An adequacy theorem for partial type theory

j.w.w. Simon Huber

Göteborg, May 11, 2017

Goals

Agda as a total fragment of a partial type theory

When we write a function without the termination checker we are in the ⟪partial fragment⟫

(1) Provide a denotational semantics, so that we can justify program transformation at this level (cf. L. Augustsson's PhD thesis on the semantics of lazy ML)

(2) Prove subject reduction and inversion rules for this fragment, e.g.

$$\frac{\vdash \mathsf{S}\ M = \mathsf{S}\ M' : \mathsf{N}}{\vdash M = M' : \mathsf{N}} \qquad\qquad \frac{\vdash \Pi(x:A)B = \Pi(x:A')B'}{\vdash A = A' \qquad x : A \vdash B = B'}$$

Goals

This is part of an *adequacy* result

E.g. if $[\![M]\!] = 0$ then $M \to^* 0$

If $[\![M]\!] = [\![M']\!]$ then M and M' have the same observational behavior

In particular if a program transformation is justified semantically it will also be correct operationally

Type system

As in *Constructive mathematics and computer programming* but

-without the identity type and

-with *typed* abstraction $\lambda(x : A)M$

-we add an element $\omega : \mathsf{N}$ with $\omega = \mathsf{S}\ \omega : \mathsf{N}$

Even without this element, it might be interesting to have a proof of subject reduction in a *weak* metatheory

Domain of values

$$D \;=\; [D \to D] + \Pi\ D\ [D \to D] + \mathsf{U}_k + \mathsf{N} + \mathsf{0} + \mathsf{S}\ D$$

Non lifted sum: abstraction is not a constructor

We define an application operation $b(a)$ on D, e.g.

$(\Pi\ c\ f)(a) = \perp$ and $\mathsf{0}(a) = \perp$

Finite elements

- $\perp$ or
- $\mathsf{U}_k,\ \mathsf{N}$ or $\mathsf{0}$ or
- $\mathsf{S}\ u$ where u is finite
- $\Pi\ a\ f$ where a is finite and f is a finite function or
- a finite function

and a finite function is of the form $\perp$ or $(u_1 \mapsto v_1), \ldots, (u_n \mapsto v_n)$

Finite elements

Among these finite elements we can define the *consistent* elements

E.g. $(\perp \mapsto 0),\ (0 \mapsto \mathsf{S}\ 0)$ is not consistent

We define a predorder relation and a compatibility predicate

We get a *conditional sup semilattice*

A (general) element of the domain D is an *ideal* of this semilattice

Abstraction is not *a constructor*

Finite elements, examples

N finite element

$\perp \mapsto \mathsf{0}$ represents the constant function $\mathsf{0}$ (which does not look at its argument)

We can define $a \rightarrow b$ as $\Pi\ a\ (\perp \mapsto b)$

$\perp \rightarrow \mathsf{N}$ and $\mathsf{N} \rightarrow \perp$ are compatible and the sup is $\mathsf{N} \rightarrow \mathsf{N}$

The *strict* constant function $\mathsf{0}$ will be an 《infinite》 element, sup of all

$\mathsf{0} \mapsto \mathsf{0}, \qquad \mathsf{S}\ \mathsf{0} \mapsto \mathsf{0}, \qquad \mathsf{S}\ (\mathsf{S}\ \mathsf{0}) \mapsto \mathsf{0}, \ldots$

Typing rules

$$\frac{}{\bot : \mathsf{U}_n} \qquad \frac{}{\bot : \bot}$$

$$\frac{}{\mathsf{U}_i : \mathsf{U}_j}\, i < j \qquad \frac{}{\mathsf{N} : \mathsf{U}_j} \qquad \frac{}{0 : \mathsf{N}} \qquad \frac{u : \mathsf{N}}{\mathsf{S}\ u : \mathsf{N}} \qquad \frac{}{\bot : \mathsf{N}}$$

$$\frac{a : \mathsf{U}_j \quad u_1 : a \quad t_1 : \mathsf{U}_j \ \ \ldots \ \ u_n : a \quad t_n : \mathsf{U}_j}{\Pi\ a\ (u_1 \mapsto t_1) \ldots (u_n \mapsto t_n) : \mathsf{U}_j}$$

$$\frac{u_1 : a \quad v_1 : f(u_1) \ \ \ldots \ \ u_n : a \quad v_n : f(u_n)}{(u_1 \mapsto v_1) \ldots (u_n \mapsto v_n) : \Pi\ a\ f}$$

Typing rules

The relation $t : a$ is *decidable* on finite elements

If $t_0 : a$ and $t_1 : a$ and t_0, t_1 compatible then $t_0 \vee t_1 : a$

If $a \leqslant b$ and $t : a$ then $t : b$

We *don't* always have $u : a$ if $u \leqslant v$ and $v : a$

E.g. we have $(\mathsf{U}_1 \mapsto \mathsf{0}) \leqslant (\bot \mapsto \mathsf{0})$ and $\bot \mapsto \mathsf{0} : \mathsf{N} \to \mathsf{N}$

but *not* $\mathsf{U}_1 \mapsto \mathsf{0} : \mathsf{N} \to \mathsf{N}$

Finitary projections

$p : D \to D$ such that

$p \circ p = p$

$p\ x \leqslant x$

the image of p is a Scott domain, $D(p) = \{x \in D \mid p\ x = x\}$

This is exactly determined by a set S of finite elements such that if a and b in S compatible then $a \vee b$ in S

Finitary projections

We write $D(p) \lhd D$

$D(p)$ is a so-called subdomain of D

Finitary projections

Any finite a defines a finitary projection

$p_a\ u = \bigvee\{x : a \mid x \leqslant u\}$

If we have $a : \mathsf{U}_n$ and $f : a \rightarrow \mathsf{U}_n$ then $\Pi\ a\ f : \mathsf{U}_n$ and we have an *isomorphism* between $D(\Pi\ a\ f)$ and $\Pi(x \in D(a))D(f\ x)$

We have $u : a$ if, and only if, $p_a\ u = u$

E.g. for $a = \mathsf{N} \rightarrow \mathsf{N}$ we have $p_a\ (\mathsf{U}_3 \mapsto 0) = \perp$ and $p_a\ (0 \mapsto 0) = (0 \mapsto 0)$

Finitary projections

If $a : \mathsf{U}_k$ we write $D(a)$ for $D(p_a)$

We have $D(\mathsf{U}_n) \lhd D(\mathsf{U}_{n+1})$

The union of all $D(\mathsf{U}_n)$ is a subdomain $\mathsf{Type} \lhd D$

This is the domain of (partial) types

If a in Type we have $D(a) \lhd D$ domain of (partial) elements of type a

We define in this way the *denotational model* of type theory

Semantics

$$[\![x]\!]\rho = \rho(x) \qquad [\![M\ N]\!]\rho = [\![M]\!]\rho([\![N]\!]\rho)$$

$$[\![\mathsf{N}]\!]\rho = \mathsf{N} \qquad [\![\mathsf{U}_i]\!]\rho = \mathsf{U}_i \qquad [\![\mathsf{0}]\!]\rho = \mathsf{0} \qquad [\![\mathsf{S}\ M]\!]\rho = \mathsf{S}\ ([\![M]\!]\rho)$$

$$[\![\lambda(x : A)M]\!]\rho \ = \ u \mapsto [\![M]\!](\rho, x : [\![A]\!]\rho = u)$$

$$[\![\Pi(x : A)B]\!]\rho \ = \ \Pi([\![A]\!]\rho)\ (u \mapsto [\![B]\!](\rho, x : [\![A]\!]\rho = u))$$

where $\rho, x : a = u$ means $\rho, x = p_a\ u$.

Semantics and typing system

We could define the semantics of terms without typing rules

We consider now typing system with judgements $\Gamma \vdash M : A$ and $\Gamma \vdash M = M' : A$

If $\vdash M : A$ then $[\![M]\!] : [\![A]\!]$

If $\vdash M = M' : A$ then $[\![M]\!] = [\![M']\!]$

Semantics and typing system

The use of finitary projections takes care of eta-conversion

$$[\![\lambda(x:\mathsf{N}\to\mathsf{N})x]\!] = [\![\lambda(x:\mathsf{N}\to\mathsf{N})\lambda(y:\mathsf{N})x\ y]\!]$$

Abstraction cannot be a constructor if we want to validate the rule

$$\frac{\vdash f:\Pi(x:A)B \qquad \vdash g:\Pi(x:A)B \qquad x:A\vdash f\ x = g\ x : B}{\vdash f = g : \Pi(x:A)B}$$

Adequacy theorem

We define the relation $M \to M'$ (weak-head reduction)

E.g. $(\lambda(x : A)M)\ N \to M(x/N)$

This is defined at the level of pure expressions (no need of typing)

We define $M \to_A M'$ to mean $M \to M'$ and $M = M' : A$

We define $M \to_{\mathsf{type}} M'$ to mean $M \to M'$ and $M = M'\ \mathsf{type}$

Semantics and typing system

By induction on a finite element in Type we define a predicate $A|a$ on type expressions

We also define an equivalence relation $A = A'|a$ on the corresponding subset

If we have $A|a$ and u finite in $D(a)$ we define a predicate $M : A|u : a$ on expressions of type A

and an equivalence relation $M = M' : A|u : a$ on the corresponding subset

E.g. $A| \perp$ means that A is a type expression (no information)

Semantics and typing system

$A|\mathsf{N}$ means $A \rightarrow^*_{\mathsf{type}} \mathsf{N}$

$A = A'|\mathsf{N}$ means $A \rightarrow^*_{\mathsf{U}_n} \mathsf{N}$ and $A' \rightarrow^*_{\mathsf{type}} \mathsf{N}$

$M : A|u : \mathsf{N}$ is defined by induction on u

$M : A|0 : \mathsf{N}$ means $M \rightarrow^*_A 0$

$M : A|\mathsf{S}\ u : \mathsf{N}$ means $M \rightarrow^*_A \mathsf{S}\ M'$ and $M' : A|u : \mathsf{N}$

For instance we have $\omega : \mathsf{N}|\mathsf{S}^k\ \perp : \mathsf{N}$ for all k

Semantics and typing system

If $a = \Pi\ b\ f$ then $A|a$ means that

-$A \rightarrow^*_{\mathsf{type}} \Pi(x : B)F$ and

-$B|b$ and $N : B|v : b$ implies $F(x/N)|f(v)$ and

-$N_0 = N_1 : B|v : b$ implies $F(x/N_0) = F(x/N_1)|f(v)$

Semantics and typing system

In this case, we define $M : A|w : a$ to mean

-$N : B|v : b$ implies $M\ N : F(x/N)|w(v) : f(v)$ and

-$N_0 = N_1 : B|v : b$ implies $M\ N_0 = M\ N_1 : F(x/N_0)|w(v) : f(v)$

and we define $M_0 = M_1 : A|w : a$ to mean

-$N : B|v : b$ implies $M_0\ N = M_1\ N : F(x/N)|w(v) : f(v)$.

Semantics and typing system

For instance $a = \Pi\ \mathsf{N}\ (\mathsf{0} \mapsto \mathsf{N},\ \mathsf{S}\ \mathsf{0} \mapsto \mathsf{U}_3)$

$A|a$ means that $A \to^*_{\mathsf{type}} \Pi(x : B)F$

with $B|\mathsf{N}$ that is $B \to^*_{\mathsf{type}} \mathsf{N}$

and if $M \to^*_{\mathsf{N}} \mathsf{0}$ and $M' \to^*_{\mathsf{N}} \mathsf{0}$ then $F(x/M) = F(x/M')|\mathsf{N}$

and if $M \to^*_{\mathsf{N}} \mathsf{S}\ \mathsf{0}$ and $M' \to^*_{\mathsf{N}} \mathsf{S}\ \mathsf{0}$ then $F(x/M) = F(x/M')|\mathsf{U}_3$

Semantics and typing system

The main properties of this relation are of the form

If $b \leqslant a$ in Type and $A|a$ then $A|b$

If $b \leqslant a$ in Type and $A|a$ and u in $D(b)$ then $M : A|u : b$ if, and only if, $M : A|u : a$

If $A|a$ and $A|a'$ and a and a' are compatible then $A|a \vee a'$

Semantics and typing system

It follows from this that we can extend the relations $A|a$ and $A = A'|a$ and $M : A|u : a$ and $M = M' : A|u : a$ to a and u infinite element

This defines a logical relation between the initial/term model and the denotational model

Hence we have $A|[\![A]\!]$ if A is a type expression and $M : A|[\![M]\!] : [\![A]\!]$ if $M : A$ and $M = M' : A|[\![M]\!] : [\![A]\!]$ if $M = M' : A$

Applications

Corollary: *If* $\mathsf{S}\ M = \mathsf{S}\ M' : \mathsf{N}$ *then* $M = M' : \mathsf{N}$

We can do the same reasoning in an arbitrary, but *fixed* context

We interpret the *variables* of this context by $\perp$

We get first that if $\Gamma \vdash \Pi(x : A)B = \Pi(x : A')B'$ then $\Gamma \vdash A = A'$ and $\Gamma \vdash M : A$ implies $\Gamma \vdash B(x/M) = B'(x/M)$

but the context is arbitrary so we also get

Corollary: *If* $\Gamma \vdash \Pi(x : A)B = \Pi(x : A')B'$ *then* $\Gamma \vdash A = A'$ *and* $\Gamma, x : A \vdash B = B'$

Applications and refinement

Subject reduction follows from the fact that Π is one-to-one

We can refine the semantics by adding a token T, with $\mathsf{T}(a) = \mathsf{T}$

It represents the information: to reduce to a neutral term

Applications and refinement

We add the following typing rule

$$\frac{}{\mathsf{T} : \mathsf{U}_m} \qquad \frac{\Pi\ a\ f : \mathsf{U}_n}{\mathsf{T} : \Pi\ a\ f}$$

$$\frac{}{\mathsf{T} : \mathsf{N}} \qquad \frac{}{\mathsf{T} : \mathsf{T}}$$

Applications and refinement

We define $M = M' : A|\mathsf{T} : a$ to mean that M and M' reduce to neutral terms of the ⟪same shape⟫

Using subject reduction, and reasoning in an arbitrary context one can then show

if $M = M' : A$ then M and M' have the same Böhm tree

If we are in the *total* fragment of type theory, the Böhm trees are finite and we get a decision algorithm for conversion

Note that all this can be done in a ⟪weak⟫ metatheory (all definitions are by induction on the finite element of the domain D)