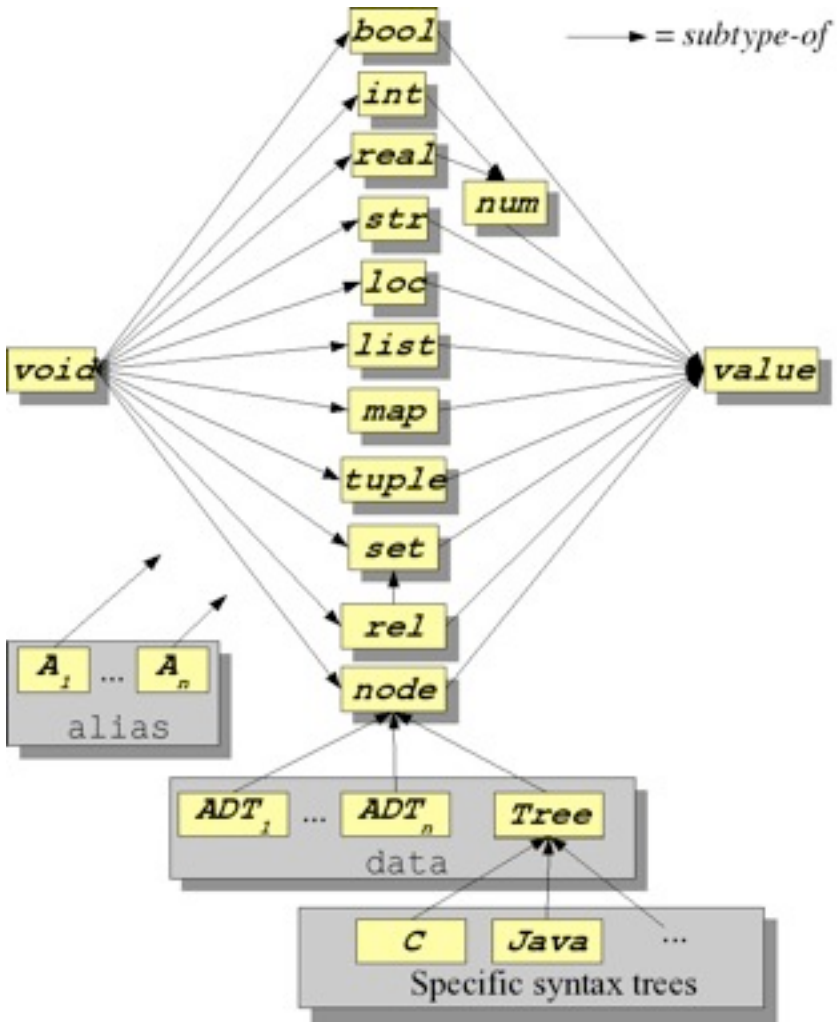
Type Checking Rascal in Rascal

Mark Hills

http://www.rascal-mpl.org

The Rascal Type System



Why Have a Type Checker?

- Rascal is defined as a statically typed language
- Helps catch many simple errors that are easy to make and fix
- Provides useful feedback to users (error markers, doc links and hovers)
- Performance optimization opportunities

So, What Should It Do?

- Find name definitions and uses (linking the two)
- Assign types to all names and expressions
- Infer types for names with no explicit declaration
- Determine which constructor and function overloads are used (this could be a set!)
- Mark locations of errors and warnings

Why Isn't This Done Yet?

- First attempt: bottom-up visit of tree
- Second attempt: constraint-based solution
- Current solution: abstract evaluation

Challenges

- Subtyping
- Local Inference
- Some features not fully statically type-able
  - Field access/projection/update
  - Reducers
- Imports, nested functions, desire for sanity in introducing variables leads to tricky scoping rules

Type Checking a Rascal File

- Extract signatures for imports
- Bring public signature items into top-level scope (for extends: bring all items in instead)
- Recursively, statically evaluate program: values are types
- Book-keeping: assign derived types into a map from locations to type; maintain a map from name declarations to name uses; both used by Eclipse to provide documentation in the IDE