Admissibility of Fixpoint Induction over Partial Types

Karl Crary

October 1998
CMU-CS-98-164

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Partial types allow the reasoning about partial functions in type theory. The partial functions of main interest are recursively computed functions, which are commonly assigned types using fixpoint induction. However, fixpoint induction is valid only on *admissible* types. Previous work has shown many types to be admissible, but has not shown any dependent products to be admissible. Disallowing recursion on dependent product types substantially reduces the expressiveness of the logic; for example, it prevents much reasoning about modules, objects and algebras.

In this paper I present two new tools, *predicate-admissibility* and *monotonicity,* for showing types to be admissible. These tools show a wide class of types to be admissible; in particular, they show many dependent products to be admissible. This alleviates difficulties in applying partial types to theorem proving in practice. I also present a general least upper bound theorem for fixed points with regard to a computational approximation relation, and show an elegant application of the theorem to compactness.

This research was conducted while the author was at Cornell University. This material is based on work supported in part by ARPA/AF grant F30602-95-1-0047, and AASERT grant N00014-95-1-0985.

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not reflect the views of these agencies.

Keywords: Admissibility, fixpoint induction, partial functions, type theory.

1 Introduction

One of the earliest logical theorem provers was the LCF system [12], based on the logic of partial computable functions [21, 22]. Although LCF enjoyed many groundbreaking successes, one problem it faced was that, although it supported a natural notion of *partial* function, it had difficulty expressing the notion of a *total* function. Later theorem provers based on constructive type theory, such as Nuprl [5], based on Martin-Löf type theory [19], and Coq [3], based on the Calculus of Constructions [10], faced the opposite problem; they had a natural notion of total functions, but had difficulty dealing with partial functions. The lack of partial functions seriously limited the scope of those theorem provers, because it made them unable to reason about programs in real programming languages where recursion does not always necessarily terminate.

This problem was addressed by Constable and Smith [8], who introduced into their type theory the *partial type* $\overline{T}$, which is like a "lifted" version of T. The type $\overline{T}$ contains all members of T as well as all divergent terms. Using the partial type, partial functions from A to B may be given the type $A \to \overline{B}$. That is, when applied to an argument in A, such a function either diverges or converges to a result in B.

In a partial type theory, recursively defined objects may be typed using the *fixpoint principle:* if f has type $\overline{T} \to \overline{T}$ then $\mathit{fix}(f)$ has type $\overline{T}$. However, the fixpoint principle is not valid for every type T; it is only valid for types that are *admissible.* This phenomenon was not unknown to LCF; LCF used the related device of fixpoint induction, which was valid only for admissible predicates. When the user attempted to invoke fixpoint induction, the system would automatically check that the goal was admissible using a set of syntactic rules [16].

Despite their obvious uses in program analysis, partial types have seen little use in theorem proving systems [9, 4, 2]. This is due in large part to the fact that too few types have been known to be admissible. Smith [24] gave a significant class of admissible types for a Nuprl-like theory, but his class required product types to be non-dependent. The type $\Sigma x{:}A.B$ (where x appears free in B) was explicitly excluded. Later, Smith [23] extended his class to include some dependent products $\Sigma x{:}A.B$, but disallowed any free occurrences of x to the left of an arrow in B. Partial type extensions to Coq [2] were also restrictive, assuming function spaces to be the only type constructor. These restrictions are quite strong; dependent products are used in encodings of modules [18], objects [20], algebras [17], and even such simple devices as variant records. Furthermore, ruling out dependent products disallows reasoning using fixpoint induction as in LCF [24, 11]. Finally, the restriction is particularly unsatisfying since most types used in practice do turn out to be admissible, and may be shown so by metatheoretical reasoning.

In this paper I present a very wide class of admissible types using two devices, a condition called *predicate-admissibility* and a monotonicity condition. In particular, many dependent products may be shown to be admissible. Predicate-admissibility relates to when the limit of a chain of type approximations contains certain terms, whereas admissibility relates to the membership of a single type. The term "predicate-admissibility" stems from its similarity to the notion of admissibility of predicates in domain theory (and LCF), where there has been considerable research (this work was influenced by Igarashi [16], for example), but I will not discuss the connection in this paper. Monotonicity is a simpler condition that will be useful for showing types admissible that do not involve partiality.

	Type Formation	Introduction	Elimination
universe i	$\mathbb{U}_i$ (for $i \geq 1$)	type formation operators	
disjoint union	$T_1 + T_2$	$inj_1(e)$ $inj_2(e)$	$case(e, x_1.e_1, x_2.e_2)$
function space	$\Pi x{:}T_1.T_2$	$\lambda x.e$	$e_1 e_2$
product space	$\Sigma x{:}T_1.T_2$	$\langle e_1, e_2 \rangle$	$\pi_1(e)$ $\pi_2(e)$
natural numbers	$\mathbb{N}$	$0, 1, 2, \ldots$	assorted operations
equality	$t_1 = t_2\ in\ T$	$\star$	
partial type	$\overline{T}$		
convergence	$t\ in!\ T$	$\star$	

Figure 1: Type Theory Syntax

The paper is organized as follows: In Section 2 I lay out the theory for which these results are formalized. In Section 3 I prove some computational lemmas needed for the admissibility results. The primary result is a least upper bound theorem for fixed points with regard to a computational approximation relation. This result is quite general, and may be applied more widely than just to the purposes for which I use it. I present my main results in Section 4, beginning with a summary of Smith's original admissibility class and then widening the class using predicate-admissibility and monotonicity. Concluding remarks appear in Section 5.

2 The Type Theory

The type theory in which I formalize the results of this paper is a variant of the Nuprl type theory [5] extended with partial types (that is, types containing possibly divergent objects). This theory is a subset of the type theory of Crary [11] and is similar to Smith's theory [24]. The major difference between the theory used here and Smith's is that the latter does not provide a notion of equality; the ramifications of handling equality are discussed in Crary [11].

2.1 Preliminaries

As data types, the theory contains natural numbers (denoted by $\mathbb{N}$), disjoint unions (denoted by $T_1 + T_2$), dependent products[1] (denoted by $\Sigma x{:}T_1.T_2$), and dependent function spaces (denoted by $\Pi x{:}T_1.T_2$). When x does not appear free in T_2, I write $T_1 \times T_2$ for $\Sigma x{:}T_1.T_2$ and $T_1 \rightarrow T_2$ for $\Pi x{:}T_1.T_2$. As usual, alpha-equivalent terms are considered identical. When t_1 and t_2 are alpha-equivalent, I write $t_1 \equiv t_2$.

Types themselves are terms in the theory and belong to a predicative hierarchy of universes,

[1]These are sometimes referred to in the literature as dependent sums, but I prefer the terminology to suggest the connection to the non-dependent type $T_1 \times T_2$.

$\mathbb{U}_1, \mathbb{U}_2, \mathbb{U}_3$, etc. The universe $\mathbb{U}_1$ contains all types built from the base types only (*i.e.,* built without universes), and the universe $\mathbb{U}_{i+1}$ contains all types build from the base types and the universes $\mathbb{U}_1, \ldots, \mathbb{U}_i$. In particular, no universe is a member of itself. Propositions are interpreted as types using the propositions-as-types principle [14], but that will only be relevant in Section 4.3.

Each type T comes with an intrinsic equality relation denoted by $t_1 = t_2 \in T$. Membership is also derived from this relation; $t \in T$ when $t = t \in T$. The equality relation is introduced into the type theory as the type $t_1 = t_2 \; in \; T$, which is inhabited by the term $\star$ when $t_1 = t_2 \in T$ and is empty otherwise, provided that $t_1, t_2 \in T$. If either of t_1 or t_2 does not belong to T, then $t_1 = t_2 \; in \; T$ is not well-formed. (Note that $t_1 = t_2 \in T$ is a metatheoretical assertion whereas $t_1 = t_2 \; in \; T$ is a type in the theory.) The empty type *Void* is defined as $0 = 1 \; in \; \mathbb{N}$.

The *partial type* $\overline{T}$ is like a "lifted" version of T; it contains all the members of T as well as all divergent terms. Partial functions from A to B may then be given the type $A \to \overline{B}$. Two terms are equal in $\overline{T}$ if they both diverge, or if they both converge and are equal in T.

Convergence is expressed within the type theory by the type $t \; in! \; T$, which is inhabited by the term $\star$ when $t \in \overline{T}$ and t converges, and is empty if $t \in \overline{T}$ but t does not converge. If $t \notin \overline{T}$ then $t \; in! \; T$ is not well-formed. (Again, note that $t \; in! \; T$ is a type in the theory, but convergence, which is defined formally below, is a metatheoretical assertion.)

2.2 Computation

Underlying the type theory is the computation system shown in Figure 2. The computation system is defined by a small-step evaluation relation (denoted by $t_1 \mapsto t_2$), and a set of canonical terms. Whether a term is canonical is governed by its outermost operator; the canonical terms are those appearing in the first and second columns of Figure 1. The computation system is call-by-name and contains operators for constructing and destructing functions, pairs and disjoint unions. The computation system also contains various standard operations for computing and analyzing natural numbers, but these are not particularly interesting and are omitted from Figure 2. Of particular interest is the operator *fix*, which allows the recursive definition of objects is evaluated by the rule $\mathit{fix}(f) \mapsto f(\mathit{fix}(f))$.[2] Two important properties of evaluation are that evaluation is deterministic and canonical forms are terminal:

Proposition 1 If $t \mapsto t_1$ and $t \mapsto t_2$ then $t_1 \equiv t_2$. Moreover, if t is canonical then $t \not\mapsto t'$ for any t'.

If $t \mapsto^* t'$ and t' is canonical then I say that t converges (abbreviated $t\!\downarrow$) and t converges to t' (abbreviated $t \Downarrow t'$). Note that if $t \Downarrow t_1$ and $t \Downarrow t_2$ then $t_1 \equiv t_2$ and that if t is canonical then $t \Downarrow t$.

The computation system is used in Figure 3 to define the relation $t_1 = t_2 \in T$, which specifies the memberships of types and when terms are equal in those types.[3] This equality relation is constructed to respect evaluation: if $t \in T$ and $t \mapsto t'$ then $t = t' \in T$.

[2] The use of a *fix* operator greatly simplifies the presentation of these results (particularly the proof of Theorem 8), but it could be eliminated and replaced with the Y combinator. Similarly, the choice of a call-by-name computation system simplifies the formalism, but is also not critical to the results.

[3] Since the definition contains negative occurrences of $t_1 = t_2 \in T$, it is not immediately clear that it is a valid definition. Allen [1] and Harper [13] have shown how such a definition may be converted to a conventional inductive definition.

$$\frac{f \mapsto f'}{f\,e \mapsto f'\,e} \qquad \frac{t \mapsto t'}{\pi_i(t) \mapsto \pi_i(t')} \qquad \frac{t \mapsto t'}{\mathit{case}(t, x.e_1, x.e_2) \mapsto \mathit{case}(t', x.e_1, x.e_2)}$$

$$\frac{}{(\lambda x.e)t \mapsto e[t/x]} \qquad \frac{}{\pi_i(\langle t_1, t_2\rangle) \mapsto t_i} \qquad \frac{}{\mathit{case}(\mathit{inj}_i(t), x.e_1, x.e_2) \mapsto e_i[t/x]}$$

$$\frac{}{\mathit{fix}(f) \mapsto f\,\mathit{fix}(f)}$$

Figure 2: The Computation System

2.3 The Fixpoint Principle

The central issue of this paper is the *fixpoint principle:*

$$f \in \overline{T} \to \overline{T} \Rightarrow \mathit{fix}(f) \in \overline{T}$$

The fixpoint principle allows us to type recursively defined objects, such as recursive functions. Unfortunately, unlike in programming languages, where the principle can usually be invoked on arbitrary types, expressive type theories such as the one in this paper contain types for which the fixpoint principle is not valid. I shall informally say that a type is *admissible* if the fixpoint principle is valid for that type and give a formal definition in Section 4. To make maximum use of a partial type theory, one wants as large a class of admissible types as possible.

In Section 4 I will explore two wide classes of admissible types, one derived from a *predicate-admissibility* condition and another derived from a monotonicity condition. But first, it is worthwhile to note that there are indeed inadmissible types:

Theorem 2 There exist inadmissible types.

Proof Sketch

This example is due to Smith [24]. Let T be the type of functions that do not halt for all inputs, and let f be the function that halts on zero, and on any other n immediately recurses with $n-1$. This is formalized as follows:

$$\begin{aligned} T &\stackrel{\text{def}}{=} \Sigma h{:}(\mathbb{N} \to \overline{\mathbb{N}}).\,((\Pi x{:}\mathbb{N}.\, h\; x\; \mathit{in}!\; \mathbb{N}) \to \mathit{Void}) \\ f &\stackrel{\text{def}}{=} \lambda p.\langle \lambda x.\, \mathit{if}\; x = 0\; \mathit{then}\; 0\; \mathit{else}\; \pi_1(p)(x-1), \lambda y.\star\rangle \end{aligned}$$

Intuitively, any finite approximation of $\mathit{fix}(f)$ will recurse some limited number of times and then give up, placing it in T, but $\mathit{fix}(f)$ will halt for every input, excluding it from T. Formally, the function f has type $\overline{T} \to \overline{T}$, but $\mathit{fix}(f) \notin \overline{T}$. (The proof of these two facts appears in Appendix A.) Therefore T is not admissible.

3 Computational Lemmas

Before presenting my main results in Section 4, I first require some lemmas about the computational behavior of the fixpoint operator. The central result is that $\mathit{fix}(f)$ is the least upper bound of the

$t \in T$	iff $t = t \in T$
T *type*	iff $T = T$
$T_1 = T_2$	iff $\exists T_1', T_2'.\ (T_1 \Downarrow T_1') \wedge (T_2 \Downarrow T_2') \wedge (T_1' = T_2')$
$t_1 = t_2 \in T$	iff $\exists t_1', t_2', T'.\ (t_1 \Downarrow t_1') \wedge (t_2 \Downarrow t_2') \wedge (T \Downarrow T') \wedge (t_1' = t_2' \in T')$
$n = n' \in \mathbb{N}$ (n, n' natural numbers)	iff $n \equiv n'$
$inj_1(a) = inj_1(a') \in A + B$	iff $A + B\ \ type \wedge a = a' \in A$
$inj_2(b) = inj_2(b') \in A + B$	iff $A + B\ \ type \wedge b = b' \in B$
$\langle a, b\rangle = \langle a', b'\rangle \in \Sigma x{:}A.B$	iff $\Sigma x{:}A.B\ \ type \wedge (a = a' \in A) \wedge (b = b' \in B[a/x])$
$\lambda x.b = \lambda x.b' \in \Pi x{:}A.B$	iff $\Pi x{:}A.B\ \ type \wedge \forall a = a' \in A.\ b[a/x] = b'[a'/x] \in B[a/x]$
$t = t' \in \overline{T}$	iff $\overline{T}\ \ type \wedge (t\downarrow \Leftrightarrow t'\downarrow) \wedge (t\downarrow \Rightarrow t = t' \in T)$
$\star \in (a = a'\ in\ A)$	iff $(a = a'\ in\ A)\ \ type \wedge (a = a' \in A)$
$\star \in (a\ in!\ A)$	iff $(a\ in!\ A)\ \ type \wedge a\downarrow$

For $T \simeq T'$ iff $T = T'$, and for $T \simeq T'$ iff $T = T' \in \mathbb{U}_i$:

$\mathbb{N} \simeq \mathbb{N}$	
$A + B \simeq A' + B'$	iff $A \simeq A' \wedge B \simeq B'$
$\Sigma x{:}A.B \simeq \Sigma x{:}A'.B'$	iff $A \simeq A' \wedge \forall a = a' \in A.\ B[a/x] \simeq B'[a'/x]$
$\Pi x{:}A.B \simeq \Pi x{:}A'.B'$	iff $A \simeq A' \wedge \forall a = a' \in A.\ B[a/x] \simeq B'[a'/x]$
$\overline{T} \simeq \overline{T'}$	iff $T \simeq T' \wedge \forall t \in T.\ t\downarrow$
$(a_1 = a_2\ in\ A) \simeq$ $(a_1' = a_2'\ in\ A')$	iff $A \simeq A' \wedge a_1 = a_1' \in A \wedge a_2 = a_2' \in A$
$(a\ in!\ A) \simeq (a'\ in!\ A')$	iff $A \simeq A' \wedge a = a' \in \overline{A}$
$\mathbb{U}_i$ *type*	
$\mathbb{U}_i \in \mathbb{U}_j$	iff $i < j$

Figure 3: Type Definitions

finite approximations $\bot, f(\bot), f(f(\bot)), \ldots$ with regard to a computational approximation relation defined below. The compactness of *fix* (if *fix*(f) halts then one of its finite approximations halts) will be a simple corollary of this result. However, the proof of the least upper bound theorem is considerably more elegant than most proofs of compactness.

3.1 Computational Approximation

For convenience, throughout this section we will frequently consider terms using a unified representation scheme for terms: A term is either a variable or a compound term $\theta(x_{11}\cdots x_{1k_1}.t_1, \ldots, x_{n1}\cdots x_{nk_n}.t_n)$ where the variables $x_{i1}, \ldots, x_{ik_i}$ are bound in the subterm t_i. For example, the term $\Pi x{:}T_1.T_2$ is represented $\Pi(T_1, x.T_2)$ and the term $\langle t_1, t_2\rangle$ is represented $\langle\rangle(t_1, t_2)$.

Informally speaking, a term t_1 approximates the term t_2 when: if t_1 converges to a canonical form then t_2 converges to a canonical form with the same outermost operator, and the subterms of t_1's canonical form approximate the corresponding subterms of t_2's canonical form. The formal definition appears below and is due to Howe [15].[4] Following Howe, when R is a binary relation on closed terms, I adopt the convention extending R to possibly open terms that if t and t' are possibly open then $t \; R \; t'$ if and only if $\sigma(t) \; R \; \sigma(t')$ for every substitution σ such that $\sigma(t)$ and $\sigma(t')$ are closed.

Definition 3 (Computational Approximation)

- Let R be a binary relation on closed terms and suppose e and e' are closed. Then $e \; C(R) \; e'$ exactly when if $e \Downarrow \theta(\vec{x}_1.t_1, \ldots, \vec{x}_n.t_n)$ then there exists some closed $e'' = \theta(\vec{x}_1.t'_1, \ldots, \vec{x}_n.t'_n)$ such that $e' \Downarrow e''$ and $t_i \; R \; t'_i$.
- $e \leq_0 e'$ whenever e and e' are closed.
- $e \leq_{i+1} e'$ if and only if $e \; C(\leq_i) \; e'$
- $e \leq e'$ if and only if $e \leq_i e'$ for every i

The following are facts about computational approximation that will be used without explicit reference. The first two follow immediately from the definition, the third is easy using determinism (Proposition 1) and the last is proven using Howe's method [15].

Proposition 4

- $\leq$ and $\leq_i$ are reflexive and transitive.
- If $t \mapsto t'$ then $t' \leq t$ and $t' \leq_i t$.
- If $t \mapsto t'$ then $t \leq t'$ and $t \leq_i t'$.
- **(Congruence)** If $e \leq e'$ and $t \leq t'$ then $e[t/x] \leq e'[t'/x]$.

3.2 Finite Approximations

With this notion of computational approximation in hand, we may now show that the terms $\bot, f\,\bot, f(f\,\bot), \ldots$ form a chain of approximations to the term $\mathit{fix}(f)$. Let $\bot$ be the divergent term $\mathit{fix}(\lambda x.x)$. Since $\bot$ never converges, $\bot \leq t$ for any term t. Let f^i be defined as follows:

$$\begin{aligned} f^0 &\stackrel{\text{def}}{=} \bot \\ f^{i+1} &\stackrel{\text{def}}{=} f(f^i) \end{aligned}$$

Certainly $f^0 \leq f^1$, since $f^0 \equiv \bot$. By congruence, $f(f^0) \leq f(f^1)$, and thus $f^1 \leq f^2$. Similarly, $f^i \leq f^{i+1}$ for all i. Thus $f^0, f^1, f^2, \ldots$ forms a chain; I now wish to show that $\mathit{fix}(f)$ is an upper bound of the chain. Certainly $f^0 \leq \mathit{fix}(f)$. Suppose $f^i \leq \mathit{fix}(f)$. By congruence $f(f^i) \leq f(\mathit{fix}(f))$.

[4] Howe's definition actually differs slightly from the one here; he defines $\leq$ as the greatest fixed point of the operator C. It is not difficult to show that the two definitions are equivalent, as long as the computation system is deterministic (Proposition 1). If the computation system is nondeterministic, the definition here fails to be a fixed point, and the more complicated greatest fixed point definition must be employed.

Thus, since $\mathit{fix}(f) \mapsto f(\mathit{fix}(f))$, it follows that $f^{i+1} \equiv f(f^i) \leq f(\mathit{fix}(f)) \leq \mathit{fix}(f)$. By induction it follows that $\mathit{fix}(f)$ is an upper bound of the chain. The following corollary follows from congruence and the definition of approximation:

Corollary 5 If there exists j such that $e[f^j/x]\!\downarrow$ then $e[\mathit{fix}(f)/x]\!\downarrow$. Moreover, the canonical forms of $e[f^j/x]$ and $e[\mathit{fix}(f)/x]$ must have the same outermost operator.

3.3 Least Upper Bound Theorem

In this section I summarize the proof of the least upper bound theorem. The full proof appears in Appendix A. To begin, we need a lemma stating a general property of evaluation. Lemma 6 captures the intuition that closed, noncanonical terms that lie within a term being evaluated are not destructed; they either are moved around unchanged (the lemma's first case) or are evaluated in place with the surrounding term left unchanged (the lemma's second case). The variable x indicates positions where the term of interest is found and, in the second case, the variable y indicates which of those positions, if any, is about to be evaluated.

Lemma 6 If $e_1[t/x] \mapsto e_2$, and $e_1[t/x]$ is closed, and t is closed and noncanonical, then either

- there exists e_2' such that for any closed t', $e_1[t'/x] \mapsto e_2'[t'/x]$, or
- there exist e_1' and t' such that $e_1 \equiv e_1'[x/y]$, $t \mapsto t'$ and for any closed t'', $e_1'[t'', t/x, y] \mapsto e_1'[t'', t'/x, y]$.

It is worthwhile to note that Propositions 1 and 4 and Lemma 6 are the only properties of evaluation used in the proof of the least upper bound theorem, and that these properties are true in computational systems with considerable generality. Consequently, the theorem may be used in a variety of applications beyond the computational system of this paper.

Lemma 7 shows that *fix* terms may be effectively simulated in any particular computation by sufficiently large finite approximations. The lemma is simplified by using computational approximation instead of evaluation for the simulation, which makes it unnecessary to track which of the approximations are unfolded and which are not, an issue that often complicates compactness proofs.

Lemma 7 (Simulation) For all f, e_1 and e_2 (where f is closed and x is the only free variable of e_1), there exist j and e_2' such that if $e_1[\mathit{fix}(f)/x] \mapsto^* e_2$ then $e_2 \equiv e_2'[\mathit{fix}(f)/x]$ and for all $k \geq j$, $e_2'[f^{k-j}/x] \leq e_1[f^k/x]$.

Theorem 8 (Least Upper Bound) For all f, t and e (where f is closed), if $\forall j.\, e[f^j/x] \leq t$, then $e[\mathit{fix}(f)/x] \leq t$.

Proof Sketch

By induction on l that $e[\mathit{fix}(f)/x] \leq_l t$. (The complete proof in Appendix A addresses free variables.) Suppose $e[\mathit{fix}(f)/x]$ evaluates to some canonical form $e'[\mathit{fix}(f)/x]$ (where e' is chosen by Lemma 7). Let e' be of the form $\theta(\vec{x}_1.t_1, \ldots, \vec{x}_n.t_n)$. Using Lemma 7, the assumption $\forall k.\, e[f^k/x] \leq t$, and transitivity, we may show that $e'[f^j/x] \leq t$ for all j. Therefore $t \Downarrow \theta(\vec{x}_1.t'_1, \ldots, \vec{x}_n.t'_n)$ and $t_i[f^j/x] \leq t'_i$ for all j. Now, by induction, $t_i[\mathit{fix}(f)/x] \leq_l t'_i$. Thus $e[\mathit{fix}(f)/x] \leq_{l+1} t$.

There are two easy corollaries to the least upper bound theorem. One is that $\mathit{fix}(f)$ is the least fixed point of f, and the other is compactness.

Corollary 9 (Least Fixed Point) For all closed f and t, if $f(t) \leq t$ then $\mathit{fix}(f) \leq t$.

Proof

Certainly $f^0 \equiv \bot \leq t$. Then $f^1 \equiv f(f^0) \leq f(t) \leq t$. Similarly, by induction, $f^j \leq t$ for any j. Therefore $\mathit{fix}(f) \leq t$ by Theorem 8. □

Corollary 10 (Compactness) If f is closed and $e[\mathit{fix}(f)/x]\downarrow$ then there exists some j such that $e[f^j/x]\downarrow$. Moreover, the canonical forms of $e[\mathit{fix}(f)/x]$ and $e[f^j/x]$ must have the same outermost operator.

Proof

Suppose there does not exist j such that $e[f^j/x]\downarrow$. Then $e[f^j/x] \leq \bot$ for all j. By Theorem 8, $e[\mathit{fix}(f)/x] \leq \bot$. Therefore $e[\mathit{fix}(f)/x]$ does not converge, but this contradicts the assumption,[5] so there must exist j such that $e[f^j/x]\downarrow$. Since $e[f^j/x] \leq e[\mathit{fix}(f)/x]$, the canonical forms of $e[f^j/x]$ and $e[\mathit{fix}(f)/x]$ must have the same outermost operator. □

4 Admissibility

I am now ready to begin specifying some wide classes of types for which the fixpoint principle is valid. First we define admissibility. The simple property of validating the fixpoint principle is too specific to allow any good closure conditions to be shown easily, so we generalize a bit to define admissibility. A type is *admissible* if the upper bound $t[\mathit{fix}(f)]$ of an approximation chain $t[f^0], t[f^1], t[f^2], \ldots$ belongs to the type whenever a cofinite subset of the chain belongs to the type. This is formalized as Definition 12, but first I define some convenient notation.

Notation 11 For any natural number j, the notation $t^{[j]_f}$ means $t[f^j/w]$, and the notation $t^{[\omega]_f}$ means $t[\mathit{fix}(f)/w]$. Also, the f subscript is dropped when the intended term f is unambiguously clear.

[5] Although this proof is non-constructive, a slightly less elegant constructive proof is derivable directly from Lemma 7.

Definition 12 A type T is *admissible* (abbreviated $\mathrm{Adm}(T)$) if:

$$\forall f, t, t'. (\exists j. \forall k \geq j. t^{[k]} = t'^{[k]} \in T) \Rightarrow t^{[\omega]} = t'^{[\omega]} \in T$$

As expected, admissibility is sufficient to guarantee applicability of the fixpoint principle:

Theorem 13 For any T and f, if T is admissible and $f = f' \in \overline{T} \to \overline{T}$ then $\mathit{fix}(f) = \mathit{fix}(f') \in \overline{T}$.

Proof

$\overline{T}$ *type* since $\overline{T} \to \overline{T}$ *type*. Note that $f^j = f'^j \in \overline{T}$ for every j. Suppose $\mathit{fix}(f)\downarrow$. By compactness, $f^j\downarrow$ for some j. Since $f^j = f'^j \in \overline{T}$, it follows that $f'^j\downarrow$ and thus $\mathit{fix}(f')\downarrow$ by Corollary 5. Similarly $\mathit{fix}(f')\downarrow$ implies $\mathit{fix}(f)\downarrow$. It remains to show that $\mathit{fix}(f) = \mathit{fix}(f') \in T$ when $\mathit{fix}(f)\downarrow$. Suppose again that $\mathit{fix}(f)\downarrow$. As before, there exists j such that $f^j\downarrow$ by compactness. Hence $f^j = f'^j \in T$. Since T is admissible, $\mathit{fix}(f) = \mathit{fix}(f') \in T$. □

A number of closure conditions exist on admissible types and are given in Lemma 14. Informally, basic compound types other than dependent products are admissible so long as their component types in positive positions are admissible. Base types—natural numbers, convergence types, and (for this lemma only) equality types—are always admissible. These are essentially the admissible types of Smith [24], except that for a function type to be admissible Smith required that its domain type be admissible.

Lemma 14

- $\mathrm{Adm}(A + B)$ if $\mathrm{Adm}(A)$ and $\mathrm{Adm}(B)$
- $\mathrm{Adm}(\Pi x{:}A.B)$ if $\forall a \in A. \mathrm{Adm}(B[a/x])$
- $\mathrm{Adm}(A \times B)$ if $\mathrm{Adm}(A)$ and $\mathrm{Adm}(B)$
- $\mathrm{Adm}(\mathbb{N})$
- $\mathrm{Adm}(a = a' \ \mathit{in}\ A)$
- $\mathrm{Adm}(\overline{A})$ if $\mathrm{Adm}(A)$
- $\mathrm{Adm}(a\ \mathit{in}!\ A)$

Proof

The proof follows the same lines as Smith's proof, except that handling equality adds a small amount of complication to the proof. I show the function case by way of example.

Let f, t and t' be arbitrary. Suppose j is such that $\forall k \geq j. t^{[k]} = t'^{[k]} \in \Pi x{:}A.B$. I need to show that $t^{[\omega]} = t'^{[\omega]} \in \Pi x{:}A.B$. Since $\Pi x{:}A.B$ is inhabited it is a type. Both $t^{[j]}$ and $t'^{[j]}$ converge to lambda abstractions, so, by Corollary 5, $t^{[\omega]} \Downarrow \lambda x.b$ and $t'^{[\omega]} \Downarrow \lambda x.b'$ for some terms b and b'. Suppose $a = a' \in A$. To get that $b[a/x] = b'[a'/x] \in B[a/x]$ it suffices to show that $t^{[\omega]}a = t'^{[\omega]}a' \in B[a/x]$. Since $\mathrm{Adm}(B[a/x])$, it suffices to show that $\forall k \geq j. t^{[k]}a = t'^{[k]}a' \in B[a/x]$, which follows from the supposition. □

Unfortunately, Lemma 14 can show the admissibility of a product space only if it is *non-dependent.* Dependent products do not have an admissibility condition similar to that of dependent functions. This reason for this is as follows: Admissibility states that a *single fixed type* contains the limit of an approximation chain if it contains a cofinite subset of that chain. For functions, disjoint union, partial types, and non-dependent products it is possible to decompose prospective members in such a way that admissibility may be applied to a single type (such as the type $B[a/x]$ used in the proof of Lemma 14). In contrast, for a dependent product, the right-hand term's desired type depends upon the left-hand term, which is changing at the same time as the right-hand term. Consequently, there is no single type into which to place the right-hand term.

However, understanding the problem with dependent products suggests a solution, to generalize the definition of admissibility to allow the type to vary. This leads to the notion of *predicate-admissibility* that I discuss in the next section.

4.1 Predicate-Admissibility

Definition 15 A type T is *predicate-admissible* for x in S (abbreviated $\mathrm{Adm}(T \mid x : S)$) if:

$$\forall f, t, t', e.\, e^{[\omega]} \in S \wedge (\exists j.\, \forall k \geq j.\, e^{[k]} \in S \wedge t^{[k]} = t'^{[k]} \in T[e^{[k]}/x]) \Rightarrow t^{[\omega]} = t'^{[\omega]} \in T[e^{[\omega]}/x]$$

The term "predicate-admissibility" stems from its similarity to the notion of admissibility of predicates in domain theory (and LCF). If one ignores the inhabiting terms t and t', which may be seen as evidences of the truth of the predicate $T[\,]$, then predicate-admissibility is saying $T[e^{[\omega]}]$ if $T[e^{[k]}]$ for all k greater than some j. This is precisely the notion of admissibility of predicates in domain theory. Indeed, the results here were influenced by the work of Igarashi [16], who established conditions on admissibility of domain-theoretic predicates.

To show the admissibility of a dependent product type, it is sufficient to show predicate-admissibility of the right-hand side (along with admissibility of the left):

Lemma 16 The type $\Sigma x{:}A.B$ is admissible if $\mathrm{Adm}(A)$ and $\mathrm{Adm}(B \mid x : A)$.

Proof

Let f, t and t' be arbitrary. Suppose j is such that $\forall k \geq j.\, t^{[k]} = t'^{[k]} \in \Sigma x{:}A.B$. It is necessary to show that $t^{[\omega]} = t'^{[\omega]} \in \Sigma x{:}A.B$. Since $\Sigma x{:}A.B$ is inhabited it is a type. Both $t^{[j]}$ and $t'^{[j]}$ converge to pairs, so, by Corollary 5, $t^{[\omega]} \Downarrow \langle a, b\rangle$ and $t'^{[\omega]} \Downarrow \langle a', b'\rangle$ for some terms a, b, a' and b'. To get that $a = a' \in A$ it suffices to show that $\pi_1(t^{[\omega]}) = \pi_1(t'^{[\omega]}) \in A$. Since $\mathrm{Adm}(A)$, it suffices to show that $\forall k \geq j.\, \pi_1(t^{[k]}) = \pi_1(t'^{[k]}) \in A$, which follows from the supposition.

To get that $b = b' \in B[a/x]$ (the interesting part), it suffices to show that $\pi_2(t^{[\omega]}) = \pi_2(t'^{[\omega]}) \in B[\pi_1(t^{[\omega]})/x]$. Since $\mathrm{Adm}(B \mid x : A)$, it suffices to show that $\pi_1(t^{[\omega]}) \in A$, which has already been shown, and $\forall k \geq j.\, \pi_1(t^{[k]}) \in A \wedge \pi_2(t^{[k]}) = \pi_2(t'^{[k]}) \in B[\pi_1(t^{[k]})/x]$, which follows from the supposition. □

The conditions for predicate-admissibility are more elaborate, but also more general. I may immediately state conditions for basic types other than functions. Informally, basic compound types other than functions are predicate-admissible so long as their component types are predicate-admissible, and base types are always predicate-admissible. (The proof for these conditions, and all other remaining proofs, appear in Appendix A.)

Lemma 17

- $\mathrm{Adm}(A+B \mid y:S)$ if $\forall s \in S.\,(A+B)[s/y]$ *type* and $\mathrm{Adm}(A \mid y:S)$ and $\mathrm{Adm}(B \mid y:S)$.
- $\mathrm{Adm}(\Sigma x{:}A.B \mid y:S)$ if $\forall s \in S.\,(\Sigma x{:}A.B)[s/y]$ *type* and $\Sigma y{:}S.A$ *type* and $\mathrm{Adm}(A \mid y:S)$ and $\mathrm{Adm}(B[\pi_1(z),\pi_2(z)/y,x] \mid z:(\Sigma y{:}S.A))$
- $\mathrm{Adm}(\mathbb{N} \mid y:S)$
- $\mathrm{Adm}(a_1 = a_2\ in\ A \mid y:S)$ if $\forall s \in S.\,(a_1 = a_2\ in\ A)[s/y]$ *type* and $\mathrm{Adm}(A \mid y:S)$
- $\mathrm{Adm}(\overline{A} \mid y:S)$ if $\forall s \in S.\,\overline{A}[s/y]$ *type* and $\mathrm{Adm}(A \mid y:S)$
- $\mathrm{Adm}(a\ in!\ A \mid y:S)$ if $\forall s \in S.\,(a\ in!\ A)[s/y]$ *type*

Predicate-admissibility of a function type is more complicated because a function argument with the type $A[e^{[\omega]}/x]$ does not necessarily belong to any of the finite approximations $A[e^{[j]}/x]$. To settle this, it is necessary to require a *coadmissibility* condition on the domain type. Then a function type will be predicate-admissible if the domain is weakly coadmissible and the codomain is predicate-admissible.

Definition 18 A type T is *weakly coadmissible* for x in S (abbreviated $\mathrm{WCoAdm}(T \mid x:S)$) if:

$$\begin{aligned}\forall f,t,t',e.\ e^{[\omega]} \in S \wedge (\exists j.\,\forall k \geq j.\ e^{[k]} \in S) \wedge t = t' \in T[e^{[\omega]}/x] \Rightarrow \\ (\exists j.\,\forall k \geq j.\ t = t' \in T[e^{[k]}/x])\end{aligned}$$

A type T is *coadmissible* for x in S (abbreviated $\mathrm{CoAdm}(T \mid x:S)$) if:

$$\begin{aligned}\forall f,t,t',e.\ e^{[\omega]} \in S \wedge (\exists j.\,\forall k \geq j.\ e^{[k]} \in S) \wedge t^{[\omega]} = t'^{[\omega]} \in T[e^{[\omega]}/x] \Rightarrow \\ (\exists j.\,\forall k \geq j.\ t^{[k]} = t'^{[k]} \in T[e^{[k]}/x])\end{aligned}$$

Lemma 19 $\mathrm{Adm}(\Pi x{:}A.B \mid y:S)$ if $\forall s \in S.\,(\Sigma x{:}A.B)[s/y]$ *type* and $\mathrm{WCoAdm}(A \mid y:S)$ and $\forall s \in S, a \in A[s/y].\,\mathrm{Adm}(B[a/x] \mid y:S)$

Clearly coadmissibility implies weak coadmissibility. A general set of conditions listed in Lemma 20 establish weak and full coadmissibility for various types. Weak and full coadmissibility are closed under disjoint union and dependent sum formation, and full coadmissibility is additionally closed under equality-type formation. I use both notions of coadmissibility, rather than just adopting one or the other, because full coadmissibility is needed for equality types but under certain circumstances weak coadmissibility is easier to show (Proposition 21 below).

Lemma 20

- $A + B$ is (weakly) coadmissible for y in S if $\forall s \in S.\, (A+B)[s/y]$ *type* and A and B are (weakly) coadmissible for y in S
- $\mathrm{WCoAdm}(\Sigma x{:}A.B \mid y : S)$ if $\forall s \in S.\, (\Sigma x{:}A.B)[s/y]$ *type* and $\mathrm{WCoAdm}(A \mid y : S)$ and $\forall s \in S, a \in A[s/y].\, \mathrm{WCoAdm}(B[a/x] \mid y : S)$
- $\mathrm{CoAdm}(\Sigma x{:}A.B \mid y : S)$ if $\forall s \in S.\, (\Sigma x{:}A.B)[s/y]$ *type* and $\Sigma y{:}S.A$ *type* and $\mathrm{CoAdm}(A \mid y : S)$ and $\mathrm{CoAdm}(B[\pi_1(z), \pi_2(z)/y, x] \mid z : (\Sigma y{:}S.A))$
- $\mathbb{N}$ is strongly or weakly coadmissible for y in any S
- $\mathrm{CoAdm}(a_1 = a_2 \text{ in } A \mid y : S)$ if $\forall s \in S.\, (a_1 = a_2 \text{ in } A)[s/y]$ *type* and $\mathrm{CoAdm}(A \mid y : S)$
- $\overline{A}$ is (weakly) coadmissible for y in S if $\forall s \in S.\, \overline{A}[s/y]$ *type* and A is (weakly) coadmissible for y is S
- $a \text{ in! } A$ is strongly or weakly coadmissible for y in S if $\forall s \in S.\, (a \text{ in! } A)[s/y]$ *type*

When T does not depend upon S, predicate-admissibility and weak coadmissibility become easier to show:

Proposition 21 Suppose x does not appear free in T. Then:

- $\mathrm{Adm}(T)$ if $\mathrm{Adm}(T \mid x : S)$ and S is inhabited
- $\mathrm{Adm}(T \mid x : S)$ if $\mathrm{Adm}(T)$
- $\mathrm{WCoAdm}(T \mid x : S)$

There remains one more result related to predicate-admissibility. Suppose one wishes to show $\mathrm{Adm}(T|x : S)$ where T depends upon x. There are two ways that x may be used in T. First, T might contain an equality type where x appears in one or both of the equands. In that case, predicate-admissibility can be shown with the tools discussed above. Second, T may be an expression that computes a type from x. In this case, T can be simplified using untyped reasoning [15], but another tool will be needed if T performs any case analysis.

Lemma 22 Consider a type $\mathit{case}(d, x.A, x.B)$ that depends upon y from S. Suppose there exist T_1 and T_2 such that:

- $\forall s \in S.\, d[s/y] \in (T_1 + T_2)[s/y]$
- $\forall s \in S, t \in T_1[s/y].\, A[s, t/y, x]$ *type*
- $\forall s \in S, t \in T_2[s/y].\, B[s, t/y, x]$ *type*
- $\Sigma y{:}S.T_1$ *type* and $\Sigma y{:}S.T_2$ *type*

Then the following are the case:

- $\mathrm{Adm}(case(d, x.A, x.B) \mid y : S)$ if $\mathrm{Adm}(A[\pi_1(z), \pi_2(z)/y, x] \mid z : (\Sigma y{:}S.T_1))$ and $\mathrm{Adm}(B[\pi_1(z), \pi_2(z)/y, x] \mid z : (\Sigma y{:}S.T_2))$
- $\mathrm{WCoAdm}(case(d, x.A, x.B) \mid y : S)$ if $\mathrm{WCoAdm}(A[\pi_1(z), \pi_2(z)/y, x] \mid z : (\Sigma y{:}S.T_1))$ and $\mathrm{WCoAdm}(B[\pi_1(z), \pi_2(z)/y, x] \mid z : (\Sigma y{:}S.T_2))$
- $\mathrm{CoAdm}(case(d, x.A, x.B) \mid y : S)$ if $\mathrm{CoAdm}(A[\pi_1(z), \pi_2(z)/y, x] \mid z : (\Sigma y{:}S.T_1))$ and $\mathrm{CoAdm}(B[\pi_1(z), \pi_2(z)/y, x] \mid z : (\Sigma y{:}S.T_2))$

4.2 Monotonicity

In some cases a very simple device may be used to show admissibility. We say that a type is monotone if it respects computational approximation, and it is easy to show that all monotone types are admissible.

Definition 23 A type T is *monotone* (abbreviated $\mathrm{Mono}(t)$) if $t = t' \in T$ whenever $t \in T$ and $t \leq t'$.

Lemma 24 All monotone types are admissible.

Proof

Let f, t and t' be arbitrary and suppose there exists j such that $t^{[j]} = t'^{[j]} \in T$. Since $t^{[j]} \leq t^{[\omega]}$ and $t'^{[j]} \leq t'^{[\omega]}$, it follows that $t^{[j]} = t^{[\omega]} \in T$ and $t'^{[j]} = t'^{[\omega]} \in T$. The result follows directly. $\Box$

All type constructors are monotone except universes and partial types, which are never monotone. The proof of this fact is easy [15].

Proposition 25

- $\mathrm{Mono}(A + B)$ if $\mathrm{Mono}(A)$ and $\mathrm{Mono}(B)$
- $\mathrm{Mono}(\Pi x{:}A.B)$ if $\mathrm{Mono}(A)$ and $\forall a \in A.\, \mathrm{Mono}(B[a/x])$
- $\mathrm{Mono}(\Sigma x{:}A.B)$ if $\mathrm{Mono}(A)$ and $\forall a \in A.\, \mathrm{Mono}(B[a/x])$
- $\mathrm{Mono}(\mathbb{N})$, $\mathrm{Mono}(a_1 = a_2 \in A)$ and $\mathrm{Mono}(a\ in!\ A)$

4.3 Set and Quotient Types

In addition to the type constructors discussed so far, the Nuprl type theory also contains two fairly novel types, Constable's set and quotient types [6]. These types allow the use of logical predicates

$$
\begin{array}{ll}
a = a' \in \{x : A \mid B\} & \text{iff } \{x : A \mid B\}\ \mathit{type} \wedge a = a' \in A \wedge \exists b.\, b \in B[a/x] \\
a = a' \in xy{:}A/\!/B & \text{iff } (xy{:}A/\!/B)\ \mathit{type} \wedge a \in A \wedge a' \in A \wedge \exists b.\, b \in B[a, a'/x, y]
\end{array}
$$

For $T \simeq T'$ iff $T = T'$, and for $T \simeq T'$ iff $T = T' \in \mathbb{U}_i$:

$$
\begin{array}{ll}
\{x : A \mid B\} \simeq \{x : A' \mid B'\} & \text{iff } A \simeq A' \wedge \forall a = a' \in A.\, B[a/x] \simeq B[a'/x] \wedge \\
& \quad \forall a = a' \in A.\, B'[a/x] \simeq B'[a'/x] \wedge \\
& \quad (B \text{ if and only if } B') \\
& \quad \exists t.\, t \in \Pi x{:}A.\, B \to B' \wedge \exists t.\, t \in \Pi x{:}A.\, B' \to B \\
(xy{:}A/\!/B) \simeq (xy{:}A'/\!/B') & \text{iff } A \simeq A' \wedge \\
& \quad \forall a_1 = a_1' \in A.\, \forall a_2 = a_2' \in A.\, B[a_1, a_2/x, y] \simeq B[a_1', a_2'/x, y] \wedge \\
& \quad \forall a_1 = a_1' \in A.\, \forall a_2 = a_2' \in A.\, B'[a_1, a_2/x, y] \simeq B'[a_1', a_2'/x, y] \wedge \\
& \quad (B \text{ if and only if } B') \\
& \quad \exists t.\, t \in \Pi x{:}A.\, \Pi y{:}A.\, B \to B' \wedge \exists t.\, t \in \Pi x{:}A.\, \Pi y{:}A.\, B' \to B \wedge \\
& \quad (\text{reflexivity}) \\
& \quad \exists t.\, t \in \Pi x{:}A.\, B[x/y] \wedge \\
& \quad (\text{symmetry}) \\
& \quad \exists t.\, t \in \Pi x{:}A.\, \Pi y{:}A.\, B \to B[y, x/x, y] \wedge \\
& \quad (\text{transitivity}) \\
& \quad \exists t.\, t \in \Pi x{:}A.\, \Pi y{:}A.\, \Pi z{:}A.\, B \to B[y, z/x, y] \to B[z/y]
\end{array}
$$

Figure 4: Set and Quotient Type Definitions

(encoded as types using the propositions-as-types principle [14]) to refine or coarsen types in various ways.

The *set type* $\{x{:}T \mid P\}$ is the subtype of T that contains all $t \in T$ such that $P[t/x]$ is inhabited (*i.e.,* such that the proposition corresponding to $P[t/x]$ is true). The *quotient type* $xy{:}T/\!/E[x, y]$ (when $E[-, -]$ corresponds to an equivalence relation on T) is the supertype of T that coarsens the equality on T as follows: $t_1 = t_2 \in xy{:}T/\!/E$ if and only if $t_1, t_2 \in T$ and $E[t_1, t_2/x, y]$ is inhabited (*i.e.,* true). The set and quotient types are defined formally in Figure 4.

The set type $\{x{:}A \mid B[x]\}$ is much like the dependent product type $\Sigma x{:}A.B[x]$ in that both provide a member a of A such that $B[a]$ is inhabited, but differ in that the dependent product provides that inhabitant and the set type suppresses it. Given this parallel between the dependent product and set types, it is natural to expect that set types would have a similar admissibility rule: that $\{x : A \mid B\}$ if A is admissible and B is predicate-admissible for x in A. Somewhat surprisingly, this turns out not to be the case. Suppose that $t^{[k]} \in \{x : A \mid B\}$ for all $k \geq j$. Then for every $k \geq j$, there exists some term $b_k \in B[t^{[k]}/x]$. We would like it to follow by predicate-admissibility that there exists $b_\omega \in B[t^{[\omega]}/x]$, but it does not. The problem is that each b_k can be a completely different term, and predicate-admissibility applies only when each b_k is of the form $b[f^k/w]$ for a single fixed b.

Intuitively, the desired rule fails because the set type $\{x : A \mid B\}$ suppresses the computational content of B and therefore B can be inhabited *non-uniformly,* by unrelated terms for related members of A. In contrast, if the chain $t^{[j]}, t^{[j+1]}, t^{[j+2]}, \ldots$ belongs to $\Sigma x{:}A.B$, then the chain $\pi_2(t)^{[j]}, \pi_2(t)^{[j+1]}, \pi_2(t)^{[j+2]}, \ldots$ uniformly inhabits B.

For a concrete example, consider:

$$
\begin{array}{rcl}
T & \stackrel{\text{def}}{=} & \{g : \mathbb{N} \to \overline{\mathbb{N}} \mid \exists n{:}\mathbb{N}.\, (g\, n\ in!\ \mathbb{N}) \to \mathit{Void}\} \\
f & \stackrel{\text{def}}{=} & \lambda h.\, \lambda x.\ \mathit{if}\ x = 0\ \mathit{then}\ 0\ \mathit{else}\ h(x-1) \\
t & \stackrel{\text{def}}{=} & \lambda y.\, wy
\end{array}
$$

The type T is not admissible: For all k, $(t^{[k]_f})k$ diverges, so $t^{[k]_f} \in T$; but $t^{[\omega]_f}$ converges for all arguments, so $t^{[\omega]_f} \notin T$. However, $\exists n{:}\mathbb{N}.\, (gn\ in!\ \mathbb{N}) \to \mathit{Void}$ is predicate-admissible for g in $\mathbb{N} \to \overline{\mathbb{N}}$. The problem is that the inhabiting integers are not related by computational approximation; that is, they are not uniform.

To show a set type admissible, we need to be able to show that the selection predicate can be inhabited uniformly:

Lemma 26 The type $\{x : A \mid B\}$ is admissible if:

- $\mathrm{Adm}(A)$, and
- $\mathrm{Adm}(B \mid x : A)$, and
- there exists b such that $b[a/x] \in B[a/x]$ whenever $a \in A$ and $\exists b'.\, b' \in B[a/x]$.

We may give a similar predicate-admissibility condition:

Lemma 27 $\mathrm{Adm}(\{x : A \mid B\} \mid y : S)$ if $\forall s \in S.\ (\{x : A \mid B\})[s/y]$ *type* and $\Sigma y{:}S.A$ *type* and $\mathrm{Adm}(A \mid y : S)$ and $\mathrm{Adm}(B[\pi_1(z), \pi_2(z)/y, x] \mid z : \Sigma y{:}S.A)$, and there exists b such that $b[a, s/x, y] \in B[a, s/x, y]$ whenever $s \in S$ and $a \in A[s/y]$ and $\exists b'.\, b' \in B[a, s/x, y]$

Coadmissibility and monotonicity work on single terms, not chains, so the uniformity issue does not arise, resulting in conditions fairly similar to those for dependent products:

Lemma 28

- $\mathrm{WCoAdm}(\{x : A \mid B\} \mid y : S)$ if $\forall s \in S.\, \{x : A \mid B\}[s/y]$ *type* and $\mathrm{WCoAdm}(A \mid y : S)$ and $\forall s \in S, a \in A[s/y].\, \mathrm{WCoAdm}(B[a/x] \mid y : S)$
- $\mathrm{CoAdm}(\{x : A \mid B\} \mid y : S)$ if $\forall s \in S.\, \{x : A \mid B\}[s/y]$ *type* and $\Sigma y{:}S.A$ *type* and $\mathrm{CoAdm}(A \mid y : S)$ and $\mathrm{WCoAdm}(B[\pi_1(z), \pi_2(z)/y, x] \mid z : (\Sigma y{:}S.A))$
- $\mathrm{Mono}(\{x : A \mid B\})$ if $\mathrm{Mono}(A)$

The conditions for quotient types are similar to those for set types:

Lemma 29

- $\mathrm{Adm}(xy{:}A/\!/B)$ if $\mathrm{Adm}(A)$ and $\mathrm{Adm}(B[\pi_1(z), \pi_2(z)/x, y] \,|\, z : A \times A)$, and there exists b such that $b[a, a'/x, y] \in B[a, a'/x, y]$ whenever $a \in A$ and $a' \in A$ and $\exists b'.\, b' \in B[a, a'/x, y]$.
- $\mathrm{Adm}(xy{:}A/\!/B \,|\, z : S)$ if $\forall s \in S.\,(xy{:}A/\!/B)[s/z]$ *type* and $\Sigma z{:}S.(A \times A)$ *type* and $\mathrm{Adm}(A \,|\, z : S)$ and $\mathrm{Adm}(B[\pi_1(z'), \pi_1(\pi_2(z')), \pi_2(\pi_2(z'))/z, x, y] \,|\, z' : \Sigma z{:}S.(A \times A))$, and there exists b such that $b[a, a', s/x, y, z] \in B[a, a', s/x, y, z]$ whenever $s \in S$ and $a \in A[s/z]$ and $a' \in A[s/z]$ and $\exists b'.\, b' \in B[a, a', s/x, y, z]$.
- $\mathrm{WCoAdm}(xy{:}A/\!/B \mid z : S)$ if $\forall s \in S.\,(xy{:}A/\!/B)[s/z]$ *type* and $\mathrm{WCoAdm}(A \mid z : S)$ and $\forall s \in S, a \in A[s/z], a' \in A[s/z].\, \mathrm{WCoAdm}(B[a, a'/x, y] \,|\, z : S)$
- $\mathrm{CoAdm}(xy{:}A/\!/B|z : S)$ if $\forall s \in S.\,(xy{:}A/\!/B)[s/z]$*type* and $\Sigma z{:}S.(A\times A)$ *type* and $\mathrm{CoAdm}(A|z : S)$ and $\mathrm{CoAdm}(B[\pi_1(z'), \pi_1(\pi_2(z')), \pi_2(\pi_2(z'))/z, x, y] \,|\, z' : \Sigma z{:}S.(A \times A))$
- $\mathrm{Mono}(xy{:}A/\!/B)$ if $\mathrm{Mono}(A)$

4.4 Summary

Figure 5 provides a summary of the basic admissibility results of this chapter. It is worthwhile to note that all these results are proved constructively, with the exception of (weak and full) coadmissibility of partial types. The following theorem shows that the proofs of coadmissibility of partial types are necessarily classical; if a constructive proof existed then one could extract an algorithm meeting the theorem's specification, which can be used to solve the halting problem.

Theorem 30 There does not exist an algorithm that computes an integer j such that $\forall k \geq j.\, t = t' \in \overline{T}[e^{[k]}/x]$, when given S, T, f, t, t', e and i such that:

- $\forall s \in S.\, \overline{T}[s/x]$ *type*
- $\mathrm{CoAdm}(T \mid x : S)$
- $e^{[\omega]} \in S$
- $\forall k \geq i.\, e^{[k]} \in S$
- $t = t' \in \overline{T}[e^{[\omega]}/x]$

Recall the inadmissible type T from Theorem 2. That type fails the predicate-admissibility condition because of the negative appearance of a function type, which could not be shown weakly coadmissible, and it fails the monotonicity condition because it contains the partial type $\overline{\mathbb{N}}$.

5 Conclusions

An interesting avenue for future investigation would be to find some negative results characterizing inadmissible types. Such negative results would be particularly interesting if they could be given

	For $T \equiv$		
	$A + B$	$\Pi x{:}A.B$	$\Sigma x{:}A.B$
$\mathrm{Adm}(T)$ if T *type* and	$\mathrm{Adm}(A) \wedge \mathrm{Adm}(B)$	$\forall a \in A.\, \mathrm{Adm}(B[a/x])$	$\mathrm{Adm}(A) \wedge \mathrm{Adm}(B \mid x : A)$
$\mathrm{Adm}(T \mid y : S)$ if $\forall s \in S.\, T[s/y]$ *type* and	$\mathrm{Adm}(A \mid y : S) \wedge$ $\mathrm{Adm}(B \mid y : S)$	$\mathrm{WCoAdm}(A \mid y : S) \wedge$ $\forall s \in S, a \in A[s/y].$ $\mathrm{Adm}(B[a/x] \mid y : S)$	$\mathrm{Adm}(A \mid y : S) \wedge$ $\mathrm{Adm}(B[\pi_1(z), \pi_2(z)/y, x]$ $\mid z : (\Sigma y : S.A)) \wedge$ $\Sigma y{:}S.A$ *type*
$\mathrm{WCoAdm}(T \mid y : S)$ if $\forall s \in S.\, T[s/y]$ *type* and	$\mathrm{WCoAdm}(A \mid y : S) \wedge$ $\mathrm{WCoAdm}(B \mid y : S)$	—	$\mathrm{WCoAdm}(A \mid y : S) \wedge$ $\forall s \in S, a \in A[s/y].$ $\mathrm{WCoAdm}(B[a/x] \mid y : S)$
$\mathrm{CoAdm}(T \mid y : S)$ if $\forall s \in S.\, T[s/y]$ *type* and	$\mathrm{CoAdm}(A \mid y : S) \wedge$ $\mathrm{CoAdm}(B \mid y : S)$	—	$\mathrm{CoAdm}(A \mid y : S) \wedge$ $\mathrm{CoAdm}(B[\pi_1(z), \pi_2(z)/y, x]$ $\mid z : (\Sigma y : S.A)) \wedge$ $\Sigma y{:}S.A$ *type*
$\mathrm{Mono}(T)$ if	$\mathrm{Mono}(A) \wedge \mathrm{Mono}(B)$	$\mathrm{Mono}(A) \wedge$ $\forall a \in A.\, \mathrm{Mono}(B[a/x])$	$\mathrm{Mono}(A) \wedge$ $\forall a \in A.\, \mathrm{Mono}(B[a/x])$

	For $T \equiv$		
	$\mathbb{N}, a\ in!\ A$	$a_1 = a_2\ in\ A$	$\overline{A}$
$\mathrm{Adm}(T)$ if T *type* and	yes	yes	$\mathrm{Adm}(A)$
$\mathrm{Adm}(T \mid y : S)$ if $\forall s \in S.\, T[s/y]$ *type* and	yes	$\mathrm{Adm}(A \mid y : S)$	$\mathrm{Adm}(A \mid y : S)$
$\mathrm{WCoAdm}(T \mid y : S)$ if $\forall s \in S.\, T[s/y]$ *type* and	yes	$\mathrm{CoAdm}(A \mid y : S)$	$\mathrm{WCoAdm}(A \mid y : S)$
$\mathrm{CoAdm}(T \mid y : S)$ if $\forall s \in S.\, T[s/y]$ *type* and	yes	$\mathrm{CoAdm}(A \mid y : S)$	$\mathrm{CoAdm}(A \mid y : S)$
$\mathrm{Mono}(T)$ if	yes	yes	—

Figure 5: Admissibility, coadmissibility and monotonicity conditions

a syntactic character, like the results of this chapter. Along these lines, it would be interesting to find whether the inability to show coadmissibility of function types represents a weakness of this proof technique or an inherent limitation.

The results presented above provide *metatheoretical* justification for the fixpoint principle over many types. In order for these results to be useful in theorem proving, they must be introduced into the logic. One way to do this, and the way it is done in my implementation of partial types in the Nuprl proof assistant [11], is to introduce types to represent the assertions $\mathrm{Adm}(T)$, $\mathrm{Adm}(T \mid x : S)$, etc., that are inhabited exactly when the underlying assertion is true (in much that same way as the equality type is inhabited exactly when the equands are equal), and to add rules relating to these types that correspond to the lemmas of Section 4. This brings the tools into the system in a semantically justifiable way, but it is unpleasant in that it leads to a proliferation of new types and inference rules stemming from discoveries outside the logic. It would be preferable to deal with admissibility within the logic. A theory with intensional reasoning principles, such as the one proposed in Constable and Crary [7], would allow reasoning about computation internally. Then these results could be proved within the theory and the only extra rule that would be required would be a single rule relating admissibility to the the fixpoint principle.

However they are placed into the logic, these results allow for recursive computation on a wide variety of types. This make partial types and fixpoint induction a useful tool in type-theoretic theorem provers. It also makes it possible to study many recursive programs that used to be barred from the logic because they could not be typed.

References

[1] Stuart Allen. A non-type-theoretic definition of Martin-Löf's types. In *Second IEEE Symposium on Logic in Computer Science*, pages 215–221, Ithaca, New York, June 1987.

[2] Philippe Audebaud. Partial objects in the calculus of constructions. In *Sixth IEEE Symposium on Logic in Computer Science*, pages 86–95, Amsterdam, July 1991.

[3] Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Jean-Christophe Filliâtre, Eduardo Giménez, Hugo Herbelin, Gérard Huet, César Muñoz, Chetan Murthy, Catherine Parent, Christine Paulin-Mohring, Amokrane Saïbi, and Benjamin Werner. *The Coq Proof Assistant Reference Manual.* INRIA-Rocquencourt, CNRS and ENS Lyon, 1996.

[4] David A. Basin. An environment for automated reasoning about partial functions. In *Ninth International Conference on Automated Deduction*, volume 310 of *Lecture Notes in Computer Science.* Springer-Verlag, 1988.

[5] R.L. Constable, S.F. Allen, H.M. Bromley, W.R. Cleaveland, J.F. Cremer, R.W. Harper, D.J. Howe, T.B. Knoblock, N.P. Mendler, P. Panangaden, J.T. Sasaki, and S.F. Smith. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice-Hall, 1986.

[6] Robert L. Constable. Constructive mathematics as a programming logic I: Some principles of theory. In *Topics in the Theory of Computation*, volume 24 of *Annals of Discrete Mathematics*, pages 21–37. Elsevier, 1985. Selected papers of the International Conference on Foundations of Computation Theory 1983.

[7] Robert L. Constable and Karl Crary. Computational complexity and induction for partial computable functions in type theory. Technical report, Department of Computer Science, Cornell University, 1997.

[8] Robert L. Constable and Scott Fraser Smith. Partial objects in constructive type theory. In *Second IEEE Symposium on Logic in Computer Science*, pages 183–193, Ithaca, New York, June 1987.

[9] Robert L. Constable and Scott Fraser Smith. Computational foundations of basic recursive function theory. In *Third IEEE Symposium on Logic in Computer Science*, pages 360–371, Edinburgh, Scotland, July 1988.

[10] Thierry Coquand and Gérard Huet. The calculus of constructions. *Information and Computation*, 76:95–120, 1988.

[11] Karl Crary. *Type-Theoretic Methodology for Practical Programming Languages.* PhD thesis, Department of Computer Science, Cornell University, Ithaca, New York, August 1998.

[12] Michael J. Gordon, Arthur J. Milner, and Christopher P. Wadsworth. *Edinburgh LCF: A Mechanised Logic of Computation*, volume 78 of *Lecture Notes in Computer Science*. Springer-Verlag, 1979.

[13] Robert Harper. Constructing type systems over an operational semantics. *Journal of Symbolic Computation*, 14:71–84, 1992.

[14] W. Howard. The formulas-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism*, pages 479–490. Academic Press, 1980.

[15] Douglas J. Howe. Equality in lazy computation systems. In *Fourth IEEE Symposium on Logic in Computer Science*, 1989.

[16] Shigeru Igarashi. Admissibility of fixed-point induction in first-order logic of typed theories. Technical Report AIM-168, Computer Science Department, Stanford University, May 1972.

[17] Paul Bernard Jackson. *Enhancing the Nuprl Proof Development System and Applying it to Computational Abstract Algebra*. PhD thesis, Department of Computer Science, Cornell University, Ithaca, New York, January 1995.

[18] David MacQueen. Using dependent types to express modular structure. In *Thirteenth ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, pages 277–286, St. Petersburg Beach, Florida, January 1986.

[19] Per Martin-Löf. Constructive mathematics and computer programming. In *Sixth International Congress of Logic, Methodology and Philosophy of Science*, volume 104 of *Studies in Logic and the Foundations of Mathematics*, pages 153–175. North-Holland, 1982.

[20] Benjamin C. Pierce and David N. Turner. Simple type-theoretic foundations for object-oriented programming. *Journal of Functional Programming*, 4(2):207–247, April 1994.

[21] Dana Scott. Outline of a mathematical theory of computation. In *Fourth Princeton Conference on Information Sciences and Systems*, pages 169–176, 1970.

[22] Dana Scott. Lattice theoretic models for various type-free calculi. In *Fourth International Congress of Logic, Methodology and Philosophy of Science*. North-Holland, 1972.

[23] Scott F. Smith. Hybrid partial-total type theory. *International Journal of Foundations of Computer Science*, 6:235–263, 1995.

[24] Scott Fraser Smith. *Partial Objects in Type Theory*. PhD thesis, Department of Computer Science, Cornell University, Ithaca, New York, January 1989.

A Proofs

Theorem 2 There exist inadmissible types.

Proof

This example is due to Smith [24]. Let the type T and the function f be defined as follows:

$$\begin{array}{lcl} T & \stackrel{\text{def}}{=} & \Sigma h{:}(\mathbb{N} \to \overline{\mathbb{N}}).\, ((\Pi x{:}\mathbb{N}.\, h\, x\ in!\ \mathbb{N}) \to \mathit{Void}) \\ f & \stackrel{\text{def}}{=} & \lambda p.\langle g, \lambda y.\, \star\rangle \\ g & \stackrel{\text{def}}{=} & \lambda x.\, \mathit{if}\ x = 0\ \mathit{then}\ 0\ \mathit{else}\ \pi_1(p)(x-1) \end{array}$$

It is easy to verify that $\overline{T}$ *type*. We wish to show that f has type $\overline{T} \to \overline{T}$. Suppose $t = t' \in \overline{T}$. We need $g[t/p] = g[t'/p] \in \mathbb{N} \to \overline{\mathbb{N}}$ and $\lambda y.\, \star \in (\Pi x{:}\mathbb{N}.\, (g[t/p])x\ in!\ \mathbb{N}) \to \mathit{Void}$. The former is easily shown; to show the latter, I assume that $(g[t/p])n$ converges for every natural number n and draw a contradiction. It follows that $\Pi x{:}\mathbb{N}.\, (g[t/p])x\ in!\ \mathbb{N}$ is empty and $\lambda y.\, \star$ is vacuously a function from any empty type to *Void*. Suppose $(g[t/p])n$ converges for every natural number n. Then the term $\mathit{if}\ n = 0\ \mathit{then}\ 0\ \mathit{else}\ \pi_1(t)(n-1)$ also converges for every natural number n. It follows that $t{\downarrow}$, since $\pi_1(t)(0){\downarrow}$, and hence $t \in T$. Thus $(\Pi x{:}\mathbb{N}.\, \pi_1(t)(x)\ in!\ \mathbb{N}) \to \mathit{Void}$ is inhabited (by $\pi_2(t)$) and consequently it cannot be the case that $\pi_1(t)(n){\downarrow}$ for every natural number n. But this is a contradiction since $\pi_1(t)(n-1){\downarrow}$ for every $n > 1$. Therefore $f \in \overline{T} \to \overline{T}$.

However, it is not the case that $\mathit{fix}(f) \in \overline{T}$. Suppose $\mathit{fix}(f) \in \overline{T}$. Then $\mathit{fix}(f) \in T$ since $\mathit{fix}(f)$ converges (in two steps). Thus $\pi_2(\mathit{fix}(f)) \in (\Pi x{:}\mathbb{N}.\, \pi_1(\mathit{fix}(f))(x)\ in!\ \mathbb{N}) \to \mathit{Void}$, which implies that $\pi_1(\mathit{fix}(f))$ is not total on $\mathbb{N}$, but it is easy to show by induction that $\pi_1(\mathit{fix}(f))$ is in fact total (on $\mathbb{N}$). Therefore $\mathit{fix}(f) \notin \overline{T}$ and hence T is not admissible. □

Lemma 6 If $e_1[t/x] \mapsto e_2$, and $e_1[t/x]$ is closed, and t is closed and noncanonical, then either

- there exists e_2' such that for any closed t', $e_1[t'/x] \mapsto e_2'[t'/x]$, or
- there exist e_1' and t' such that $e_1 \equiv e_1'[x/y]$, $t \mapsto t'$ and for any closed t'', $e_1'[t'', t/x, y] \mapsto e_1'[t'', t'/x, y]$.

Proof

Suppose $e_1 \equiv x$. Then $t \equiv e_1[t/x] \mapsto e_2$. Let $e_1' = y$ and $t' = e_2$. Then $e_1'[t'', t/x, y] \equiv t \mapsto t' \equiv e_1'[t'', t'/x, y]$. The remaining cases are by induction on the derivation of $e_1[t/x] \mapsto e_2$. I show the lambda rules; the other cases are similar.

Suppose the rule used is:

$$\frac{}{(\lambda z.b)a \mapsto b[a/z]}$$

The term $e_1[t/x]$ must have the form of a lambda abstraction applied to an argument. Thus e_1 must be of the form $(\lambda z.b)a$, since $e_1 \equiv x$ is already handled and $e_1 \equiv x\, a$ is impossible because t is noncanonical. Let $e_2' = b[a/z]$ and suppose t' is closed. Then:

$$\begin{array}{rcll} e_1[t'/x] & \equiv & (\lambda z.b[t'/x])(a[t'/x]) & \\ & \mapsto & b[t'/x](a[t'/x]/z) & \\ & \equiv & b[a/z][t'/x] & \text{(since } t' \text{ is closed)} \\ & \equiv & e_2'[t'/x] & \end{array}$$

Suppose the rule used is:

$$\frac{f \mapsto f'}{f\,a \mapsto f'\,a}$$

Then it must be the case that e_1 is of the form $f_1\,a$ (since $e_1 \equiv x$ is already handled) and $f_1[t/x] \mapsto f_2$ (for some f_2). Hence the induction hypothesis holds for f_1. Suppose the first case holds: there exists f_2' such that for any closed t', $f_1[t'/x] \mapsto f_2'[t'/x]$. Let $e_2' = f_2'\,a$ and suppose t' is closed. Then:

$$\begin{array}{rcl} e_1[t'/x] & \equiv & (f_1[t'/x])(a[t'/x]) \\ & \mapsto & (f_2'[t'/x])(a[t'/x]) \\ & \equiv & e_2'[t'/x] \end{array}$$

Suppose the second case holds: there exist f_1' and t' such that $f_1 \equiv f_1'[x/y]$, $t \mapsto t'$ and for any closed t'', $f_1'[t'', t/x, y] \mapsto f_1'[t'', t'/x, y]$. Let $e_1' = f_1'\,a$ and suppose t'' is closed. Then:

$$\begin{array}{rcll} e_1'[t'', t/x, y] & \equiv & (f_1'[t'', t/x, y])(a[t'', t/x, y]) & \\ & \equiv & (f_1'[t'', t/x, y])(a[t'', t'/x, y]) & \text{(since } y \text{ is not free in } a) \\ & \mapsto & (f_1'[t'', t'/x, y])(a[t'', t'/x, y]) & \\ & \equiv & e_1'[t'', t'/x, y] & \end{array}$$

□

Lemma 7 For all f, e_1 and e_2 (where f is closed and x is the only free variable of e_1), there exist j and e_2' such that if $e_1[\mathit{fix}(f)/x] \mapsto^* e_2$ then $e_2 \equiv e_2'[\mathit{fix}(f)/x]$ and for all $k \geq j$, $e_2'[f^{k-j}/x] \leq e_1[f^k/x]$.

Proof

I show the lemma for evaluations of length exactly one. The result then follows by induction on the length of the evaluation sequence, summing the numbers j.

Use Lemma 6. Suppose the first case holds: there exists e_2' such that for any closed t, $e_1[t/x] \mapsto e_2'[t/x]$. Then $e_1[\mathit{fix}(f)/x] \mapsto e_2'[\mathit{fix}(f)/x]$ and, for any k, $e_1[f^k/x] \mapsto e_2'[f^k/x]$. Thus $e_2'[f^{k-0}/x] \leq e_1[f^k/x]$. Suppose the second case holds: there exists e_1' such that $e_1 \equiv e_1'[x/y]$, and for any closed t, $e_1'[t, \mathit{fix}(f)/x, y] \mapsto e_1'[t, f(\mathit{fix}(f))/x, y]$. Let $e_2' = e_1'[f\,x/y]$. Then $e_1[\mathit{fix}(f)/x] \mapsto e_2'[\mathit{fix}(f)/x]$. Suppose $k \geq 1$, then $e_2'[f^{k-1}/x] \equiv e_1'[f^{k-1}, f^k/x, y] \leq e_1'[f^k, f^k/x, y] \equiv e_1[f^k/x]$. □

Theorem 8 For all f, t and e (where f is closed), if $\forall j.\, e[f^j/x] \leq t$, then $e[\mathit{fix}(f)/x] \leq t$.

Proof

By induction on l that for all f, t and e (where f is closed), $(\forall j.\, e[f^j/x] \leq t) \Rightarrow e[\mathit{fix}(f)/x] \leq_l t$. The result follows by the definition of $\leq$. The basis is trivial.

Assume the induction hypothesis for l and $\forall j.\, e[f^j/x] \leq t$. Let σ be a substitution such that $\sigma(e[\mathit{fix}(f)/x])$ and $\sigma(t)$ are closed and suppose, without loss of generality, that σ does not substitute for x. Then $\sigma(e[\mathit{fix}(f)/x]) \equiv \sigma(e)[\mathit{fix}(f)/x]$, $\sigma(e[f^j/x]) \equiv \sigma(e)[f^j/x]$ (for any j), and the only free variable of $\sigma(e)$ is x. Suppose $\sigma(e[\mathit{fix}(f)/x]) \Downarrow e'$. By Lemma 7, $e' \equiv e''[\mathit{fix}(f)/x]$ and, for some j and all $k \geq j$, $e''[f^{k-j}/x] \leq \sigma(e)[f^k/x]$. Then, by assumption and transitivity,

$\forall k \geq j.\, e''[f^{k-j}/x] \leq \sigma(t)$. Therefore, changing variables to replace $k-j$ with k, $e''[f^k/x] \leq \sigma(t)$ for any k.

Let $e'' = \theta(\vec{x}_1.t_1, \ldots, \vec{x}_n.t_n)$ (and suppose, without loss of generality, that x does not appear in any $\vec{x}_i$). Then $\sigma(t) \Downarrow \theta(\vec{x}_1.t'_1, \ldots, \vec{x}_n.t'_n)$ where, for $1 \leq i \leq n$ and any k, $t_i[f^k/x] \leq t'_i$. By induction, $t_i[\mathit{fix}(f)/x] \leq_l t'_i$, for any i. Therefore $\sigma(e[\mathit{fix}(f)/x]) \leq_{l+1} \sigma(t)$ and hence $e[\mathit{fix}(f)/x] \leq_{l+1} t$. □

Lemma 17

- $\mathrm{Adm}(A + B \mid y : S)$ if $\forall s \in S.\, (A+B)[s/y]$ *type* and $\mathrm{Adm}(A \mid y : S)$ and $\mathrm{Adm}(B \mid y : S)$.
- $\mathrm{Adm}(\Sigma x{:}A.B \mid y : S)$ if $\forall s \in S.\, (\Sigma x{:}A.B)[s/y]$ *type* and $\Sigma y{:}S.A$ *type* and $\mathrm{Adm}(A \mid y : S)$ and $\mathrm{Adm}(B[\pi_1(z), \pi_2(z)/y, x] \mid z : (\Sigma y{:}S.A))$
- $\mathrm{Adm}(\mathbb{N} \mid y : S)$
- $\mathrm{Adm}(a_1 = a_2 \; in \; A \mid y : S)$ if $\forall s \in S.\, (a_1 = a_2 \; in \; A)[s/y]$ *type* and $\mathrm{Adm}(A \mid y : S)$
- $\mathrm{Adm}(\overline{A} \mid y : S)$ if $\forall s \in S.\, \overline{A}[s/y]$ *type* and $\mathrm{Adm}(A \mid y : S)$
- $\mathrm{Adm}(a \; in! \; A \mid y : S)$ if $\forall s \in S.\, (a \; in! \; A)[s/y]$ *type*

Proof

I show the product and equality cases; the other cases are similar but easier.

Case 1: For the product case, let f, t, t' and e be arbitrary. Suppose $e^{[\omega]} \in S$ and j is such that $\forall k \geq j.\, e^{[k]} \in S \wedge t^{[k]} = t'^{[k]} \in (\Sigma x{:}A.B)[e^{[k]}/y]$. It is necessary to show that $t^{[\omega]} = t'^{[\omega]} \in (\Sigma x{:}A.B)[e^{[\omega]}/y]$. Since $e^{[\omega]} \in S$, it follows that $(\Sigma x{:}A.B)[e^{[\omega]}/y]$ *type*. Both $t^{[j]}$ and $t'^{[j]}$ converge to pairs, so, by Corollary 5, $t^{[\omega]} \Downarrow \langle a, b\rangle$ and $t'^{[\omega]} \Downarrow \langle a', b'\rangle$ for some terms a, b, a' and b'. To get that $b = b' \in B[e^{[\omega]}/y][a/x]$, it suffices to show that $\pi_2(t^{[\omega]}) = \pi_2(t'^{[\omega]}) \in B[e^{[\omega]}/y][\pi_1(t^{[\omega]})/x]$. Rearranging, it suffices to show equality in $B[\pi_1(z), \pi_2(z)/y, x][\langle e, \pi_1(t)\rangle^{[\omega]}/z]$.

Since $\mathrm{Adm}(B[\pi_1(z), \pi_2(z)/y, x] \mid z : (\Sigma y{:}S.A))$, it suffices to show that $\langle e, \pi_1(t)\rangle^{[\omega]} \in \Sigma y{:}S.A$ and $\forall k \geq j.\, \langle e, \pi_1(t)\rangle^{[k]} \in \Sigma y{:}S.A \wedge \pi_2(t^{[k]}) = \pi_2(t'^{[k]}) \in B[\pi_1(z), \pi_2(z)/y, x][\langle e, \pi_1(t)\rangle^{[k]}/z]$. The former will follow from $a \in A[e^{[\omega]}/y]$ and the supposition. The left half of the latter also follows from the supposition. Rearranging the right half, it suffices to show that $\pi_2(t^{[k]}) = \pi_2(t'^{[k]}) \in B[e^{[k]}/y][\pi_1(t^{[k]})/x]$, which follows from the supposition. The proof that $a = a' \in A[e^{[\omega]}/y]$ is similar but easier. Hence $t^{[\omega]} = t'^{[\omega]} \in (\Sigma x{:}A.B)[e^{[\omega]}/y]$.

Case 2: For the equality case, again let f, t, t' and e be arbitrary. Suppose $e^{[\omega]} \in S$ and j is such that $\forall k \geq j.\, e^{[k]} \in S \wedge t^{[k]} = t'^{[k]} \in (a_1 = a_2 \; in \; A)[e^{[k]}/y]$. It is necessary to show that $t^{[\omega]} = t'^{[\omega]} \in (a_1 = a_2 \; in \; A)[e^{[\omega]}/y]$. Since $e^{[\omega]} \in S$, it follows that $(a_1 = a_2 \; in \; A)[e^{[\omega]}/y]$ *type*. Both $t^{[j]}$ and $t'^{[j]}$ converge to $\star$, so, by Corollary 5, $t^{[\omega]}$ and $t'^{[\omega]}$ converge to $\star$. It remains to show that $a_1[e^{[\omega]}/y] = a_2[e^{[\omega]}/y] \in A[e^{[\omega]}/y]$. Since $\mathrm{Adm}(A \mid y : S)$, it suffices to show that $\forall k \geq j.\, e^{[k]} \in S \wedge a_1[e^{[k]}/y] = a_2[e^{[k]}/y] \in A[e^{[k]}/y]$. This follows since $(a_1 = a_2 \; in \; A)[e^{[k]}/y]$ is inhabited for all $k \geq j$. □

Lemma 19 $\mathrm{Adm}(\Pi x{:}A.B \mid y : S)$ if $\forall s \in S.\, (\Sigma x{:}A.B)[s/y]$ *type* and $\mathrm{WCoAdm}(A \mid y : S)$ and $\forall s \in S, a \in A[s/y].\, \mathrm{Adm}(B[a/x] \mid y : S)$

Proof

Let f, t, t' and e be arbitrary. Suppose $e^{[\omega]} \in S$ and j is such that $\forall k \geq j.\, e^{[k]} \in S \wedge t^{[k]} = t'^{[k]} \in (\Pi x{:}A.B)[e^{[k]}/y]$. I need to show that $t^{[\omega]} = t'^{[\omega]} \in (\Pi x{:}A.B)[e^{[\omega]}/y]$. Since $e^{[\omega]} \in S$, it follows that $(\Pi x{:}A.B)[e^{[\omega]}/y]$ *type*. Both $t^{[j]}$ and $t'^{[j]}$ converge to lambda abstractions, so, by Corollary 5, $t^{[\omega]} \Downarrow \lambda x.b$ and $t'^{[\omega]} \Downarrow \lambda x.b'$ for some terms b and b'. Suppose $a = a' \in A[e^{[\omega]}/y]$. To get that $b[a/x] = b'[a'/x] \in B[e^{[\omega]}, a/y, x]$ it suffices to show that $t^{[\omega]}a = t'^{[\omega]}a' \in B[e^{[\omega]}, a/y, x]$.

Since $e^{[\omega]} \in S$, it follows that $\mathrm{Adm}(B[a/x] \mid y : S)$. Therefore, it suffices to show that for some j' and all $k \geq j'$, $t^{[k]}a = t'^{[k]}a' \in B[e^{[k]}, a/y, x]$. Since $\mathrm{WCoAdm}(A \mid y : S)$, there exists j'' such that $\forall k \geq j''.\, a = a' \in A[e^{[k]}/y]$. Therefore $j' = \max(j, j'')$ suffices. □

Lemma 20

- $A + B$ is (weakly) coadmissible for y in S if $\forall s \in S.\, (A+B)[s/y]$ *type* and A and B are (weakly) coadmissible for y in S
- $\mathrm{WCoAdm}(\Sigma x{:}A.B \mid y : S)$ if $\forall s \in S.\, (\Sigma x{:}A.B)[s/y]$ *type* and $\mathrm{WCoAdm}(A \mid y : S)$ and $\forall s \in S, a \in A[s/y].\, \mathrm{WCoAdm}(B[a/x] \mid y : S)$
- $\mathrm{CoAdm}(\Sigma x{:}A.B \mid y : S)$ if $\forall s \in S.\, (\Sigma x{:}A.B)[s/y]$ *type* and $\Sigma y{:}S.A$ *type* and $\mathrm{CoAdm}(A \mid y : S)$ and $\mathrm{CoAdm}(B[\pi_1(z), \pi_2(z)/y, x] \mid z : (\Sigma y{:}S.A))$
- $\mathbb{N}$ is strongly or weakly coadmissible for y in any S
- $\mathrm{CoAdm}(a_1 = a_2 \text{ } in \text{ } A \mid y : S)$ if $\forall s \in S.\, (a_1 = a_2 \text{ } in \text{ } A)[s/y]$ *type* and $\mathrm{CoAdm}(A \mid y : S)$
- $\overline{A}$ is (weakly) coadmissible for y in S if $\forall s \in S.\, \overline{A}[s/y]$ *type* and A is (weakly) coadmissible for y is S
- a *in*! A is strongly or weakly coadmissible for y in S if $\forall s \in S.\, (a \text{ } in!\text{ } A)[s/y]$ *type*

Proof

The proof is largely similar to the preceding proofs, but inverted. I show the proofs for full coadmissibility of products and partial types.

Case 1: For the product case, let f, t, t' and e be arbitrary. Suppose $e^{[\omega]} \in S$, j is such that $\forall k \geq j.\, e^{[k]} \in S$, and $t^{[\omega]} = t'^{[\omega]} \in (\Sigma x{:}A.B)[e^{[\omega]}/y]$. It is necessary to show that there exists j' such that $\forall k \geq j'.\, t^{[k]} = t'^{[k]} \in (\Sigma x{:}A.B)[e^{[k]}/y]$. For any $k \geq j$, $(\Sigma x{:}A.B)[e^{[k]}/y]$ *type*. Both $t^{[\omega]}$ and $t'^{[\omega]}$ converge to pairs, so, by compactness, there exists some j'' such that for all $k \geq j''$, $t^{[k]}$ and $t^{[k]}$ converge to pairs. Thus it suffices to show that for some $j' \geq \max(j, j'')$ and all $k \geq j'$, $\pi_1(t^{[k]}) = \pi_1(t'^{[k]}) \in A[e^{[k]}/y]$ and $\pi_2(t^{[k]}) = \pi_2(t'^{[k]}) \in B[e^{[k]}, \pi_1(t^{[k]})/y, x]$. I show the latter; the former is similar.

Rearranging, it suffices to show equality in $B[\pi_1(z), \pi_2(z)/y, x][\langle e, \pi_1(t)\rangle^{[k]}/z]$. By coadmissibility, it suffices to show $\langle e, \pi_1(t)\rangle^{[\omega]} \in \Sigma y{:}S.A$ and $\exists j'''.\, \forall k \geq j'''.\, \langle e, \pi_1(t)\rangle^{[k]} \in \Sigma y{:}S.A$ and $\pi_2(t^{[\omega]}) = \pi_2(t'^{[\omega]}) \in B[\pi_1(z), \pi_2(z)/y, x][\langle e, \pi_1(t)\rangle^{[\omega]}/z]$. The first follows from the supposition and the second will follow from $\pi_1(t^{[k]}) \in A[e^{[k]}/y]$ and the supposition. Rearranging the third, is suffices to show that $\pi_2(t^{[\omega]}) = \pi_2(t'^{[\omega]}) \in B[e^{[\omega]}/y][\pi_1(t^{[\omega]})/x]$, which follows from the supposition.

Case 2: For the partial type case, let f, t, t' and e be arbitrary. Suppose $e^{[\omega]} \in S$, j is such that $\forall k \geq j.\, e^{[k]} \in S$, and $t^{[\omega]} = t'^{[\omega]} \in \overline{A}[e^{[\omega]}/y]$. I need to show that there exists j' such that $\forall k \geq j'.\, t^{[k]} = t'^{[k]} \in \overline{A}[e^{[k]}/y]$. For any $k \geq j$, $\overline{A}[e^{[k]}/y]$ *type*. Suppose $t^{[\omega]}$ does not converge. (Note the non-constructivity of this argument.) Then $t'^{[\omega]}$ does not converge and neither does $t^{[k]}$ or $t'^{[k]}$ for any k (since $t^{[k]} \leq t^{[\omega]}$ and $t'^{[k]} \leq t'^{[k]}$ for all k). Thus for all $k \geq j$, $t^{[k]} = t'^{[k]} \in \overline{A}[e^{[k]}/y]$.

Suppose $t^{[\omega]}$ converges. Then $t^{[\omega]} = t'^{[\omega]} \in A[e^{[\omega]}/y]$. By coadmissibility, there exists j' such that $\forall k \geq j'.\, t^{[k]} = t'^{[k]} \in A[e^{[k]}/y]$. Hence $\forall k \geq \max(j, j').\, t^{[k]} = t'^{[k]} \in \overline{A}[e^{[k]}/y]$. □

Lemma 22 (Predicate-admissibility and weak and full coadmissibility of case analysis.)

Proof

The proof follows the same lines as those of Lemmas 17 and 20.

Lemma 26 The type $\{x : A \mid B\}$ is admissible if:

- $\mathrm{Adm}(A)$, and
- $\mathrm{Adm}(B \mid x : A)$, and
- there exists b such that $b[a/x] \in B[a/x]$ whenever $a \in A$ and $\exists b'.\, b' \in B[a/x]$.

Proof

Let f, t and t' be arbitrary. Suppose j is such that $\forall k \geq j.\, t^{[k]} = t'^{[k]} \in \{x : A \mid B\}$. Since $\{x : A \mid B\}$ is inhabited it is a type. Since $\mathrm{Adm}(A)$, $t^{[\omega]} = t'^{[\omega]} \in A$. By set membership, for all $k \geq j$ there exists b' such that $b' \in B[t^{[k]}/x]$. Thus $b[t^{[k]}/x] \in B[t^{[k]}/x]$ for all $k \geq j$. Hence $b[t^{[\omega]}/x] \in B[t^{[\omega]}/x]$ follows by predicate-admissibility. □

Lemma 27 (Predicate-admissibility of set types.)

Proof

The proof follows the same lines as Lemma 26.

Lemma 28 (Weak and full coadmissibility and monotonicity of set types.)

Proof

The proof follows the same lines as Lemma 20 and Proposition 25.

Lemma 29 (Admissibility, predicate-admissibility, weak and full coadmissibility and monotonicity of quotient types.)

Proof

The proof follows the same lines as the proofs for the set type.

Theorem 30 There does not exist an algorithm that computes an integer j such that $\forall k \geq j. t = t' \in \overline{T}[e^{[k]}/x]$, when given S, T, f, t, t', e and i such that:

- $\forall s \in S. \overline{T}[s/x]$ *type*
- $\mathrm{CoAdm}(T \mid x : S)$
- $e^{[\omega]} \in S$
- $\forall k \geq i. e^{[k]} \in S$
- $t = t' \in \overline{T}[e^{[\omega]}/x]$

Proof

Suppose such an algorithm exists. Let g be an arbitrary term that computes a total function on integers; that is, $g \in \mathbb{N} \to \mathbb{N}$. Given the algorithm, we may effectively determine whether g iterated on 1 ever computes 0, which is certainly undecidable. Let $f = \lambda h. \lambda n.\ \mathit{if}\ n =_N 0\ \mathit{then}\ 0\ \mathit{else}\ h(gx)$ and let $h = \mathit{fix}(f)$. Note that $f \in \overline{\mathbb{N} \to \overline{\mathbb{N}}} \to \mathbb{N} \to \overline{\mathbb{N}}$ and $h \in \mathbb{N} \to \overline{\mathbb{N}}$. By construction, g iterated on 1 computes 0 if and only if $h(1)\downarrow$.

We will use the algorithm to determine an upper bound on the number of recursive calls needed to simulate h. Let

$$\begin{array}{rcl} S &=& \overline{\mathbb{N}} \\ T &=& x\ \mathit{in}!\ \mathbb{N} \\ f &=& \text{as above} \\ t, t' &=& \mathit{let}\ \ y = h(1)\ \ \mathit{in}\ \star \\ e &=& w(1) \\ i &=& 0 \end{array}$$

Observe that $e^{[\omega]} = h(1)$ and $e^{[k]} = (f^k)(1)$, so the first four preconditions of the algorithm are satisfied. Moreover, $\mathrm{CoAdm}(T \mid x : S)$ can be shown constructively. For the final precondition, suppose $t\downarrow$. Then $h(1)\downarrow$ so $t \in h(1)\ \mathit{in}!\ \mathbb{N}$.

Therefore let j be the result computed by the algorithm. I show that $f^j(1)\downarrow$ exactly when $h(1)\downarrow$. Since $f^j(1)$ approximates $h(1)$, it follows that $f^j(1)\downarrow$ implies $h(1)\downarrow$. By the algorithm specification, $t \in \overline{f^j(1)\ \mathit{in}!\ \mathbb{N}}$. If $h(1)\downarrow$ then $t\downarrow$, so $t \in f^j(1)\ \mathit{in}!\ \mathbb{N}$ and consequently $f^j(1)\downarrow$.

Let $h' = \overbrace{f(f \cdots f}^{j \text{ times}}(\lambda y.1) \cdots)$, and observe that $f^j(1)\downarrow$ exactly when $h'(1) \Downarrow 0$. Consequently $h(1)\downarrow$ exactly when $h'(1) \Downarrow 0$. However, h' is total, so we may decide whether $h(1)\downarrow$ by running $h'(1)$. $\square$