A generic proof assistant

Andrej Bauer, Philipp Haselwarter, Anja Petković
University of Ljubljana

Foundations and Applications of Univalent Mathematics
18–20 December 2019, Herrsching, Germany

Andromeda 1

- Π, Id, Type
- Type : Type
- equality reflection

Andromeda 2

- Ready for α-testing since lunch!
- Generic:
  - user-definable primitive rules
  - derivable rules
  - extensible equality checker
- Overall structure:
  - nucleus – trusted 4400 lines of OCaml
  - meta-level (ML) – untrusted 12000 lines of OCaml

Structure of Andromeda

- **Trusted** nucleus
  - 4400 lines of OCaml code
  - syntactic operations (substitution, abstraction)
  - applies rules of inference
- **Untrusted** meta-level (ML)
  - 12000 lines of OCaml code
  - ML language to manage proof development
  - extensible equality checker (1400 lines of code)

General type theories

- syntax with binding
- judgement forms
- boundaries
- acceptable rules

Judgement forms

$$\Gamma \vdash A \text{ type}$$

A is a type

$$\Gamma \vdash t : A$$

t is a term of type A

$$\Gamma \vdash A \equiv B$$

types A and B are equal

$$\Gamma \vdash t \equiv u : A$$

terms t and u of type A are equal

Judgement forms

$$\Gamma \vdash A \ \text{type}$$

A is a type

$$\Gamma \vdash t : A$$

t is a term of type A

$$\Gamma \vdash A \equiv B \ \text{by} \ \xi$$

types A and B are equal

$$\Gamma \vdash t \equiv u : A \ \text{by} \ \xi$$

terms t and u of type A are equal

Boundaries

$$\Gamma \vdash \Box \ \mathsf{type}$$

construct a type

$$\Gamma \vdash \Box : A$$

inhabit type A

$$\Gamma \vdash A \equiv B \text{ by } \Box$$

prove that A and B are equal

$$\Gamma \vdash t \equiv u : A \text{ by } \Box$$

prove that t and u are equal

Structural rules

- Equality rules (user-definable)
- Congruence rules (`congruence`)
- Conversion rules (`convert`)
- Substitution rules (derivable)

Rules demo

Equality checking

- Type-directed phase, followed by normalization phase
- User-definable computation rules
- User-definable extensionality rules

Computation rule (β)

```
rule R P₁ ⋯ Pᵢ : e₁ ≡ e₂ : A
```

- P_1 … P_i are object premises
- e_1 has a head symbol and mentions all meta-variables
- examples will be shown
- counter-example:

```
rule unit_β (x : unit) : x ≡ tt : unit
```

Extensionality rule

```
rule R
  P₁ ⋯ Pᵢ (x:A) (y:A) Q₁ ⋯ Qⱼ
  : x ≡ y : A
```

- P_1 … P_i are object premises
- Q_1 … Q_j are equation premises
- Examples will be shown
- Extensionality rules are inter-derivable with η-rules

Equations demo

Future

- Use existing ML architecture to implement:
  - automatic passage between types and universe codes
  - simple tactics à la `auto`
- Missing ML architecture:
  - better meta-variables & implicit arguments
  - Agda holes
- Your music wish here