Rapid Language Prototyping using Spoofax

Aron Zwaan

June 5, 2023

Delft University of Technology

About Me

- Aron Zwaan
- PhD Candidate @ TU Delft – Programming Languages Group
- Static Semantics (Type Systems)
- Developing *Statix* (DSL for Type System Specificiation)

This Lecture

1. Background on Languages and Compilers (brief)
2. Language Prototyping with Spoofax
3. Demo: Implementing Simple Types (LN. ch. 5)
4. Demo: Adding Security Labels (LN. ch. 6)

Compilers and Languages

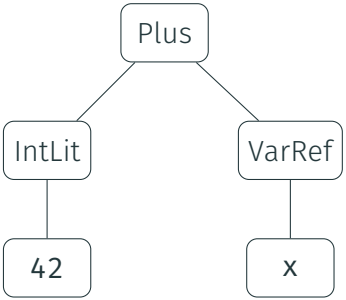
Compilers

- Humans write Programs using some Language
- Computers cannot execute these directly
- Compilers: Human-Understandable Code $\Rightarrow$ Executable Bytecode

Abstract Syntax Trees

- Tree-Structured Way of Representing Programs
- Discard irrelevant details
  - Keywords
  - Fences
  - Operators
  - Comments
  - Layout
  - …

```
42 + x
```



Designing Programming Languages

- Many Programming Languages exist, but why?
- Offer right Abstractions for Particular Domain

	OS	Web App	DB Query
C	+++	~	-
Java	--	++	~
SQL	n.a.	n.a.	+++

Designing Programming Languages

- Many Programming Languages exist, but why?
- Offer right Abstractions for Particular Domain

	OS	Web App	DB Query
C	+++	~	-
Java	--	++	~
SQL	n.a.	n.a.	+++

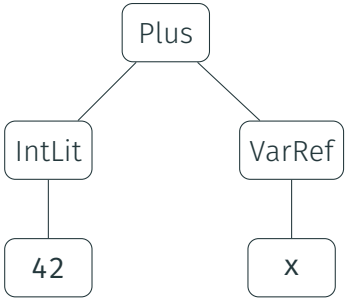
How to Design proper Abstractions?

{S} spoofax

- Language Workbench: Collection of Tools for implementing Compilers
- Dedicated *Meta-Language* for each Component of a Compiler
- *Declarative*: Specify only *what* You Want
- *Executable*: Implementations Generated from Specifications

Abstract Syntax Trees in Spoofax

- ATerm (Annotated Term) Format




```
Plus(
  IntLit("42")
, VarRef("x")
)
```

Parsing in Spoofax

- SDF3 (Syntax Definition Formalism 3)
- Specify Grammar
- Generate:
  - Parser
  - Syntax Highlighting
  - Pretty-Printing
  - ...

$$\langle exp \rangle ::= \langle int \rangle \mid \langle exp \rangle \text{ '+' } \langle exp \rangle \mid \ldots$$

$$\langle int \rangle ::= \ldots$$

```
context-free sorts
  Exp

context-free syntax
  Exp.IntLit = INT
  Exp.Plus   = <<Exp> + <Exp>>
```

Type-Checking in Spoofax

- Statix
- Specify Type System Rules
  - Scope Graphs for Name Binding
- Generate:
  - Executable Type Checker
  - Editor Services
  - Code Completion
  - Refactorings
  - …

$$\frac{}{\Gamma \vdash i : \mathsf{int}}$$

$$\frac{\Gamma \vdash e_1 : \mathsf{int} \quad \Gamma \vdash e_2 : \mathsf{int}}{\Gamma \vdash e_1 + e_2 : \mathsf{int}}$$

```
rules

  typeOfExp(ENV, IntLit(_)) = INT().

  typeOfExp(ENV, Plus(e1, e2)) = INT() :-
    typeOfExp(ENV, e1) = INT(),
    typeOfExp(ENV, e2) = INT().
```

Demo

Security Types: Design

- Introduce *labeled types*
- `typeOfExpr` return such a type
- Assignment statements check validity
- Environment contains flow-sensitivity information.

```
sorts SEC constructors
  LOW  : SEC
  HIGH : SEC

sorts LTYPE = (TYPE * SEC)

rules

  typeOfExpr: Env * Expr -> LTYPE
```

Demo 2.0

Conclusion: What have we seen?

- Languages aim to offer right abstractions for some domain.
- Language design is an art!
- Spoofax facilitates exploration by generating compilers from high-level specifications.

Explore More

- Want to explore more?
- Spoofax: *spoofax.dev*
- Demo Language: *github.com/MetaBorgCube/metaborg-seclang/*
- Contact me: *a.s.zwaan@tudelft.nl*