Separation of Proof and Program

The Trellys Project

University of Iowa
Harley Eades, Garrin Kimmel, Fu Peng, and Aaron Stump.

University of Pennsylvania
Chris Casinghino, Vilhelm Sjöberg, and Stephaine Weirich.

Portland State University
Nathan Collins, Ki-Yung Han, and Tim Sheard.

MVD 2011

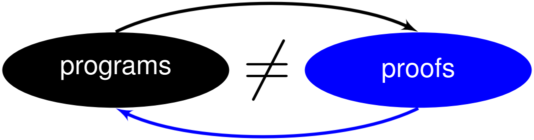
Introduction

- The design of a core language of a new dependently typed programming language Trellys.
  - Separation of Proof and Program (Sep^3).
- The logical fragment.
  - Equality, explicit conversions, a new termination predicate, case splitting on programs, and induction using a primitive ordering.
- The programmatic fragment.
  - General recursion, explicit conversions, and case splitting on programs.

Sep^3

- Sep^3 is a call-by-value language that consists of two fragments, a logical fragment and a programmatic fragment.
  - The language syntactically separates the logical and programmatic fragments.
  - The logical fragment is a predicative higher-order logic.
  - While the programmatic fragment contains general recursion and type:type.

- The two fragments are separate, but they are linked.
  - Proofs can "talk" about programs, but are not allowed to run them.
  - Programs can contain proofs.



The Logical Fragment

- A predicative higher order logic.
  - The logic is weakly constructive. What this means is that there is only one predicate that forces the logic to be non-constructive.
- The logical fragment is compile time only.
  - That is all proofs are erased during compile time.
- The logic has the following as primitives.
  - Disjunction, existential types, absurdity, higher-order predicative quantification, implication, propositional equality, explicit conversions, induction, and a new termination predicate.

Equality and Conversion

- The logic of Sep^3 has a primitive notion of propositional equality.
  - This equality is a typed equality and expresses when two programs are equivalent.
  - Intro. form: $\Gamma \vdash join\ n\ m : t_1 = t_2$.
  - Use: Suppose $|t|$ is a function that erases all the proofs from the program t then we if $|t_1| \leadsto^n t'$ and $|t_2| \leadsto^m t'$ and t_1 and t_2 are typeable then we may conclude that t_1 and t_2 are equivalent with the proof *join*.

- Explicit conversion adds the ability to make use of equalities between programs.
  - Elim. form: $\Gamma \vdash conv\ p\ by\ eqpf\ at\ hole.p : [t_2/hole]P$
  - Use: If we know p is a proof of $[p_1/hole]P$, and we can prove $t_1 = t_2$ then we can replace t_1 with t_2 in $[p_1/hole]P$ and obtain a new proof of $[p_2/hole]P$.

Termination

- The logic contains a new predicate called the termination predicate.
  - The termination predicate internalizes the notion of termination.
  - Predicate form: $t!$.
  - Explanation: For some program t if we can prove t normalizes then we may conclude $t!$.
- We not only need to show that t normalizes, but that the normal form of t can be judged a value.
  - Intro. form: $\Gamma \vdash \textit{valax}\ t : t!$.
  - Use: If we can judge t a value, denoted $\Gamma \vdash \textit{val}\ t$, then we may conclude with the proof $\Gamma \vdash \textit{valax}\ t : t!$ which states that t has a value.

  - Forms that may be judged values:
    - λ-abstractions, *Type*, recursors, data type constructors whose arguments are values, and variables marked as values.

Termination: An Example

Example

Suppose t is a program and v is t's value. Then

$$\dfrac{\dfrac{\begin{array}{l}|t| \downarrow |v| \\ \Gamma \vdash t : t' \\ \Gamma \vdash v : t'\end{array}}{\Gamma \vdash \mathit{join}\ m\ n : t = v} \qquad \dfrac{\Gamma \vdash \mathit{val}\ v}{\Gamma \vdash \mathit{valax}\ v : [v/\mathit{hole}](\mathit{hole}!)}}{\Gamma \vdash \mathit{conv}(\mathit{valax}\ v)\ \mathit{by}\ (\mathit{join}\ m\ n)\ \mathit{at}\ \mathit{hole}.(\mathit{valax}\ \mathit{hole}) : t!}$$

Termination

- How can we use the termination predicate?
  - If p is a proof of $t!$ for some program t then t can be used as a value.
    - Form: $\Gamma \vdash val\ tcast\ t\ by\ p$.
  - *tcase* provides the ability to case split on the termination behavior of programs.
    - DISCLAIMER: $t!$ is not constructive.
    - Form: $\Gamma \vdash tcase\ t\ [u]\ of\ abort\ \rightarrow p_1 \mid ! \rightarrow p_2 : P$.
    - Use: For some program t if p_1 is a proof of some predicate P assuming $t!$ and p_2 is a proof of P assuming t diverges then $tcase\ t\ [u]\ of\ abort\ \rightarrow p_1 \mid ! \rightarrow p_2$ is a proof of P.

Induction

- Sep^3 has a primitive notion of structural course-of-values induction.
  - Form: $\Gamma \vdash \mathit{ind}\ f\ x : t_1, u.p : \forall x : t_1.\forall u : x!.P$.
  - Use: If p is a proof of some predicate P assuming $\forall y : t_2.\forall u : y < x.[y/x]P$ holds then we can prove P for any program x of type t_1.

The Programmatic Fragment

- The programmatic fragment has a collapsed syntax. Terms and types are all generated by the same syntactic category.
- Type:Type
- General recursion.
  - Form: $\Gamma \vdash rec\ f\ x : t_1.t : \Pi x : t_1.t_2$.
- Explicit conversions.
- Data types and case splitting on programs.
  - Intro. Form: $data\ C\ t\ where\ \{C_1 : t_1, \ldots, C_n : t_n\}$.
  - Elm. Form: $case\ t\ [eq\_pf]\ of\ C_1\ t_1 \ldots t_n \mid \cdots \mid C_k\ t'_1 \ldots t'_2 \mid done$.

Concluding remarks

- Future work.
  - Complete the meta-theory.
  - Design and implement the surface language.
- Thank you all for listening.