"A Predicative Strong Normalisation Proof for a λ-Calculus with Interleaving Inductive Types" Errata and New Ideas

Andreas Abel

Oct 2001, Mar 2003, Apr 2004, Jun 2012, Nov/Dec 2017

The article *A Predicative Strong Normalisation Proof for a λ-Calculus with Interleaving Inductive Types* by Thorsten Altenkirch and me appeared in 2000 in Coquand et al., *Types for Proofs and Programs (TYPES'99)*, pp. 21–40, LNCS 1956, Springer-Verlag. I found some errors which I list below. The page number is the one of the published version. To get the relative page number, subtract 20.

p. 28 Definition of weak head reduction. The correct definition is:

$$\frac{t \rhd_\beta u}{t \rhd_{\mathsf{whd}} u} \qquad \frac{t \rhd_{\mathsf{whd}} u}{E[t] \rhd_{\mathsf{whd}} E[u]}$$

p. 28 Lemma 1. Proposition 3

$$\frac{t \rhd_{\mathsf{whd}} t' \qquad E[t'] \in \mathsf{SN}}{E[t] \in \mathsf{SN}}$$

is wrong. Counterexample: Let Ω be a diverging term, $t \equiv (\lambda x.y)\,\Omega$ and $t' = y$. The correct proposition is

$$\frac{t \rhd_{\mathsf{whd}} t' \qquad t, E[t'] \in \mathsf{SN}}{E[t] \in \mathsf{SN}}$$

Proof by induction on $t, E[x] \in \mathsf{SN}$, analysing the reducts of $E[t]$. Since t has a head redex and cannot be in constructor form, each reduction of $E[t]$ has to take place in $E[x]$ or t. The interesting case is $t \rhd_1 u$ with $u \not\equiv t'$. Then by proposition 1 there exists an u' such that $u \rhd_{\mathsf{whd}} u'$ and $t' \rhd^* u'$. The last fact implies that $E[u'] \in \mathsf{SN}$. Hence, we can invoke the induction hypothesis on $u \rhd_{\mathsf{whd}} u'$ and obtain $E[u] \in \mathsf{SN}$. □

p. 36 A typo in rule (non-rec) at bottom of page: The superscript to $\mathcal{U}_i$ in the conclusion must be $\nu X.\rho$ (instead of $\mu X.\rho$).

Partiality in the composition of well-typed terms

(Erratum added 2017-11-11.)

On p. 26 we define terms of type σ via $\mathsf{Tm}^\sigma = \{t \mid \exists \Gamma.\, \Gamma \vdash t : \sigma\}$. This makes application of well-typed terms a partial operation, since, for instance, the term $\lambda z.\, x\, \mathsf{unit} \in \mathsf{Tm}^{1\to 1}$ cannot be applied to $x \in \mathsf{Tm}^1$: The first term requires x to be of function type, but the second states it is of type 1.

As a consequence, we need to be careful whenever we talk about the application of well-typed terms. This affects many places. Let us write $t\, u_0 \ldots u_n \downarrow$ to mean that this application is defined.

p. 27 Definition of the function space on term sets.

$$P \Rightarrow Q := \{t \in \mathsf{SN}^{\sigma\to\tau} \mid \forall u \in P.\, t\, u \in Q\}$$

Here, we would have to say "$\forall u \in P.\, t\, u \downarrow \Longrightarrow t\, u \in Q$".

p. 30 The urelement relation for case **(Arr)** should be

$$u\, \mathcal{U}_i^{\sigma\to\tau}\, t \iff \exists t' \in [\![\sigma]\!].\, t\, t' \downarrow \wedge\, u\, \mathcal{U}_i^{\tau}\, t\, t'$$

p. 30 Case **(Prod)**: Correct definition is:

$$[\![\sigma_1 \times \sigma_2]\!](\boldsymbol{P}) = \{\mathsf{pair}\, t_1 t_2 \mid \mathsf{pair}\, t_1 t_2 \downarrow \wedge \forall j \in \{1,2\}.\, t_j \in [\![\sigma_j]\!](\boldsymbol{P})\}^*$$

p. 14 Proposition 7 should be:

$$\frac{t \in [\![\sigma \to \tau]\!] \qquad u \in [\![\sigma]\!] \qquad t\, u \downarrow}{t\, u \in [\![\tau]\!]}\ (\text{sem-app})$$

$$\frac{\forall u \in [\![\sigma]\!].\, t[x := u] \downarrow \Longrightarrow t[x := u] \in [\![\tau]\!]}{\lambda x.t \in [\![\sigma \to \tau]\!]}\ (\text{sem-lam})$$

Paying extra attention to partiality of application (and substitution) seems rather tedious. A simple solution is to do the semantics entirely on raw terms. I.e., we simply speak of Tm instead of Tm^σ and SN instead of SN^σ and SAT instead of SAT^σ etc. This is a proven method in strong normalization proofs and we do not really lose any of our results.

Alternatively, we could define a Kripke model of well-typed terms, in the spirit of:

$$[\![\sigma \to \tau]\!]_\Gamma = \{t \mid \Gamma \vdash t : \sigma \to \tau \mid \forall \Delta,\, u \in [\![\sigma]\!]_{\Gamma,\Delta}.\ t\, u \in [\![\tau]\!]_{\Gamma,\Delta}\}$$

But this would a slightly more invasive choice.

The definition of strength for closed types

(Erratum added 2017-12-08.)

Page 6 defines the functorial strength aka map function for an open type $\rho \in \mathsf{Ty}(\boldsymbol{X})$. This definition validates the following substitution principle: Given $\boldsymbol{\sigma} \in \mathsf{Ty}(\boldsymbol{Y})$, and $f_i : \tau_i \to \tau_i'$ for $i = 1..|Y|$, then

$$(\rho(\boldsymbol{\sigma}))(\boldsymbol{f}) = \rho(\sigma_1(\boldsymbol{f}), \ldots, \sigma_{|\boldsymbol{X}|}(\boldsymbol{f})).$$

However, **it is not true that** $\rho() = \lambda x^{\rho}.\, x$ **for closed types** ρ**.** In fact, the strength for a closed type is a recursive identity function that only behaves like $\lambda x.\, x$ on closed terms.

Thus, the following remark should be deleted from page 6:

> We allow a partial instantiation of ρ which can be defined by instantiating all other places with the identity function $\lambda x^{\sigma}.x$.

Further, the proof of Proposition 10.2, case **(cons)** on page 15 is slightly wrong, as Henning Basold pointed out to me. The problem is that footnote 6 defines $\rho(\mathsf{It}\, f)$ "as an abbreviation of $\rho(\boldsymbol{\lambda x^{\tau}.x}, \mathsf{It}\, f)$".

Footnote 6 should be deleted. Instead, $\rho(\mathsf{It}\, f)$ should be understood as $\rho(\boldsymbol{\tau}, \mathsf{It}\, f)$ which more precisely is $\rho(\boldsymbol{\tau}(), \mathsf{It}\, f)$, and thanks to the substitution principle equal to $(X.\rho(\boldsymbol{\tau}, X))(\mathsf{It}\, f)$.

Addendum

Here is some new ideas on details of the paper.

On the type Fin

An inductive type of the form $\mu X.\, \mu Y.\, F(X,Y)$ is isomorphic to $\mu X.\, F(X,X)$. Hence, the type of unlabelled finitely branching (and finite-depth) trees

$$\mathsf{Fin} = \mu X.\, \mu Y.\, 1 + X \times Y$$

is isomorphic to the type of unlabelled binary trees $\mu X.1+X\times X$. (Implementing arbitrary trees by binary trees is folklore in computer science.)

On empty and unit type

In the presentation of the calculus the empty type 0 with its eliminator case_0 are explicitely added. In the presence of inductive types, both constants are also *definable*:

$$\begin{array}{lcl} 0 & \equiv & \mu X.X \\ \mathsf{case}_0 & \equiv & \mathsf{It}^{X.X}(\lambda x^{\sigma}.x) : 0 \to \sigma \end{array}$$

It holds that

$$\mathsf{case}_0(\mathsf{c}\, t) \equiv \mathsf{It}(\lambda x.x)(\mathsf{c}\, t) = (\lambda x.x)(\mathsf{It}(\lambda x.x)t) = \mathsf{case}_0\, t.$$

In the extended calculus which includes coinductive types the singleton set is definable as well:

$$\begin{array}{lcl} 1 & \equiv & \nu X.X \\ \mathsf{unit} & \equiv & \mathsf{Co}^{X.X,\, 0\to 0}(\lambda x^{0\to 0}.x)(\lambda y^{0}.y) : 1 \end{array}$$

We have

$$\mathsf{d}\,\mathsf{unit} \equiv \mathsf{d}(\mathsf{Co}(\lambda x.x)(\lambda y.y)) = \mathsf{Co}(\lambda x.x)((\lambda x.x)(\lambda y.y)) = \mathsf{unit}.$$

On the Interpretation of Coinductive Types

In Section 6.2 we have defined the interpretation of coinductive types elimination-based as the greatest fixed-point of the destructor-rule. If we define it introduction-based instead, we do not even require the urelement relation $\mathcal{U}$ for a predicative definition. The changes are the following:

We first add a new definition.

> Let $\rho \in \mathsf{Ty}(\boldsymbol{X}, X)$ a strictly-positive type, $n := |\boldsymbol{X}|$ and $P_1, \ldots, P_n$ saturated sets. A term set S is called *self-supported w.r.t. ρ and $\boldsymbol{P}$* if $S \in \mathsf{SAT}$ and $\mathsf{d}s \in [\![\rho]\!](\boldsymbol{P}, S)$ for all terms $s \in S$.

Introducing the notation $\mathsf{d}S = \{\mathsf{d}s \mid s \in S\}$ we could also have required $\mathsf{d}S \subseteq [\![\rho]\!](\boldsymbol{P}, S)$. Disregarding the destructor d which comes from the "iso"-formulation of coinductive types we could say that a self-supported set w.r.t. ρ and $\boldsymbol{P}$ is a post-fixed point of the operator $[\![\rho]\!](\boldsymbol{P}, \_)$.

The name *self-supported* comes from the notion of *support* of Pierce [Pie02, p.290] which coincides with our urelement relation $\mathcal{U}$. Using Prop. 1 we can split the post-fixed point condition on S into the two conditions $\mathsf{d}S \subseteq [\![\rho]\!](\boldsymbol{P}, \mathsf{SN}^{\nu X.\rho})$ and $\mathcal{U}^{\rho}_{n+1}(\mathsf{d}S) \subseteq S$, using the generalization $\mathcal{U}(S) := \bigcup_{s \in S} \mathcal{U}(s)$. In Pierce's terms, the first condition expresses that the support of $\mathsf{d}S$ is defined and the second condition expresses that the support of $\mathsf{d}S$ is contained in S. Hence we speak, again ignoring the "iso"-issue, of a self-supporting set.

The introduction rule for greatest fixed-point (co-elim, p. 3) states that each post-fixed point is contained in the greatest fixed point. We turn that into a introduction rule for coinductive types:

$$\frac{S \text{ self-supported w.r.t. } \rho \text{ and } \boldsymbol{P} \qquad t \in S}{t \in [\![\nu X.\rho]\!](\boldsymbol{P})} \text{ (coind)}$$

Meta-erratum, June 2012: This makes the definition of $[\![-]\!]$ impredicative, since we are existentially quantifying over a saturated set S here. In the light of this, the rest of the note seems rather pointless.

Lemma 21 *If S is self-supported then $S \subseteq [\![\nu X.\rho]\!](\boldsymbol{P})$.*

Proof. Immediately by (coind). □

The rule (destr') is now admissible:

Lemma 22 *If $t \in [\![\nu X.\rho]\!](\boldsymbol{P})$ then $\mathsf{d}t \in [\![\rho]\!](\boldsymbol{P}, [\![\nu X.\rho]\!](\boldsymbol{P}))$.*

Proof. By inversion of the assumption we have that $t \in S$ for some self-supported set S. By definition of self-support, $\mathsf{d}t \in [\![\rho]\!](\boldsymbol{P}, S)$. By the previous lemma and monotonicity of $[\![\rho]\!]$ we conclude $\mathsf{d}t \in [\![\rho]\!](\boldsymbol{P}, [\![\nu X.\rho]\!](\boldsymbol{P}))$. □

The remainder of Section 6 stays unaffected by these changes. But to improve clarity, we should say we prove that a set is self-supported instead of that it is closed under (destr'). This applies to the formulation of the proofs of Prop. 12, Lemma 3 and Prop. 15, but technically, the proofs remain unchanged.

Summarizing, we now handle coinductive types in a predicative meta-theory without coinduction, which simplifies the setting a bit. It shows that the urelement relation $\mathcal{U}$ is only required to give a predicative interpretation of inductive types, not for coinductive types. Note that here a trick similar to using a self-supportive set would not work. We could define inductive types elimination-based, but this would result in the impredicative rule

$$\frac{\forall S \in \mathsf{SAT}, f \in [\![\sigma]\!](\boldsymbol{P}, S) \Rightarrow S.\ \mathsf{It}\ f\, t \in S}{t \in [\![\mu X \sigma]\!](\boldsymbol{P})}.$$

How does one check whether $t \in [\![\nu X.\rho]\!](\boldsymbol{P})$? How does one construct a self-supported set S in this case? Pierce gives a semi-algorithm in his book on page 292: Start with the singleton $S_0 = \{t\}$ and then enrich it by its support $S_{n+1} = S_n \cup \mathcal{U}^{\rho}_{n+1}(S_n)$ until it is self-supported. When it becomes inconsistent on the way, i.e., $S_n \subseteq [\![\rho]\!](\boldsymbol{P}, \mathsf{SN}^{\nu X.\rho})$ does not hold, then t is not in the interpretation of the coinductive type, otherwise, it is. Of course, over infinite domains, this may never terminate. E.g., we could not check whether the stream of all natural numbers is in the semantics.

References

[Pie02] Benjamin C. Pierce. *Types and Programming Languages.* MIT Press, 2002.