Equations: a tool for dependent pattern-matching

Cyprien Mangin
cyprien.mangin@m4x.org

Matthieu Sozeau
matthieu.sozeau@inria.fr

Inria Paris & IRIF, Université Paris-Diderot

January 14, 2017

Outline

1. Setting and overview
2. Main features
3. Recent improvements

Outline

Setting

- CIC : dependent type theory + W-types.
- Type families.
- We only allow basic pattern-matching on W-types (eliminators).
- From a list of clauses, build a splitting tree.
- From the splitting tree, build a term in CIC.

Compilation of the splitting tree

Say we want to split on a variable $(x : I\vec{u})$.

1. Generalize the variable by introducing fresh indices $\vec{v}$, a fresh variable $(y : I\vec{v})$, and added equalities between $\vec{u}$ and $\vec{v}$, and x and y.
2. Eliminate the fresh variable y.
3. Simplify the equalities.

Outline

A few examples

- Simple function definition.
- Refinement (`with` clause)
- Well-founded recursion (`by rec` keyword)

Reasoning support: equations

- Automatic generation of equations for each leaf of the splitting tree.
- Reduction of function calls without going through the reduction itself.
- Sometimes, the function does not even compute definitionally.

Reasoning support: functional elimination

EQUATIONS will automatically generate a principle of functional elimination.

- Useful to show properties about a function.
- No unnecessary cases, all the splitting and the logical reasoning is already done.

Other tools

- `depelim` tactic, which reuses the splitting mechanism inherent to Equations.
- Automatic derivation of various classes about inductive types:
  - Decidable equality.
  - Signature (pack a term in an inductive type with its indices).
  - Well-founded subterm relationship (structural recursion without the guard condition).
  - Principle of no confusion (injection and disjointness of constructors).

Outline

Local definition (`where` keyword)

- Similar to a let-in.
- Provide a definition through a splitting tree, as usual.
- Possible to combine it with well-founded recursion to obtain nested or mutual recursion.

Less axioms

Proof irrelevance was used to prove the fixpoint lemmas about well-founded recursion. We avoid it by proving it directly for the accessibility relation.
Additionally, a lot of work about the axiom K...

From heterogeneous to homogeneous equalities

When we generalize a variable $(x : I\vec{u})$, we introduce equalities. Before, we used heterogeneous equalities:

- Easy to manipulate (less dependency between equalities).
- Entails the use of the axiom K.

Now we use homogeneous equalities between telescopes.

- Have to be careful because each equality depends on the previous one.
- The use of the rule K is targeted to a specific type.

Conclusion

EQUATIONS was already used succesfully for a few applications:

- Normalization of LF.
- Consistency of predicative System F.
- Reflexive tactic to decide equality of polynomials.

For now, the main focus is to polish the current features to allow a first stable release soon.

EQUATIONS is available on GitHub [1] and OPAM.

[1] https://github.com/mattam82/coq-equations