Dependently Typed Languages in Statix

Jonathan Brouwer Jesper Cockx Aron Zwaan

Delft University of Technology, The Netherlands

March 15, 2023



Background: What are Dependent Types?

Types may depend on values!

Example

```
concat : (A: Set) \rightarrow (n m : Nat) \rightarrow Vec A n \rightarrow Vec A m \rightarrow Vec A (n + m)
```

- Proof assistants
- For example: Agda, Coq, Lean, ...

Research Question

How suitable is Statix for defining a dependently-typed language?

Why is this important?

Spoofax perspective

Developing a language with a complex type system tests the boundaries of what Spoofax can do.

Dependent Types perspective

Using a language workbench helps with rapid prototyping.

Primary Contribution

tododododo

Calculus of Constructions

A lambda calculus with dependent types.

```
Example 1
(\v: Type. v) T
```

Example 2

```
let f = \T: Type. \x: T. x;
f (T: Type -> Type) (\y: Type. y)
```

Type Checking

Type checking relation

```
typeOfExpr : scope * Expr -> Expr
```

How do we use scopes?

A scope is used as a combination of an environment and a context. One relation name → NameEntry, NameEntry is either:

- NType: Stores a type -> Corresponds with context
- NSubst: Stores a substitution -> Corresponds with environment

Type Checking: Requires Evaluation

Example 1

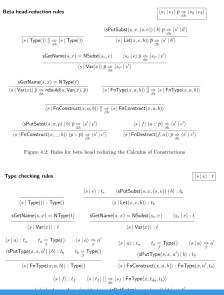
```
let T = if false then Int else Bool end;
let b: T = true;
```

Evaluation relation

```
betaHeadReduce : scope * Expr -> scope * Expr
betaReduce : scope * Expr -> Expr
exectBetaEq : (scope * Expr) * (scope * Expr)
```

Type Checking: From inference rules to Statix code

tododododod From inference rules to Statix code



Extra contributions

Features

- 1 Implemented Inference
- 2 Implemented Inductive Data Types
- 3 Implemented Universes
- 4 Interpreter
- 6 Compiler to Clojure

Evaluation

- ① Comparison with implementation in Haskell
- 2 Comparison with implementation in LambdaPi
- 3 Evaluation of Spoofax

Conclusions

Spoofax is a great tool for developing dependently typed languages! 1 - using scopes as env+context todo beperkingen noemen - statix enforces syntactic uniqueness of metavariable solutions but dependently typed languages use a weaker notion of equality - full thesis coming soon TM $\,$

¹But there is still room for improvement