Partiality and Dependent Types
Implementing a specification logic in a DTT

Kasper Svendsen, Lars Birkedal and Aleksandar Nanevski

June 2, 2011
Introduction

Dependent Type Theory

- higher-order functional programming language
- with integrated reasoning & verification
Introduction

Dependent Type Theory

- higher-order functional programming language
- with integrated reasoning & verification
- lacks general recursion
- lacks effects such as state, IO, etc.
Introduction

Dependent Type Theory

- higher-order functional programming language
- with integrated reasoning & verification
- lacks general recursion
- lacks effects such as state, IO, etc.

This talk: cheap way of adding general recursion
Partiality

unrestricted recursion breaks propositions-as-types
Partiality

unrestricted recursion breaks propositions-as-types

Our setting

- treat partiality as an effect
- use monads to add encapsulate effects in a pure language

\[
O : \text{Set} \to \text{Set} \\
\text{fix}_\tau : (O(\tau) \to O(\tau)) \to O(\tau) + \text{ret, bind}
\]
Admissibility: The problem

Problem: $\text{fix}_\tau$ unsound in sufficient expressive TTs

- the type of $\text{fix}_\tau$

$$\text{fix}_\tau : (O(\tau) \to O(\tau)) \to O(\tau)$$

corresponds to fixpoint induction

$$\forall f : X \to X. \forall P \subseteq_{adm} X. (\forall x \in P. f(x) \in P) \Rightarrow \text{fix}(f) \in P$$
Admissibility: The problem

Problem: \( \text{fix}_\tau \) unsound in sufficient expressive TTs

- the type of \( \text{fix}_\tau \)
  
  \[
  \text{fix}_\tau : (O(\tau) \to O(\tau)) \to O(\tau)
  \]
  
  corresponds to fixpoint induction

\[
\forall f : X \rightarrow X. \forall P \subseteq_{adm} X. (\forall x \in P. f(x) \in P) \Rightarrow \text{fix}(f) \in P
\]

- in STT all partial types are admissible
- but in DTT there exists inadmissible types, e.g.,

\[
O(\{ c : \mathbb{N} \to O(\mathbb{N}) \mid \exists n \in \mathbb{N}. c(n) = \Omega_{\mathbb{N}} \})
\]

where \( \Omega_{\mathbb{N}} = \text{fix}_\mathbb{N}(id_{O(\mathbb{N})}) \)
Admissibility: Previous work

**Crary**: introduce explicit admissibility proofs on \( \text{fix} \)

- very expressive & allows for easy implementation
- significant proof obligation for every use of \( \text{fix} \)

\[
\text{fix}_\tau : \text{adm}(O(\tau)) \rightarrow (O(\tau) \rightarrow O(\tau)) \rightarrow O(\tau)
\]

- complicated admissibility theory for subset- & \( \Sigma \)-types
Admissibility: Previous work

Crary: introduce explicit admissibility proofs on fix

- very expressive & allows for easy implementation
- significant proof obligation for every use of fix

\[
\text{fix}_\tau : \text{adm}(O(\tau)) \rightarrow (O(\tau) \rightarrow O(\tau)) \rightarrow O(\tau)
\]

- complicated admissibility theory for subset- & Σ-types

HTT: restrict to admissible types

- omits subset-types, strong Σ-types, inductive families
Admissibility: This talk

Idea

Only allow reasoning about effectful computations through specs (as in a program logic for an imperative language.)
Admissibility: This talk

Idea
Only allow reasoning about effectful computations through specs (as in a program logic for an imperative language.)

How?
- collapse equality on effectful computations
  
  \[ \text{if } M, N: O(\tau) \text{ then } M =_{O(\tau)} N \]

- types as only specification
Admissibility: This talk

Collapsed equality

- usual type constructors closed under admissible types
  \[ \Sigma, \Pi, \{ x : \tau \mid P \}, W, O \]
- \( \{ x : O(\tau) \mid P(x) \} \) trivially admissible, as \( P \) is constant
Admissibility: This talk

Collapsed equality

- usual type constructors closed under admissible types
  \[ \Sigma, \Pi, \{x : \tau \mid P\}, W, O \]

- \{x : O(\tau) \mid P(x)\} trivially admissible, as \(P\) is constant

- in particular,

\[ \{c : \mathbb{N} \to O(\mathbb{N}) \mid \exists n \in \mathbb{N}. c(n) = \Omega_\mathbb{N}\} \cong \mathbb{N} \to O(\mathbb{N}) \]
Admissibility: This talk

Collapsed equality

- subsets of partial types useless

\[ \{ c : \mathbb{N} \rightarrow O(\mathbb{N}) \mid \exists n \in \mathbb{N}. \ c(n) = \Omega_{\mathbb{N}} \} \cong \mathbb{N} \rightarrow O(\mathbb{N}) \]

- but partial subset types are not

\[ \prod n : \mathbb{N}. \ \prod G : \mathcal{G}. \ O(1 + \{ f : V_G \rightarrow \mathbb{N} \mid \text{coloring}(G, f, n) \}) \]

- they express partial correctness specs
Admissibility: This talk

Benefits

- avoid all admissibility conditions
- full power of underlying dependent type theory
- easily implementable as extension of existing DTT

Drawbacks

- no equational reasoning about effectful computations

Cheap implementation of a spec logic in a DTT
Hoare Type Theory

- extends DTT with partial stateful computations
Hoare Type Theory

- extends DTT with partial stateful computations
- new version: extends CIC
- implementable as axiomatic extension of Coq
Hoare Type Theory

- extends DTT with partial stateful computations
- new version: extends CIC
- implementable as axiomatic extension of Coq
- demonstrate approach scales to realistic DTTs
- illustrate expressiveness despite collapsed equality
HTT: Underlying DTT

Universes

- Prop and Set (impredicative) and Type (predicative)

\[
\begin{align*}
\text{Prop} & : \text{Type} \\
\text{Set} & : \text{Type}
\end{align*}
\]

and Prop \( \subseteq \text{Set} \subseteq \text{Type} \)

Type constructors

- Set, Type: 1, \( \Sigma \), \( \Pi \), \( W \)
- Prop: 1, \text{weak} \( \Sigma \), \( \Pi \)
HTT: Effectful computations

- partial stateful computations
- index partial types by pre- and post-condition

\[ \Gamma \vdash \{P\}_{\tau}\{Q\} : \text{Set} \]
HTT: Effectful computations

- partial stateful computations
- index partial types by pre- and post-condition
  \[ \Gamma \vdash \{P\}_\tau \{Q\} : \text{Set} \]
- heap type to reason about computation states
  \[ \text{Heap} : \text{Type} \]
  \[ \text{empty}, \ h[l \mapsto \tau \ v], \ldots \]
HTT: Effectful computations

- partial stateful computations
- index partial types by pre- and post-condition

\[ \Gamma \vdash \{ P \}_{\tau} \{ Q \} : \text{Set} \]

- heap type to reason about computation states

\[ \text{Heap} : \text{Type} \]

- empty, \( h[l \mapsto_{\tau} v] \), ...

- pre- and post-condition expressed as \text{Heap} predicates

\[ P : \text{Heap} \rightarrow \text{Prop} \]

\[ Q : \tau \rightarrow \text{Heap} \rightarrow \text{Heap} \rightarrow \text{Prop} \]
HTT: Example

Stack ADT

\(\Pi \alpha : \text{Set. } \Sigma \beta : \text{Set. } \Sigma \text{inv} : \beta \times \alpha \text{ seq} \times \text{Heap} \rightarrow \text{Prop.}\)

\(\{\lambda i. \ i = \text{empty}\}\) \(\beta\)\(\{\lambda r, i, t. \ \text{inv}(r, [], t)\}\) \(\times\)

\(\Pi r : \beta. \ \Pi v : \alpha. \ \{\lambda i. \ \exists l, \ \text{inv}(r, l, i)\}\)

1

\(\{\lambda r, i, t. \ \forall l, \ \text{inv}(r, l, i) \Rightarrow \text{inv}(r, v :: l, t)\}\) \(\times\)

...

- \(\beta\) : abstract representation type
- \(\text{inv}\) : abstract representation predicate
Admissibility in PER models

PER models

- partial equivalence relations over universal pre-domain \( V \)
  \[
  V \cong 1 + N + (V \times V) + (V \to V_\perp) + V_\perp
  \]
- models a dependent type universe with 1, \( \Sigma \), \( \Pi \), \( W \)-types
Admissibility in PER models

PER models

- partial equivalence relations over universal pre-domain \( \mathbb{V} \)

\[
\mathbb{V} \cong 1 + \mathbb{N} + (\mathbb{V} \times \mathbb{V}) + (\mathbb{V} \rightarrow \mathbb{V}_{\perp}) + \mathbb{V}_{\perp}
\]

- models a dependent type universe with 1, \( \Sigma \), \( \Pi \), \( \mathcal{W} \)-types

Partiality

- \textbf{fix} : \(( \text{in}_T(R) \rightarrow \text{in}_T(R)) \rightarrow \text{in}_T(R)\) for admissible \( R \)
- PERs model DTT + \textbf{fix}_T with explicit adm. proofs
Admissibility in PER models

Complete PERs

- closed under limits of $\omega$-chains
- all partial types admissible
- complete PERs do not model strong $\Sigma$-types
Admissibility in PER models

Monotone PERs

- a PER $R \subseteq V \times V$ is monotone iff

$$\forall x, y \in |R|. \ x \leq y \Rightarrow (x, y) \in R$$

- collapses equality on PERs with a least element
- standard DTT types ($0, 1, \mathbb{N}, +, \Sigma, \Pi, W$) monotone
- complete monotone PERs do model strong $\Sigma$-types
- CMPERs model DTT + $\textbf{fix}_\tau$ with collapsed $O$-equality
HTT model

Scales to HTT

- contexts and types modelled with assemblies
- small types modelled with complete monotone PERs
- propositions modelled as regular subobjects of assemblies
split fibred reflection \((\mathcal{R}_{el} \dashv \mathcal{I}_{el})\)  

the coproducts induced by \((\mathcal{R}_{el} \dashv \mathcal{I}_{el})\) are strong  

split generic object for \(q\) in \(\text{Asm}(\mathbb{V})\)  

\(\mathcal{I}_{el} \circ \mathcal{R}_{el}\) preserves \(W\)-types from types in the image of \(\mathcal{I}_{el}\)  

**Theorem:** Underlying DTT is sound.
Summary

We have

- presented a new approach to general recursion in DTT
- presented a semantic account of this approach
- shown that it scales to a model of Coq
- implemented it as an axiomatic extension of Coq