “minor”
  - Typed programs hard to change + evolve
- Only the most trivial errors are detected
  - We would have found those anyway before deployment

Cakes by Darcy

Is it possible to have one's cake and eat it, too?

History: Strongtalk

- Anymorphic. Startup (ca. 1996)
  - Self team, from Sun
  - Smalltalk VM
- Smalltalk with a Type System
- Observations:
  - Types not needed for performance
  - Optional Types are nice (Documentation!)
    - *Can be introduced later when the system is stable*

Problem of Mandatory Types

- Types constrain the expressiveness of a Language
- Types make systems more brittle
  - Security and Performance
  - If types fail, behavior is undefined
- But Type-Systems are proven to be correct!?
  - Real world is too complex to formalize
  - Implementation will have bugs

Pluggable Types

- Optional: do not change the semantics
- Pluggable: many different ones
  - Especially exotic type-systems
- “Type-Systems as Tools”

Gilad Bracha, OOPSLA 04: Pluggable Type-Systems

Pluggable Types: Language

- Optional types do not constrain the expressiveness
  - We can ignore the type system if we want
  - (or turn it off completely)
- New language models can be realized faster
- Inventing the Future is hard if it needs to be type-save
  - Example: NewSpeak

Pluggable Types: Pluggability

- There is **a lot** of very interesting research on Types
- It is very hard to get it into the hands of Programmers
  - Sun will not change Java for you!
  - (even though you suffered with java for years for your research)
- Pluggable type-systems free type research from language adoption!

Pluggable Types: Types as Tools

Type Checker

We are free to explore the unthinkable in our Room

(Research!)

Type Inference?

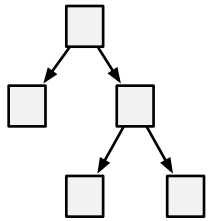
- Isn't Type Inference enough?
- Type Inference is cool. But it's a Type-system
- No Type Annotation != No Type System

Pluggable Types are very likely to use Inference

Pluggable Types at SCG

- Research about Software Evolution
  - Reflection to support dynamic change
- We like to use dynamically typed systems
  - Smalltalk / Squeak
- Thinking is not constrained by Types
  - Very important!

Methods and Reflection

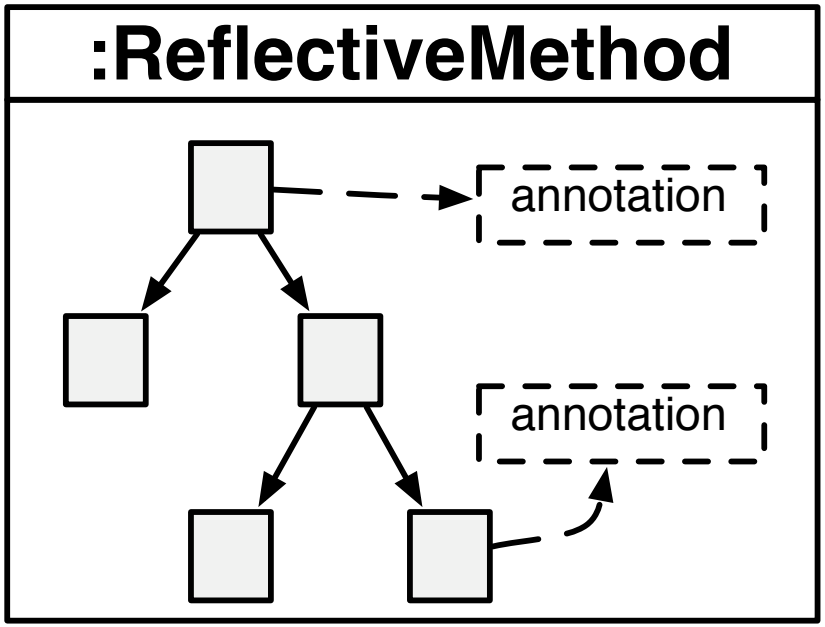


- Method are Objects
  - e.g in Smalltalk
- No high-level model for sub-method elements
  - Message sends
  - Assignments
  - Variable access
- Structural reflection stops at the granularity of methods

Sub-Method Reflection

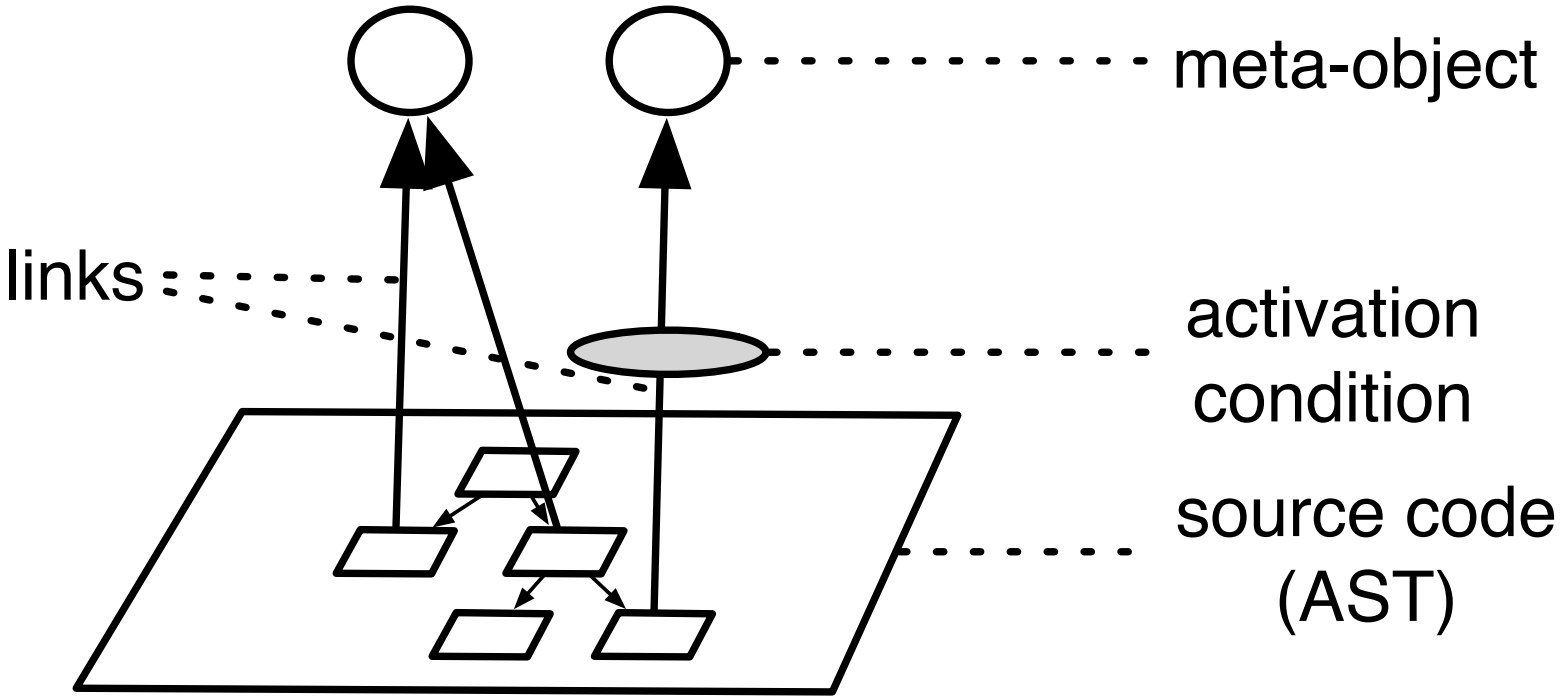
- Many tools work on sub method level
  - Profiler, Refactoring Tool, Debugger, Type Checker
- Communication between tools needed
  - example: Code coverage
- All tools use different representations
  - Tools are harder to build
  - Communication not possible

Sub-Method Reflection



- Sub-method Structure (AST)
- Annotations
  - Source visible
  - non-visible
- Causally connected

Sub-Method Reflection: Behavior



Sub-Method Reflection: Annotations

- Source visible annotations

```
(9 raisedTo: 10000) <:evaluateAtCompiletime:>
```

- Every node can be annotated
- Semantics: Compiler Plugins

- Type Annotations?

TypePlug

- Pluggable types for Squeak
- Based on sub-method reflection framework
- Case-Studies:
  - Non-Nil Types
  - Class Based Types
  - Confined Types

Master Thesis:
Nik Haldiman

The Problem

- Large, untyped code-base
- Overhead for using pluggable types is high
  - Existing code needs to be annotated with type information

Example: Non-Nil Type-System

- Declare variables to never be nil

```
Object subclass: #Line
   typedInstanceVariables: 'startPoint endPoint <:nonNil:>'
   typedClassVariables: ''
   poolDictionaries: ''
   category: 'Demo'
```

Non-Nil Type-System


```
moveHorizontally: anInteger

   startPoint := self movePoint: startPoint
                      horizontally: anInteger.

   endPoint:=self movePoint: endPoint
                  horizontally: anInteger
```

Non-Nil Type-System


```
moveHorizontally: anInteger

   startPoint := self movePoint: startPoint
                      horizontally: anInteger.

   endPoint:=self movePoint: endPoint
                  horizontally: anInteger <- type 'TopType' of
   expression is not compatible with type 'nonNil' of variable
   'endPoint'.
```

Non-Nil Type-System

UNIVERSITÄT BERN

```
movePoint: aPoint horizontally: anInteger

        ↑ (aPoint addX: anInteger y: 0) <:nonNil :>
```

The Problem (repeat)

- Large, untyped code-base
- Overhead for using pluggable types is high
  - Existing code needs to be annotated with type information

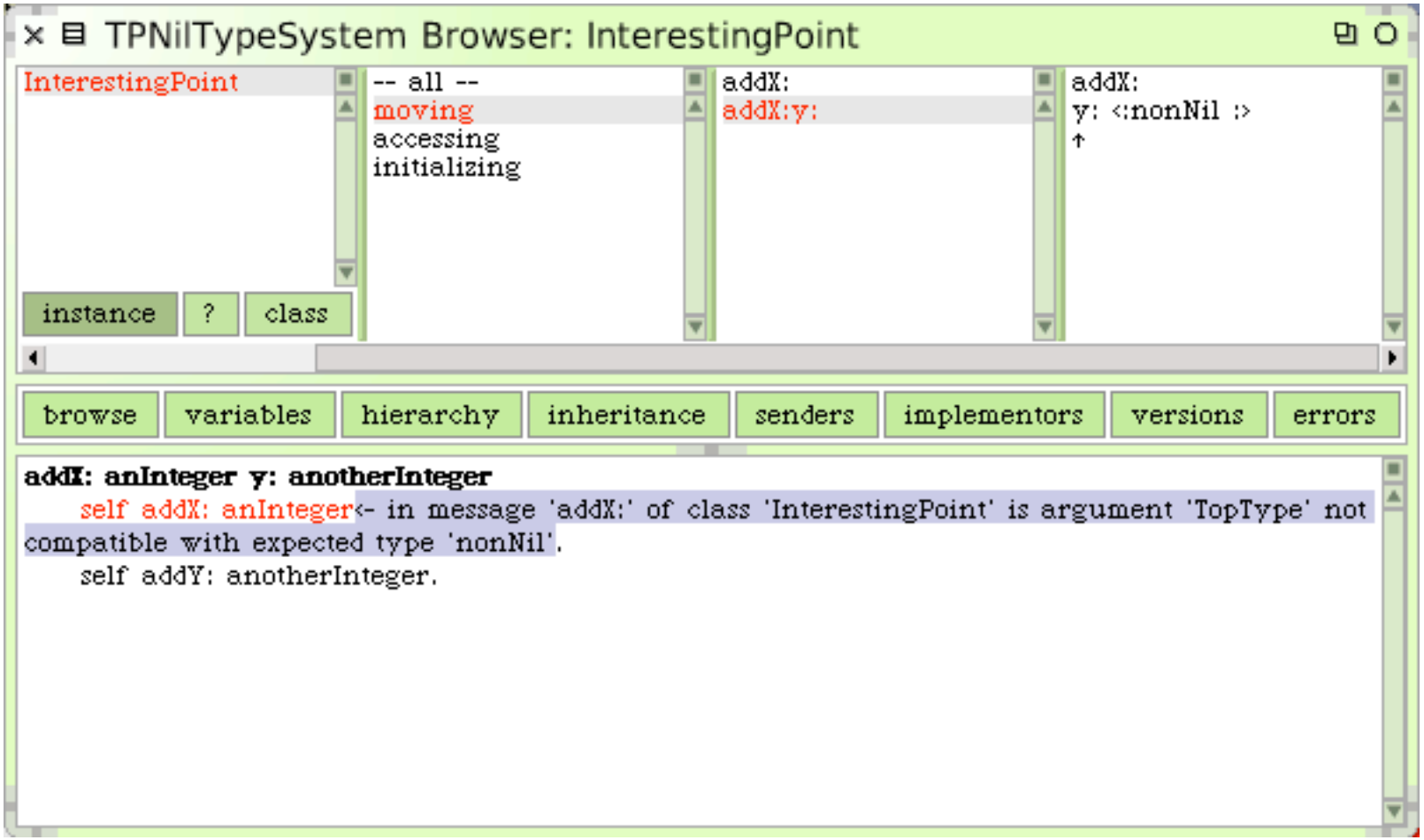
Solution

- Only type-check annotated code
- Use type-inference to infer types of non-annotated code
- Explicit type-casts
- Allow external annotations for foreign code

External Type Annotations

- We need to annotate existing code
  - Especially libraries and frameworks
  - Example: Object>>#hash is <: nonNil :>
- We do not want to change the program code!
- Solution: External Type Annotations
  - Added and modified in the TypesBrowser
  - Do not change the source
  - External representation: Type Packages

Browser



Future Work

- Improve Type-Inference
  - Better algorithms
  - Explore heuristical type inference
- Type Checking and Reflection
  - Use pluggable types to check reflective change

Conclusion

- Pluggable Types
  - All positive effects of static types
  - Avoid the problems
- TypePlug: Pragmatic framework for Pluggable Types
  - Example in the context of Smalltalk/Squeak

Conclusion

- Pluggable Types
  - All positive effects of static types
  - Avoid the problems
- TypePlug: Pragmatic framework for Pluggable Types
  - Example in the context of Smalltalk/Squeak

Questions?