A generic proof assistant

Andrej Bauer, Philipp Haselwarter, Anja Petković University of Ljubljana

Andromeda 1

- П, Id, Туре
- 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

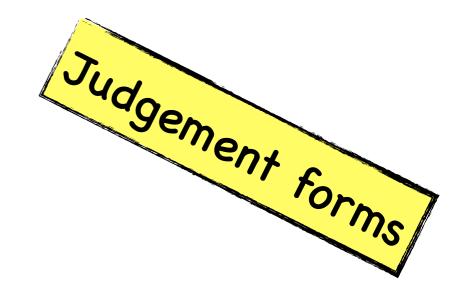
Structure of Andromeda

- 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

Γ — A type
A is a type

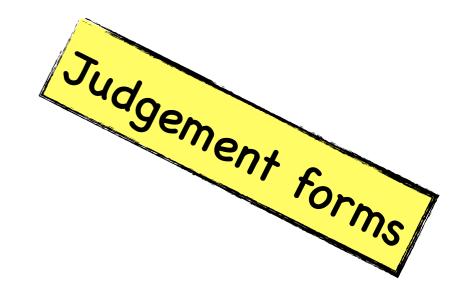


 $\Gamma \vdash t : A$ t is a term of type A

 $\Gamma \vdash A \equiv B$ types A and B are equal

 $\Gamma \vdash t \equiv u : A$ terms t and u of type A are equal

Γ — A type
A is a type

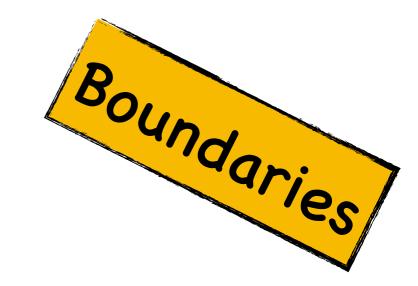


 $\Gamma \vdash t : A$ t is a term of type A

 $\Gamma \vdash A \equiv B \text{ by } \xi$ types A and B are equal

 $\Gamma \vdash t \equiv u : A by \xi$ terms t and u of type A are equal

The type construct a type



$$\Gamma \vdash A \equiv B \ by \square$$
prove that A and B are equal

$$\Gamma \vdash t \equiv u : Aby$$
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 (B)

```
rule R P1 \cdots Pi : e1 \equiv e2 : A
```

- P1 ... Pi are object premises
- e1 has a head symbol and mentions all meta-variables
- examples will be shown
- counter-example:

```
rule unit_\beta (x : unit) : x = tt : unit
```

Extensionality rule

```
rule R
  P1 ··· Pi (x:A) (y:A) Q1 ··· Q;
  : X ≡ y : A
```

- P1 ... Pi are object premises
- Q₁ ... 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