Syntactical Strong Normalization for Intersection Types with Term Rewriting Rules

Andreas Abel*
Institut für Informatik
Ludwig-Maximilians-Universität München

4 June 2007

Abstract

We investigate the intersection type system of Coquand and Spiwack with rewrite rules and natural numbers and give an elementary proof of strong normalization which can be formalized in a weak metatheory.

1 Introduction

For typed λ-calculi which are used as languages for theorem provers, such as Agda, Coq, LEGO or Isabelle, *normalization* is a crucial property; the consistency of these provers depend on it. Usually, normalization is proven by a model construction, but recently, syntactical normalization proofs have received some interest [Val01, Dav01, JM03]. One advantage of syntactical proofs is that they explain better why a calculus is normalizing; in such proofs one can see what actually decreases in each reduction step. Another advantage is that they can be formalized in weak logical theories. For instance, a syntactic normalization proof [Abe04] of the simply-typed λ-calculus (STL) can be carried out in Twelf, a logical framework supporting higher-order abstract syntax, whose proof-theoretic strength is probably ω^{ω^ω}, well below primitive recursive arithmetic. The insight that normalization of very weak languages, like the STL, can be proven using just lexicographic structural induction over Σ_1-sentences, has recently lead to a full formalization of an intermediate language for SML in Twelf [LCH07].

In this work, we consider a λ-calculus with simple and intersection types and term rewriting and show strong normalization by *structural* means, that is, no model construction, instead finitary inductive definitions and lexicographic induction. For intersection types without term rewriting, similar normalization proofs exist [Val01, Mat00]. The present system originates from work of Coquand and Spiwack [CS06]; there it serves as the basis of a filter λ model which ultimately shows normalization of a dependently-typed logical framework with bar recursion. The filter λ model "translates" a term of the logical framework into sets of finite types the term can receive in the intersection type system. If all intersection-typable terms are strongly normalizing, then so are all terms of

*Research partially supported by the EU coordination action *TYPES* (510996).

the logical framework whose denotation is not the empty set of types in the filter λ-model. Coquand and Spiwack prove normalization for the intersection type system using reducibility candidates; however, there should be a proof with weaker means. As Berger [Ber05] explained to me, the filter λ model which links the (strong) logical framework with the (presumably weak) intersection type systems uses proof-theoretically heavy tools, hence, the link between the intersection type system and strong normalization should be a lightweight one. This works tries to substantiate Berger's intuition.

2 Intersection Type System for Term Rewriting

As language, we consider the λ-calculus with constructors and functions defined by rewriting. For simplicity, we consider only the nullary constructor 0 and the unary constructor \$ (successor) for natural numbers and functions f with rewrite rules of the shape

$$\begin{array}{lcll} f(0) & \longrightarrow & \underline{z} & \underline{z} \text{ closed} \\ f(\$x) & \longrightarrow & \underline{s} & \mathsf{FV}(\underline{s}) \subseteq \{x\}. \end{array}$$

The right hand sides $\underline{z}$ and $\underline{s}$ may mention f and other defined functions, thus, recursion is *a priori* unrestricted. Although we only consider natural numbers in the following, the techniques extend to all first-order data types, such as lists or finitely branching trees. Higher-order data types, such as infinitely branching trees and tree ordinals pose yet some technical problem. Thus, we do not cover the full language of Coquand and Spiwack. Note however that first-order datatypes (natural numbers and lists) are sufficient to treat the main application of the Coquand and Spiwack's filter λ model, strong normalization of bar recursion.

The intersection type system does not know a type Nat of all natural numbers, however, it has one singleton type for each a natural number. We use the same constructors 0 and \$ for these singleton types. The type E is the least type in the subtyping relation, it is inhabited by terms blocking reduction, such as $f(\lambda xt)$.

Types. Let I, J, K denote non-empty finite index sets and let i, j, k range over indices.

$$\begin{array}{lcll} a, b, c & ::= & \mathsf{E} \mid 0 \mid \$a & \text{ground types} \\ A, B, C & ::= & a \mid \bigcap_{i\in I}(A_i \to B_i) & \text{types} \end{array}$$

The type $A \to B$ is a special case of the last alternative. A function type is a partial description of the graph of the function, for instance, the identity λxx could receive the type $(0 \to 0) \cap (\$0 \to \$0) \cap (\$\$0 \to \$\$0)$, or more generally the type $\bigcap_{i\in I}(\$^i 0 \to \$^i 0)$ for any finite set I of natural numbers.

A binary intersection $A \cap B$ is an associative commutative idempotent operation definable by induction on A and B.

$$\begin{array}{rcl} \mathsf{E} \cap A = \mathsf{E} & \qquad & 0 \cap 0 = 0 \\ 0 \cap \$a = \mathsf{E} & & \$a \cap \$a' = \$(a \cap a') \\ 0 \cap \bigcap_{i\in I}(A_i \to B_i) = \mathsf{E} & & (A \to B) \cap (A \to B') = A \to (B \cap B') \\ \$a \cap \bigcap_{i\in I}(A_i \to B_i) = \mathsf{E} & & \big(\bigcap_{i\in I}(A_i \to B_i)\big) \\ & & \quad \cap \big(\bigcap_{i\in J}(A_i \to B_i)\big) = \bigcap_{i\in I\cup J}(A_i \to B_i) \end{array}$$

In the last clause, we assume all A_i for $i \in I \uplus J$ different. This invariant can be ensured using the but-last clause.

A measure on types $|A|$ is defined by $|a| = 0$ and $|\bigcap_{i\in I}(A_i \to B_i)| = \max\{|A_i|+1, |B_i| \mid i \in I\}$.

Subtyping $A \subseteq B$ is inductively given by the following rules [CS06].

$$\frac{}{\mathsf{E} \subseteq A} \qquad \frac{}{0 \subseteq 0} \qquad \frac{a \subseteq b}{\$a \subseteq \$b}$$

$$\frac{A \subseteq B_i \to C_i \text{ for all } i \in I}{A \subseteq \bigcap_{i\in I}(B_i \to C_i)} \qquad \frac{\left(\bigcap_{i\in J} B_i\right) \subseteq B}{\bigcap_{i\in I}(A_i \to B_i) \subseteq A \to B}\ J = \{i \mid A \subseteq A_i\} \neq \emptyset$$

This definition of subtyping is syntax-directed, however, it coincides with the usual axiomatic presentation: reflexivity, transitivity, and the usual rules for binary intersection, $A_1 \cap A_2 \subseteq A_i$ and $A \subseteq A_1 \;\&\; A \subseteq A_2 \implies A \subseteq A_1 \cap A_2$, are admissible; and the contravariant subtyping rule for function spaces is an instance of the last rule with $I = J = \{1\}$.

Typing The rules for the typing judgement $\Gamma \vdash t : A$ are taken from Coquand and Spiwack [CS06] and restricted to our set of constructors and function symbols. The first five rules are just intersection typing, the other five rules deal with constructors and functions.

$$\frac{}{\Gamma \vdash x : \Gamma(x)} \qquad \frac{\Gamma, x{:}A \vdash t : B}{\Gamma \vdash \lambda x t : A \to B} \qquad \frac{\Gamma \vdash r : A \to B \qquad \Gamma \vdash s : A}{\Gamma \vdash r\, s : B}$$

$$\frac{\Gamma \vdash r : A \qquad \Gamma \vdash r : B}{\Gamma \vdash r : A \cap B} \qquad \frac{\Gamma \vdash r : A \qquad A \subseteq B}{\Gamma \vdash r : B}$$

$$\frac{}{\Gamma \vdash 0 : 0} \qquad \frac{\Gamma \vdash r : a}{\Gamma \vdash \$r : \$a} \qquad \frac{\Gamma \vdash r : A}{\Gamma \vdash f(r) : \mathsf{E}}\ A \neq 0, \$a$$

$$\frac{\Gamma \vdash r : 0 \qquad \Gamma \vdash \underline{z} : C}{\Gamma \vdash f(r) : C}\ f(0) \longrightarrow \underline{z} \qquad \frac{\Gamma \vdash r : \$a \qquad \Gamma, x{:}a \vdash \underline{s} : C}{\Gamma \vdash f(r) : C}\ f(\$x) \longrightarrow \underline{s}$$

Observe that a recursive function f is typed "through its evaluation". For instance, if $f(0) \longrightarrow 0$ and $f(\$x) \longrightarrow \$(f(x))$, then f is the recursive identity, and to derive $y : \$^n 0 \vdash f(y) : \$^n 0$ we must derive $y : \$^m 0 \vdash f(y) : \$^m 0$ for all $m < n$ first. Hence, the whole computation tree of a recursive function application is already present in its typing. Therefore, a proof of strong normalization should be easy, in principle not harder as for the STL. In the following we substantiate this claim; although technically a bit involved, the proof has low proof-theoretic complexity.

3 Strong Normalization

In this section, we extend the normalization proof of Joachimski and Matthes [JM03] to the intersection type system of the last section. To this end, we introduce a judgement $\Gamma \vdash t \Uparrow C$ stating that $\Gamma \vdash t : C$ and t is strongly

normalizing. That this judgement is closed under substitution and application will be the main technical lemma; remember that closure under application is the difficult part in strong normalization proofs and usually requires a Tait-style logical relation argument. The basic idea behind the judgement is that typed weakly normalizing terms are closed under substitution: substituting the normal form v of a term of type $A = A_1 \to \cdots \to A_m \to B$ for x into the normal form w of another term can generate redexes if v contains subterms of the form $x\, v_1^{A_1} \ldots v_n^{A_n}$. Considering such a new β-redex $(\lambda x' w')\, v_i^{A_i}$, we observe that its degree A_i is smaller than A. Thus, the new substitution of v_i into w', which is necessary to reduce the redex, occurs with a smaller type; it might again create new redexes, however, with an even smaller type, so the whole process will eventually terminate. Watkins et. al. [WCPW03] coined the name *hereditary substitutions* for this process. Yet this normalization argument for the STL is quite old, Lévy [Lév76] attributes it to D. van Dalen.

SN: Atomic terms.

$$\frac{}{\Gamma \vdash x \downarrow \Gamma(x)} \qquad \frac{\Gamma \vdash r \downarrow \bigcap_{i\in I}(A_i \to B_i) \quad \Gamma \vdash s \Uparrow A_j \text{ for all } j \in J}{\Gamma \vdash r\, s \downarrow \bigcap_{j\in J} B_j}\ J \subseteq I$$

SN: Neutral terms.

$$\frac{\Gamma \vdash r \downarrow A \qquad A \subseteq B}{\Gamma \vdash r \Downarrow B} \qquad \frac{\Gamma \vdash r \Downarrow 0 \qquad \Gamma \vdash \underline{z}\, \vec{s} \Uparrow C}{\Gamma \vdash f(r)\, \vec{s} \Downarrow C}\ f(0) \longrightarrow \underline{z}$$

$$\frac{\Gamma \vdash r \Downarrow \$a \qquad \Gamma, x\!:\!a \vdash \underline{s}\, \vec{s} \Uparrow C}{\Gamma \vdash f(r)\, \vec{s} \Downarrow C}\ \begin{array}{l} f(\$x) \longrightarrow \underline{s} \\ x \notin \mathsf{FV}(\vec{s}) \end{array}$$

SN: Values, blocked terms, and weak head expansions.

$$\frac{\Gamma \vdash r \Downarrow A \quad A \subseteq B}{\Gamma \vdash r \Uparrow B} \qquad \frac{\Gamma, x\!:\!A_i \vdash t \Uparrow B_i \text{ for all } i \in I}{\Gamma \vdash \lambda x t \Uparrow \bigcap_{i\in I}(A_i \to B_i)} \qquad \frac{}{\Gamma \vdash 0 \Uparrow 0} \qquad \frac{\Gamma \vdash r \Uparrow a}{\Gamma \vdash \$r \Uparrow \$a}$$

$$\frac{\Gamma \vdash r \Uparrow A}{\Gamma \vdash f(r) \Uparrow \mathsf{E}}\ A \neq 0, \$a \qquad \frac{\Gamma \vdash r \Uparrow \mathsf{E} \qquad \Gamma \vdash s \Uparrow A}{\Gamma \vdash r\, s \Uparrow \mathsf{E}}$$

$$\frac{\Gamma \vdash s \Uparrow A \qquad \Gamma \vdash E[[s/x]t] \Uparrow C}{\Gamma \vdash E[(\lambda x t)\, s] \Uparrow C} \qquad \frac{\Gamma \vdash E[\underline{z}] \Uparrow C}{\Gamma \vdash E[f(0)] \Uparrow C}\ f(0) \longrightarrow \underline{z}$$

$$\frac{\Gamma \vdash r \Uparrow A \qquad \Gamma \vdash E[[r/x]\underline{s}] \Uparrow C}{\Gamma \vdash E[f(\$r)] \Uparrow C}\ f(\$x) \longrightarrow \underline{s}$$

Figure 1: Inductive characterization of SN.

The crucial property of STL which makes hereditary substitutions work is that in an *atomic* term $x\, s_1 \ldots s_n$, the types of all s_i are smaller than the type of x. In the presence of term rewriting, we can have terms like $f(x)\, s$ with a variable in the head, but now the type of s is no longer guaranteed to be smaller than the type of x. Let us call such terms *neutral*; they are of shape $E[x]$ where

$E[]$ is an *evaluation context* given by the grammar

$$E[] ::= [] \mid E[]\, s \mid f(E[]).$$

The termination argument of hereditary substitutions does no longer apply. However, we can salvage our substitution lemma by the following observation: For neutral terms of the shape $t = E[f(y\,\vec{s})]$, substitution for y might simplify the atomic part $y\,\vec{s}$ to 0 or $\$r$. In the first case t is s.n. if $E[\underline{z}]$ is, and in the second case, if $E[[r/x]\underline{s}]$ is.

Taking these thoughts into account we simultaneously define the three judgements (see Fig. 1):

$$\begin{array}{ll} \Gamma \vdash t \downarrow A & t \text{ is s.n. and atomic of type } A \\ \Gamma \vdash t \Downarrow A & t \text{ is s.n. and neutral of type } A \\ \Gamma \vdash t \Uparrow A & t \text{ is s.n. of type } A \end{array}$$

Lemma 1 (Weakening) *Let* $\Gamma' \subseteq \Gamma$.

1. *If* $\mathcal{D} :: \Gamma \vdash t \downarrow B$ *then* $\Gamma' \vdash t \downarrow A$ *for some* $A \subseteq B$.
2. *If* $\mathcal{D} :: \Gamma \vdash t \Downarrow B$ *then* $\Gamma' \vdash t \Downarrow B$.
3. *If* $\mathcal{D} :: \Gamma \vdash t \Uparrow B$ *and* $B \subseteq C$ *then* $\Gamma' \vdash t \Uparrow C$.

Proof. Simultaneously by induction on $\mathcal{D}$. □

The three judgements are closed under intersection: For $\mathcal{R} \in \{\downarrow, \Downarrow, \Uparrow\}$, $\Gamma \vdash t\ \mathcal{R}\ A$ and $\Gamma \vdash t\ \mathcal{R}\ B$ imply $\Gamma \vdash t\ \mathcal{R}\ A \cap B$, which can be proven simultaneously for all three $\mathcal{R}$ by induction.

Now we come to the crucial substitution lemma. Substitution for x in a derivation of

$$\frac{\Gamma, x{:}A \vdash r \Downarrow \$b \qquad \Gamma, x{:}A, y{:}b \vdash \underline{s}\,\vec{t} \Uparrow C}{\Gamma, x{:}A \vdash f(r)\,\vec{t} \Downarrow C}\ f(\$y) \longrightarrow \underline{s}$$

might trigger a substitution for y which in turn might trigger new substitutions. To establish termination of this process we require y to be of base type b. This is the reason why we disallow higher-order datatypes, which can have elements of higher types as arguments to their constructors. We generalize the substitution lemma of Joachimski and Matthes [JM03] to simultaneous substitution, but only one substituted variable may be of higher type.

Lemma 2 (Substitution and application) *Let* $\Gamma \vdash s \Uparrow A$ *and* $\Gamma \vdash s_i \Uparrow a_i$ *for* $i \in I$. *Let* $r' = [s/x][\vec{s}/\vec{x}]r$. *Let* $\mathcal{R} \in \{\Downarrow, \Uparrow\}$.

1. *If* $\mathcal{D} :: \Gamma, x{:}A, \vec{x}{:}\vec{a} \vdash r \downarrow C$ *then either* $\Gamma \vdash r' \downarrow C$, *or* $\Gamma \vdash r' \Uparrow C$ *and* $|C| \leq |A|$.
2. *If* $\mathcal{D} :: \Gamma, x{:}A, \vec{x}{:}\vec{a} \vdash r\ \mathcal{R}\ C$ *then* $\Gamma \vdash r' \Uparrow C$.
3. *If* $\mathcal{D} :: \Gamma \vdash r\ \mathcal{R} \bigcap_{i \in I}(A_i \to C_i)$ *and* $A = A_j$ *then* $\Gamma \vdash r\,s\ \mathcal{R}\ C_j$.

Proof. By lexicographic induction on $(|A|, \mathcal{D})$. For Prop. 1 consider the cases:

Case $\Gamma, x{:}A, \vec{x}{:}\vec{a} \vdash x \downarrow A$. Then $r' = s$ and $\Gamma \vdash r' \Uparrow A$ with $|A| \leq |A|$.

Case $\Gamma, x{:}A, \vec{x}{:}\vec{a} \vdash x_i \downarrow a_i$. Then $r' = s_i$ and $\Gamma \vdash r' \Uparrow a_i$, and $|a_i| = 0 \leq |A|$.

Case $\Gamma, x{:}A, \vec{x}{:}\vec{a} \vdash y \downarrow B$ for $y \notin \{x, \vec{x}\}$. Then $r' = y$ and $\Gamma \vdash r' \downarrow B$.

Case $J \subseteq I$ and

$$\frac{\Gamma, x{:}A, \vec{x}{:}\vec{a} \vdash t \downarrow \bigcap_{i\in I}(A_i \to B_i) \qquad \Gamma, x{:}A, \vec{x}{:}\vec{a} \vdash u \Uparrow A_i \text{ for all } i \in J}{\Gamma, x{:}A, \vec{x}{:}\vec{a} \vdash t\, u \downarrow \bigcap_{i\in J} B_i}$$

Let $t' = [s/x][\vec{s}/\vec{x}]t$ and $u' = [s/x][\vec{s}/\vec{x}]u$. By second induction hypothesis, $\Gamma \vdash u' \Uparrow A_i$ for all $i \in J$. We distinguish cases on the first induction hypothesis:

Subcase $\Gamma \vdash t' \downarrow \bigcap_{i\in I}(A_i \to B_i)$. Then $\Gamma \vdash t'\, u' \downarrow \bigcap_{i\in J} B_i$.

Subcase $\Gamma \vdash t' \Uparrow \bigcap_{i\in I}(A_i \to B_i)$ and $|\bigcap_{i\in I}(A_i \to B_i)| \leq |A|$. Then for each $i \in J$, we have $|A_j| < |A|$ and, thus, can apply induction hypothesis 3 to obtain $\Gamma \vdash t'\, u' \Uparrow B_i$ and $|B_i| \leq |A|$. Thus, $|\bigcap_{i\in J} B_i| \leq |A|$, and by closure unter intersection, $\Gamma \vdash t'\, u' \Uparrow \bigcap_{i\in J} B_i$.

The principal case of Proposition 2 is:

Case $f(\$y) \longrightarrow \underline{s}$, $y \notin \mathsf{FV}(\vec{t})$ and

$$\frac{\Gamma, x{:}A, \vec{x}{:}\vec{a} \vdash r \Downarrow \$b \qquad \Gamma, x{:}A, \vec{x}{:}\vec{a}, y{:}b \vdash \underline{s}\,\vec{t} \Uparrow C}{\Gamma, x{:}A, \vec{x}{:}\vec{a} \vdash f(r)\,\vec{t} \Downarrow C}$$

By induction hypothesis, $\mathcal{D}' :: \Gamma \vdash r' \Uparrow \$b$. Let $t'_j = [s/x][\vec{s}/\vec{x}]t_j$ for all j. We show $\Gamma \vdash f(r')\,\vec{t'} \Uparrow C$ by a local induction on $\mathcal{D}'$.

Subcase $r' = \$r''$ and $\Gamma \vdash r'' \Uparrow b$. Then by main induction hypothesis $\Gamma \vdash ([r''/y]\underline{s})\,\vec{t'} \Uparrow C$ which implies $\Gamma \vdash f(\$r'')\,\vec{t'} \Uparrow C$.

Subcase $r' = E[(\lambda z t)u]$ and

$$\frac{\Gamma \vdash u \Uparrow A' \qquad \Gamma \vdash E[[u/z]t] \Uparrow \$b}{\Gamma \vdash E[(\lambda z t)\,u] \Uparrow \$b}$$

Let $E'[] = f(E[])\,\vec{t'}$. By local induction hypothesis, $\Gamma \vdash E'[[u/z]t] \Uparrow C$, hence, $\Gamma \vdash E'[(\lambda z t)\,u] \Uparrow C$. The other cases of weak head expansions are treated analogously.

Subcase

$$\frac{\Gamma \vdash r' \Downarrow \$b' \qquad b' \subseteq b}{\Gamma \vdash r' \Uparrow \$b}$$

By main induction hypothesis, $\Gamma, y{:}b \vdash \underline{s}\,\vec{t'} \Uparrow C$. By the weakening lemma, $\Gamma, y{:}b' \vdash \underline{s}\,\vec{t'} \Uparrow C$, which entails $\Gamma \vdash f(r')\,\vec{t'} \Downarrow C$.

Subcase

$$\frac{\Gamma \vdash r' \Downarrow \mathsf{E} \qquad \mathsf{E} \subseteq \$b}{\Gamma \vdash r' \Uparrow \$b}$$

Then $\Gamma \vdash f(r') \Uparrow \mathsf{E}$, which implies $\Gamma \vdash f(r')\,\vec{t'} \Uparrow \mathsf{E}$ by iterated application. We conclude $\Gamma \vdash f(r')\,\vec{t'} \Uparrow C$ by weakening.

The principal case for Proposition 3 is:

Case

$$\frac{\Gamma, x{:}A_i \vdash t \Uparrow C_i \text{ for all } i \in I}{\Gamma \vdash \lambda x t \Uparrow \bigcap_{i\in I}(A_i \to C_i)}$$

By Prop. 2, $\Gamma \vdash [s/x]t \Uparrow C_j$. By weak head expansion, $\Gamma \vdash (\lambda x t)\, s \Uparrow C_j$. □

Lemma 3 (Recursion)

1. *If* $\mathcal{D} :: \Gamma \vdash r \Uparrow 0$ *and* $\Gamma \vdash \underline{z} \Uparrow C$ *then* $\Gamma \vdash f(r) \Uparrow C$.
2. *If* $\mathcal{D} :: \Gamma \vdash r \Uparrow \$a$ *and* $\Gamma, x{:}a \vdash \underline{s} \Uparrow C$ *then* $\Gamma \vdash f(r) \Uparrow C$.

Proof. Each by induction on $\mathcal{D}$. This essentially repeats proofs carried out for Proposition 2 in the previous lemma. □

Now we have shown that the judgement $\Uparrow$ is closed under all eliminations (intersection elim., application, recursion), hence, each well-typed term is in $\Uparrow$:

Theorem 4 *If* $\Gamma \vdash t : C$ *then* $\Gamma \vdash t \Uparrow C$.

Proof. By induction on $\Gamma \vdash t : C$. □

It remains to show that if $\Gamma \vdash t \Uparrow C$ then t is indeed strongly normalizing. Consider the following rule:

$$\frac{\Gamma \vdash r \Uparrow A}{\Gamma \vdash f(r) \Uparrow \mathsf{E}}\ A \neq 0, \$a.$$

To show strong normalization of $f(r)$ from s.n. of r we additionally need to know that r will never reduce to 0 or $\$r'$ for some r'.

Theorem 5 *Let* $\mathcal{R} \in \{\downarrow, \Downarrow, \Uparrow\}$ *and* $\Gamma \vdash t\ \mathcal{R}\ C$. *Then* t *is strongly normalizing. If* $\mathcal{R} \neq \Uparrow$, *then* t *is neutral. Otherwise,*

1. *if* $C \neq 0$, *then* $t \not\longrightarrow^* 0$,
2. *if* $C \neq \$c$, *then* $t \not\longrightarrow^* \$t'$ *for any* t', *and*
3. *if* $C \neq \bigcap_{i\in I}(A_i \to B_i)$, *then* $t \not\longrightarrow^* \lambda x t'$ *for any* t'.

Proof. By induction on $\Gamma \vdash t\ \mathcal{R}\ C$. For each rule we do a local Noetherian induction on the strong normalization of the terms in the premises. □

4 Conclusion

We have proven strong normalization for a λ-calculus with intersection types and term rewriting without the use of reducibility candidates or Tait-style saturated sets. The proof is technically involved, but can be formalized in a weak metatheory. Formalization in Twelf is not directly possible, not because Twelf's induction principles are not strong enough, but because we use constructs like $\bigcap_{i\in I}(A_i \to B_i)$ or simultaneous substitutions, which are not representable in Twelf, at least not directly.

We have partially answered our conjecture in the affirmative, that the intersection type system of Coquand and Spiwack [CS06] can be proven normalizing with simple means. What remains is an extension to higher-order datatypes. Also, our proof relies substantially on a *deterministic weak head reduction* relation, it is therefore not clear how we could handle overlapping patterns, an extension Coquand and Spiwack discuss in the conclusion of their article. Neither they nor we handle non-computational rewrite rules like $(x+y)+z \longrightarrow x+(y+z)$.

Riba [Rib07] considers a similar system, without datatypes yet with rewrite rules, and with union types. He shows that the elimination rule for union types can lead to diverging terms in some cases, and isolates conditions when these cases cannot occur. It would be interesting to see whether our proof could be extended to union types, and where Riba's conditions materialize in the proof.

Thanks to Thierry Coquand for interesting discussions.

References

[Abe04] Andreas Abel. Weak normalization for the simply-typed lambda-calculus in Twelf. In *LFM'04*, 2004.

[Ber05] Ulrich Berger. Continuous semantics for strong normalization. In *CiE'05*, volume 3526 of *LNCS*, pages 23–34. Springer, 2005.

[CS06] Thierry Coquand and Arnaud Spiwack. A proof of strong normalisation using domain theory. In *LICS'06*, pages 307–316. IEEE CS Press, 2006.

[Dav01] René David. Normalization without reducibility. *APAL*, 107(1–3):121–130, 2001.

[JM03] Felix Joachimski and Ralph Matthes. Short proofs of normalization. *AML*, 42(1):59–87, 2003.

[LCH07] Daniel K. Lee, Karl Crary, and Robert Harper. Towards a mechanized metatheory of Standard ML. In *POPL'07*, pages 173–184. ACM, 2007.

[Lév76] Jean-Jacques Lévy. An algebraic interpretation of the $\lambda\beta$K-calculus; and an application of a labelled λ-calculus. *TCS*, 2(1):97–114, 1976.

[Mat00] Ralph Matthes. Characterizing strongly normalizing terms of a calculus with generalized applications via intersection types. In *ITRS'00*, pages 339–354. Carleton Scientific, 2000.

[Rib07] Colin Riba. Strong normalization as safe interaction. In *Logics in Computer Science, LICS'07*, 2007. To appear.

[Val01] Silvio Valentini. An elementary proof of strong normalization for intersection types. *AML*, 40(7):475–488, October 2001.

[WCPW03] Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. A concurrent logical framework I: Judgements and properties. Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, 2003.