Impredicative Strong Existential Equivalent to Type:Type

James G. Hook
Douglas J. Howe

TR 86-760
June 1986

Department of Computer Science
Cornell University
Ithaca, NY 14853

Impredicative Strong Existential Equivalent to Type:Type

James G. Hook
Douglas J. Howe *
Cornell University

June 12, 1986

The second-order existential quantifier provides a facility to express abstract data types in the typed λ-calculus[17]. When languages that allow "dependent" types are studied it seems natural to extend the interpretation of existential quantification to the full generality of Martin-Löf's Σ-types by allowing the carrier of an abstract type to be a legal type[13,12,10]. The languages KR and PEBBLE allow such a construction[7,1]. This paper shows that when the impredicative second-order λ-calculus is extended in this way the type system becomes (logically) inconsistent; that is, terms can be constructed in every type, including those types that correspond to absurd propositions.

The inconsistency is shown by translating a well-known inconsistent theory based on the axiom **type** : **type** into the theory with the strong existential. The most direct implication of the inconsistency is the failure of the strong normalization property—which insures that all computations on type correct terms terminate. Recent work by Cardelli suggests that these theories do have a weaker form of consistency[2]. He develops a domain theoretic account of a language with **type** : **type** and the strong existential and shows that the semantic soundness theorem (originally stated by Milner for ML[15]) holds.

*The work of this author was supported, in part, by NSF grant no. MCS-81-04018.

1 The Strong Existential

To understand the meaning of quantification in a formal system one must first know the collection over which the quantified variables range. In first-order logic the domain is defined before the quantifiers are interpreted—making their interpretation straightforward. However, in second-order logic the domain of propositional variables must include the interpretations of all propositions. For example, the statement "for all propositions P, P implies P" is a proposition; so it is itself within the range of the variable P and thus it stipulates, among other things, that "(for all propositions P, P implies P) implies (for all propositions P, P implies P)". Such a definition, where the definition of the object refers to the entire collection to which it belongs, is said to be impredicative.

The second major issue in comprehending a quantified formula is how it may be used in a logical argument. This has two parts: how can the truth of a quantified formula be demonstrated and what may be concluded from the assertion of one.

To constructively prove an existentially quantified statement, one must provide an element of the domain of quantification, called the witness, and a proof that the statement is true of the witness. For example, a constructive proof of $\exists n \,.\, n = 3$ must yield the natural number 3 together with a proof that $3 = 3$ holds. In systems based on the propositions-as-types principle of Curry and Howard[5,8], where a proposition is identified with the type of its proofs, the proof object associated with $\exists t \,.\, \sigma(t)$ is a pair, $\langle \tau, M \rangle$, where M is a proof of $\sigma(\tau)$.

Once this information has been included in the proof, it seems natural to permit explicit naming of the witness via the first projection of the existential proof object. This is called the "strong" interpretation by Howard in his paper "Formulas-as-types"[8].

In programming languages, the second-order strong existential (where the quantification ranges over all types) corresponds to a very natural expression of abstract data types. It allows values in the "implementation type" of an abstract type to be manipulated in a type-safe way without necessarily opening up the data type. For example, if $\gamma(g)$ is the signature of the abstract type "group" then the conjugation function would be

written (modulo the syntax of the function body):

$$\mathtt{conj} = \lambda G : (\exists g \,.\, \gamma(g)) \,.\, \lambda a : |G| \,.\, \lambda b : |G| \,.\, a \circ b \circ a^{-1}$$

where the notation $|G|$ refers to the witness of the existential, which, in this context, is the implementation type (carrier) of G. This operation has the type:

$$\Pi G \in (\exists g \,.\, \gamma(g)) \,.\, |G| \to |G| \to |G|$$

If H is an implementation of the abstract type of groups, i.e. $H \in \exists g \,.\, \gamma(g)$, then $\mathtt{conj}(H)$ has the type:

$$|H| \to |H| \to |H|$$

which, we claim, is its most natural type. (Compare this to the treatment in Mitchell and Plotkin's 1985 paper where no elements may be manipulated in a context where the abstract type is not open[17].) This use of the carrier projection, combined with the judicious use of λ-abstraction to hide and expose abstract types, suffices to model the highly expressive languages Russell, CLU and Pebble[7,1,9]. The details of the application of these techniques will appear in Hook's forthcoming thesis and a paper currently in preparation by Demers and Hook.

The first type theory capable of expressing the strong existential was presented in Martin-Löf's 1971 paper "A theory of types"[13]. This theory unified the notation for first and second-order quantification by eliminating the distinction between terms and types, and by adding a constant **type** (in Martin-Löf's notation V), to represent the collection of all types, with the axiom:

$$\textbf{type} : \textbf{type}$$

In the context of this axiom, special cases of the general product type, Π, can be used to express both the traditional first-order implication and second-order universal quantification, as well as functional dependence. Similarly, Σ, the general sum constructor, models both conjunction and the strong interpretation of the existential quantifier.

Unfortunately the axiom **type** : **type** admitted more than notational economy—it made the theory inconsistent, as was demonstrated by Girard in his 1972 thesis where he constructs a paradox (based on the Burali-Forti

paradox) in Martin-Löf's theory[6]. The paradox is presented in English in Coquand's paper "An Analysis of Girard's Paradox"[4] and in the unpublished paper by Martin-Löf "An Intuitionistic Theory of Types"[11] (the explanation of the paradox was omitted from the published version of this paper, entitled "An Intuitionistic Theory of Types: Predicative Part" [12]). After the discovery of this paradox Martin-Löf abandoned impredicative type theories.

This paper proves that even without collapsing types and terms and introducing the axiom **type** : **type**, an impredicative system with the second-order strong existential is logically inconsistent.

2 The Result

The presentation of Martin-Löf's 1971 theory used here is the theory $\Lambda^{\tau:\tau}$ from Meyer and Reinhold's 1986 POPL paper (this presentation was selected because it has been published)[14]. The core of this theory is given in section 2.1. The theory with strong existential that is being shown inconsistent is $\Lambda^{\exists\,|\cdot|}$; its core appears in section 2.2. Both theories have large theories of equality that must be considered for the technical completeness of the result. They are presented in the appendix; on first reading they may be ignored. They consist of the basic axioms of equivalence, the axioms of computation, and rules that lift these axioms through term constructors.

The result of this note is the construction of a simple translation of proofs in $\Lambda^{\tau:\tau}$ to $\Lambda^{\exists\,|\cdot|}$. This, together with the fact that the translation of a certain absurd statement in $\Lambda^{\tau:\tau}$ is an absurd statement in $\Lambda^{\exists\,|\cdot|}$, establishes that $\Lambda^{\exists\,|\cdot|}$ is inconsistent. The translation maps quantification over all types to quantification over the existential type $\exists t.\mathrm{N}_1$, where N_1 is any non-empty type and 0 is an element of N_1. This existential type contains the pair $\langle \tau, 0\rangle$ for all types τ—including $\exists t.\mathrm{N}_1$. So, the following statement is provable in $\Lambda^{\exists\,|\cdot|}$.

$$\vdash \langle \exists t\,.\,\mathrm{N}_1, 0\rangle : \exists t\,.\,\mathrm{N}_1$$

which is equivalent, via computation, to

$$\vdash \langle \exists t\,.\,\mathrm{N}_1, 0\rangle : |\langle \exists t\,.\,\mathrm{N}_1, 0\rangle| \qquad (1)$$

The translation function, $\overline{\cdot}$, is defined so that if $\mathcal{A} \vdash_{\tau:\tau} M : N$ then $\overline{\mathcal{A}} \vdash \overline{M} : |\overline{N}|$. Thus, by taking $\overline{\textbf{type}} = \langle \exists t \,.\, \mathrm{N}_1, 0 \rangle$, the translation of the $\Lambda^{\tau:\tau}$ axiom $\vdash_{\tau:\tau} \textbf{type} : \textbf{type}$ becomes (1). The complete definition of the translation is given in section 2.3.

2.1 $\Lambda^{\tau:\tau}$

Syntax

$$\begin{array}{rcl} M & := & x \\ & | & \textbf{type} \\ & | & \lambda x \in M \,.\, M \\ & | & MM \\ & | & \Pi x \in M \,.\, M \end{array}$$

The core theory

The sequent notation used for inference rules in this paper consists of a list of assumptions and an equality or typing assertion separated by a $\vdash$. The list of assumptions will often be treated as a finite function from identifiers to terms, as in rules $\Lambda^{\tau:\tau}2$ and $\Lambda^{\tau:\tau}3$.

Type in Type
$$\mathcal{A} \vdash_{\tau:\tau} \textbf{type} : \textbf{type} \qquad (\Lambda^{\tau:\tau}1)$$

Assumption
$$\mathcal{A} \vdash_{\tau:\tau} x : \mathcal{A}(x) \qquad (\Lambda^{\tau:\tau}2)$$

Π-Intro
$$\frac{\mathcal{A} \vdash_{\tau:\tau} \sigma : \textbf{type} \qquad \mathcal{A}[x:\sigma] \vdash_{\tau:\tau} M(x) : \tau(x) \quad x \notin dom(\mathcal{A})}{\mathcal{A} \vdash_{\tau:\tau} \lambda x \in \sigma \,.\, M(x) : \Pi x \in \sigma \,.\, \tau(x)} \qquad (\Lambda^{\tau:\tau}3)$$

Π-Elim
$$\frac{\mathcal{A} \vdash_{\tau:\tau} M : \Pi x \in \sigma \,.\, \tau(x) \qquad \mathcal{A} \vdash_{\tau:\tau} N : \sigma}{\mathcal{A} \vdash_{\tau:\tau} MN : \tau(N)} \qquad (\Lambda^{\tau:\tau}4)$$

Π-Form
$$\frac{\mathcal{A} \vdash_{\tau:\tau} \sigma : \textbf{type} \qquad \mathcal{A}[x:\sigma] \vdash_{\tau:\tau} \tau(x) : \textbf{type}}{\mathcal{A} \vdash_{\tau:\tau} \Pi x \in \sigma \,.\, \tau(x) : \textbf{type}} \qquad (\Lambda^{\tau:\tau}5)$$

Computation

$$\frac{\mathcal{A} \vdash_{\tau:\tau} M : \sigma \quad \mathcal{A} \vdash_{\tau:\tau} \tau = \sigma}{\mathcal{A} \vdash_{\tau:\tau} M : \tau} \qquad (\Lambda^{\tau:\tau}6)$$

2.2 $\Lambda^{\exists|\cdot|}$

Syntax

$$\begin{array}{rcl@{\qquad}rcl} M & ::= & x & \tau & ::= & t \\ & | & \lambda x \in \tau \,.\, M(x) & & | & \Pi x \in \tau \,.\, \tau(x) \\ & | & MM & & | & \forall x \,.\, \tau(x) \\ & | & \Lambda t \,.\, M(t) & & | & \Sigma x \in \tau \,.\, \tau(x) \\ & | & M\tau & & | & \exists x \,.\, \tau(x) \\ & | & \langle M, M \rangle_\tau & & | & |M| \\ & | & \langle \tau, M \rangle_\tau & & & \\ & | & \mathcal{L}MM & & & \end{array}$$

The construction $\mathcal{L}MN$ is used to analyze elements of Σ- and $\exists$-types. It is similar to PRL's **spread** and Cardelli's `rip` operators[3,2]. However, it is not a binding operator. Its use is defined by the Σ- and $\exists$-Elim rules ($\Lambda^{\exists|\cdot|}9$) and ($\Lambda^{\exists|\cdot|}12$) and by its reduction rules ($\Lambda^{\exists|\cdot|}33$) and ($\Lambda^{\exists|\cdot|}34$).

The core theory

In the presentation below the following abbreviations are used:

$$\begin{aligned} \pi_1 &= \lambda x \in \sigma \,.\, \lambda y \in \tau(x) \,.\, x \\ \pi_2 &= \lambda x \in \sigma \,.\, \lambda y \in \tau(x) \,.\, y \\ \pi_2^2 &= \Lambda t \,.\, \lambda x \in \sigma(t) \,.\, x \end{aligned}$$

The actual values of σ and τ are apparent from the context.

Assumption

$$\mathcal{A} \vdash x : \mathcal{A}(x) \qquad (\Lambda^{\exists|\cdot|}1)$$

Π-Intro

$$\frac{\mathcal{A} \vdash \sigma : \textbf{type} \quad \mathcal{A}[x:\sigma] \vdash M(x) : \tau(x) \quad x \notin dom(\mathcal{A})}{\mathcal{A} \vdash \lambda x \in \sigma \,.\, M(x) : \Pi x \in \sigma \,.\, \tau(x)} \qquad (\Lambda^{\exists|\cdot|}2)$$

Π-Elim
$$\frac{\mathcal{A} \vdash M : \Pi x \in \sigma \,.\, \tau(x) \qquad \mathcal{A} \vdash N : \sigma}{\mathcal{A} \vdash MN : \tau(N)} \qquad (\Lambda^{\exists\,|\cdot|}3)$$

Π-Form
$$\frac{\mathcal{A} \vdash \sigma : \textbf{type} \qquad \mathcal{A}[x : \sigma] \vdash \tau(x) : \textbf{type} \quad x \notin dom(\mathcal{A})}{\mathcal{A} \vdash \Pi x \in \sigma \,.\, \tau(x) : \textbf{type}} \qquad (\Lambda^{\exists\,|\cdot|}4)$$

$\forall$-Intro
$$\frac{\mathcal{A}[s : \textbf{type}] \vdash M(s) : \tau(s) \quad s \notin dom(\mathcal{A})}{\mathcal{A} \vdash \Lambda s \,.\, M(s) : \forall s \,.\, \tau(s)} \qquad (\Lambda^{\exists\,|\cdot|}5)$$

$\forall$-Elim
$$\frac{\mathcal{A} \vdash M : \forall s \,.\, \tau(s) \qquad \mathcal{A} \vdash \sigma : \textbf{type}}{\mathcal{A} \vdash M\sigma : \tau(\sigma)} \qquad (\Lambda^{\exists\,|\cdot|}6)$$

$\forall$-Form
$$\frac{\mathcal{A}[s : \textbf{type}] \vdash \tau(s) : \textbf{type} \quad s \notin dom(\mathcal{A})}{\mathcal{A} \vdash \forall s \,.\, \tau(s) : \textbf{type}} \qquad (\Lambda^{\exists\,|\cdot|}7)$$

Σ-Intro
$$\frac{\mathcal{A} \vdash M : \sigma \qquad \mathcal{A} \vdash N : \tau(M) \qquad \mathcal{A}[x : \sigma] \vdash \tau(x) : \textbf{type} \quad x \notin dom(\mathcal{A})}{\mathcal{A} \vdash \langle M, N \rangle_{\Sigma x \in \sigma . \tau(x)} : \Sigma x \in \sigma \,.\, \tau(x)} \qquad (\Lambda^{\exists\,|\cdot|}8)$$

Σ-Elim
$$\frac{\mathcal{A} \vdash N : \Sigma x \in \sigma \,.\, \tau(x) \qquad \mathcal{A} \vdash M : \Pi x \in \sigma \,.\, \Pi y \in \tau(x) \,.\, \rho(x, y)}{\mathcal{A} \vdash \mathcal{L}MN : \rho(\mathcal{L}\pi_1 N, \mathcal{L}\pi_2 N)} \qquad (\Lambda^{\exists\,|\cdot|}9)$$

Σ-Form
$$\frac{\mathcal{A} \vdash \sigma : \textbf{type} \qquad \mathcal{A}[x : \sigma] \vdash \tau(x) : \textbf{type} \quad x \notin dom(\mathcal{A})}{\mathcal{A} \vdash \Sigma x \in \sigma \,.\, \tau(x) : \textbf{type}} \qquad (\Lambda^{\exists\,|\cdot|}10)$$

$\exists$-Intro
$$\frac{\mathcal{A} \vdash \tau : \textbf{type} \qquad \mathcal{A}[s : \textbf{type}] \vdash \sigma(s) : \textbf{type} \quad s \notin dom(\mathcal{A}) \qquad \mathcal{A} \vdash M : \sigma(\tau)}{\mathcal{A} \vdash \langle \tau, M \rangle_{\exists s . \sigma(s)} : \exists s \,.\, \sigma(s)} \qquad (\Lambda^{\exists\,|\cdot|}11)$$

$\exists$-Elim
$$\frac{\mathcal{A} \vdash N : \exists s \,.\, \sigma(s) \quad \mathcal{A} \vdash M : \forall t \,.\, \Pi x \in \sigma(t) \,.\, \tau(t, x)}{\mathcal{A} \vdash \mathcal{L}MN : \tau(|N|, \mathcal{L}\pi_2^2 N)} \qquad (\Lambda^{\exists|\cdot|}12)$$

$\exists$-Form
$$\frac{\mathcal{A}[s : \mathbf{type}] \vdash \sigma(s) : \mathbf{type} \quad s \notin dom(\mathcal{A})}{\mathcal{A} \vdash \exists s \,.\, \sigma(s) : \mathbf{type}} \qquad (\Lambda^{\exists|\cdot|}13)$$

Carrier Form
$$\frac{\mathcal{A} \vdash M : \exists s \,.\, \sigma(s)}{\mathcal{A} \vdash |M| : \mathbf{type}} \qquad (\Lambda^{\exists|\cdot|}14)$$

Computation
$$\frac{\mathcal{A} \vdash M : \sigma \quad \mathcal{A} \vdash \sigma = \tau}{\mathcal{A} \vdash M : \tau} \qquad (\Lambda^{\exists|\cdot|}15)$$

2.3 The translation

The translation of terms

$$\begin{aligned}
\overline{x} &= x \\
\overline{\mathbf{type}} &= \langle \exists t \,.\, \mathrm{N}_1, 0 \rangle_{\exists t.\mathrm{N}_1} \\
\overline{\lambda x \in \sigma \,.\, a} &= \lambda x \in |\overline{\sigma}| \,.\, \overline{a} \\
\overline{ab} &= \overline{a}\overline{b} \\
\overline{\Pi x \in \sigma \,.\, \tau(x)} &= \langle \Pi x \in |\overline{\sigma}| \,.\, |\overline{\tau(x)}|, 0 \rangle_{\exists t.\mathrm{N}_1}
\end{aligned}$$

The translation of environments

$$\begin{aligned}
\overline{\phi} &= \phi \\
\overline{\mathcal{A}[x : \tau]} &= \overline{\mathcal{A}}[x : |\overline{\tau}|]
\end{aligned}$$

2.4 The main lemmas

The intuitive argument of the beginning of this section is formalized below. Lemma 1 shows that the translation is well behaved with respect to substitution. The technical core of the presentation is Lemma 2, which shows that the translation preserves provability. It is proved by a simple calculation which verifies the provability of the translated $\Lambda^{\tau;\tau}$ inference

rules; the proof contains no surprises. Once this machinery is in place the argument is quickly concluded by establishing Theorem 4—the assertion of the inconsistency of $\Lambda^{\exists|\cdot|}$.

Lemma 1 (Substitution) $\overline{M[N/x]} = \overline{M}[\overline{N}/x]$.

Proof The proof is a straightforward induction on M. ∎

Lemma 2 *1. If* $\mathcal{A} \vdash_{\tau:\tau} M : N$ *then* $\overline{\mathcal{A}} \vdash \overline{M} : |\overline{N}|$;

2. If $\mathcal{A} \vdash_{\tau:\tau} M = N$ *then* $\overline{\mathcal{A}} \vdash \overline{M} = \overline{N}$.

Proof The proof is by induction on the length of the derivation in $\Lambda^{\tau:\tau}$. We proceed by case analysis on the last rule used in the derivation.

[**type** : **type**]
Show $\overline{\mathcal{A}} \vdash \overline{\mathbf{type}} : |\overline{\mathbf{type}}|$. By definition, this is

$$\overline{\mathcal{A}} \vdash \langle \exists t \,.\, \mathrm{N}_1, 0\rangle_{\exists t.\mathrm{N}_1} : |\langle \exists t \,.\, \mathrm{N}_1, 0\rangle_{\exists t.\mathrm{N}_1}|$$

which, by carrier reduction is equivalent to

$$\overline{\mathcal{A}} \vdash \langle \exists t \,.\, \mathrm{N}_1, 0\rangle_{\exists t.\mathrm{N}_1} : \exists t \,.\, \mathrm{N}_1$$

which is an instance of $\exists$-introduction.

[Assumption]
Show $\overline{\mathcal{A}} \vdash \overline{x} : |\overline{\mathcal{A}(x)}|$. This follows from the definition of $\overline{\mathcal{A}}$ and the rule of assumption in DΛ.

[Π-Intro]
By induction we get

$$\overline{\mathcal{A}} \vdash \overline{\sigma} : |\overline{\mathbf{type}}|$$

and

$$\overline{\mathcal{A}}[x : |\overline{\sigma}|] \vdash \overline{M(x)} : |\overline{\tau(x)}|$$

The first of these computes to

$$\overline{\mathcal{A}} \vdash \overline{\sigma} : \exists t \,.\, \mathrm{N}_1$$

and so by carrier formation

$$\overline{\mathcal{A}} \vdash |\overline{\sigma}| : \mathbf{type}$$

Hence, by application of DΛ's Π-Intro rule,

$$\overline{\mathcal{A}} \vdash \lambda x \in |\overline{\sigma}| \,.\, \overline{M(x)} : \Pi x \in |\overline{\sigma}| \,.\, |\overline{\tau(x)}|$$

which, by carrier reduction, is equivalent to

$$\overline{\mathcal{A}} \vdash \lambda x \in |\overline{\sigma}| \,.\, \overline{M(x)} : |\langle \Pi x \in |\overline{\sigma}| \,.\, |\overline{\tau(x)}|, 0\rangle_{\exists t.\mathrm{N}_1}|$$

as required.

[Π-Elim]

By induction

$$\overline{\mathcal{A}} \vdash \overline{M} : |\langle \Pi x \in |\overline{\sigma}| \,.\, |\overline{\tau(x)}|, 0\rangle_{\exists t.\mathrm{N}_1}|$$

and

$$\overline{\mathcal{A}} \vdash \overline{N} : |\overline{\sigma}|$$

The first sequent simplifies by the carrier reduction rule to

$$\overline{\mathcal{A}} \vdash \overline{M} : \Pi x \in |\overline{\sigma}| \,.\, |\overline{\tau(x)}|$$

Hence, by Π-Elim

$$\overline{\mathcal{A}} \vdash \overline{MN} : |\overline{\tau}(\overline{N})|$$

which by Lemma 1 establishes

$$\overline{\mathcal{A}} \vdash \overline{MN} : |\overline{\tau(N)}|$$

as required.

[Π-Formation]

By translation and computation the antecedents become:

$$\overline{\mathcal{A}} \vdash \overline{\sigma} : \exists t \,.\, \mathrm{N}_1$$
$$\overline{\mathcal{A}}[x : |\overline{\sigma}|] \vdash \overline{\tau(x)} : \exists t \,.\, \mathrm{N}_1$$

hence, by carrier formation

$$\overline{\mathcal{A}} \vdash |\overline{\sigma}| : \mathbf{type}$$
$$\overline{\mathcal{A}}[x : |\overline{\sigma}|] \vdash |\overline{\tau(x)}| : \mathbf{type}$$

which, by Π-formation and ∃-intro, establishes

$$\overline{\mathcal{A}} \vdash \langle \Pi x \in |\overline{\sigma}| \,.\, |\overline{\tau(x)}|, 0\rangle_{\exists t.\mathrm{N}_1} : \exists t \,.\, \mathrm{N}_1$$

[Type computation]
The result follows from carrier congruence and type computation in $\Lambda^{\exists|\cdot|}$ applied to the second inductive hypothesis.

[Reflexive, Symmetric and Transitive][1]
By induction and application of the corresponding rules.

[Left and Right congruence]
By induction and application of the corresponding rules. (Note that even when $\sigma = \mathbf{type}$ that $\overline{M}$ is a term.)

[β-conversion]
By induction and the β-rule for Π-types.

[Weak extensionality]
By induction and the weak extensionality rule for Π-types.

[Extensionality]
By induction and extensionality.

[Type conversion]
By carrier-congruence, induction and the type conversion rule for Π-types. ∎

The argument for inconsistency is completed using the following fact:

Fact 3 *There is a term of type* $\Pi t \in (\exists t \,.\, \mathrm{N}_1) \,.\, |t| \to \forall s \,.\, s$ *in* $\Lambda^{\exists|\cdot|}$.

Proof The term $\lambda f \in (\Pi t \in (\exists t \,.\, \mathrm{N}_1) \,.\, |t|) \,.\, \Lambda s \,.\, f\langle s, 0\rangle$ inhabits the type. ∎

Theorem 4 *The type system of* $\Lambda^{\exists|\cdot|}$ *is inconsistent.*

Proof By the construction of Girard, there is a term of type $\Pi t \in \mathbf{type} \,.\, t$ in $\Lambda^{r:r}$. By the definition of the translation from $\Lambda^{r:r}$ to $\Lambda^{\exists|\cdot|}$ this type becomes $(\Pi t \in \exists t \,.\, \mathrm{N}_1)|t|$, and by Lemma 2 and Fact 3 this implies there is a term of type $\forall t \,.\, t$ in $\Lambda^{\exists|\cdot|}$. Therefore all types are inhabited. ∎

[1] The equality rules for both theories appear in the appendix.

3 Discussion

The property of being logically consistent has not been traditionally considered a requirement for programming language type systems. In fact no popular languages and only a few experimental languages have this property. It is important, however, that it is a property that distinguishes the impredicative second-order λ-calculus with Π-types, in which a weak form of the existential is definable, from $\Lambda^{\exists|\cdot|}$. One of the implications of this construction is that there can be no account of the abstract type mechanism of Russell (or similar languages) in a logically consistent extension of the second-order λ-calculus. Cardelli has shown that a weaker notion of consistency—the semantic soundness property of Milner—holds for $\Lambda^{r:r}$[2,15]. We conjecture that this property holds for $\Lambda^{\exists|\cdot|}$ as well.

The results in this paper were independently obtained by Coquand[4] and by Mitchell and Harper[16]. Coquand's development abstracts the basic structure of the Girard paradox and shows directly that it may be obtained in the theory $\Lambda^{\exists|\cdot|}$. Mitchell and Harper have not published their result, but we believe they used the same reduction as was employed in this paper.

4 Acknowledgements

Discussions with Tim Griffin and Alan Demers were very important for the discovery of this result. Todd Knoblock gave several useful comments on drafts of this paper.

References

[1] R. Burstall and B. Lampson. A kernel language for abstract data types and modules. In G. Kahn, D.B. MacQueen, and G. Plotkin, editors, *Semantics of Data Types, International Symposium, Sophia-Antipolis, France*, Springer-Verlag, Berlin, June 1984.

[2] L. Cardelli. *A Polymorphic lambda-calculus with Type:Type.* Technical Report, Digital Equipment Corporation, Systems Research Center,

Palo Alto, California, December 1985.

[3] R. Constable et al. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice Hall, Englewood Cliffs, New Jersey, 1986.

[4] Thierry Coquand. An analysis of Girard's paradox. 1986. To appear in the proceedings of the 1986 LICS conference, to be published by the IEEE.

[5] H. B. Curry and R. Feys. *Combinatory Logic.* Volume I, North-Holland, Amsterdam, 1958.

[6] J.-Y. Girard. Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur. 1972. Thèse de Doctorat d'Etat, University of Paris, France.

[7] James G. Hook. Understanding Russell—a first attempt. In G. Kahn, D.B. MacQueen, and G. Plotkin, editors, *Semantics of Data Types, International Symposium, Sophia-Antipolis, France*, Springer-Verlag, Berlin, June 1984.

[8] W. A. Howard. The formulae-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*, Academic Press, New York, 1980. A version of this paper was privately circulated in 1969.

[9] B. Liskov, R. Atkinson, T. Bloom, E. Moss, J. C. Schaffert, R. Scheifler, and A. Snyder. *CLU Reference Manual.* Volume 114 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1981.

[10] Per Martin-Löf. Constructive mathematics and computer programming. In *Sixth International Congress for Logic, Methodology, and Philosophy of Science*, pages 153–175, North Holland, 1982.

[11] Per Martin-Löf. An intuitionistic theory of types. 1972. Mimeographed manuscript. This is an unpublished version of [12]. It contains a presentation of Girard's paradox that does not appear in the final version.

[12] Per Martin-Löf. An intuitionistic theory of types: predicative part. In H. E. Rose and J. C. Shepherdson, editors, *Logic Colloquium '73*, pages 73–118, North Holland, 1975.

[13] Per Martin-Löf. A theory of types. October 1971. Mimeographed manuscript.

[14] Albert R. Meyer and Mark B. Reinhold. 'type' is not a type. In *Conference Record of the Thirteenth Annual ACM Symposium on Principles of Programming Languages*, Association for Computing Machinery, SIGACT, SIGPLAN, 1986.

[15] Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348–375, 1978.

[16] John C. Mitchell and Robert W. Harper. Personal communication.

[17] John C. Mitchell and Gordon Plotkin. Abstract types have existential type. In *Conference Record of the Twelfth Annual ACM Symposium on Principles of Programming Languages*, Association for Computing Machinery, SIGACT, SIGPLAN, 1985.

Appendix

A.1 The equality theory of $\Lambda^{\tau:\tau}$

Reflexive

$$\frac{\mathcal{A} \vdash_{\tau:\tau} M : \sigma}{\mathcal{A} \vdash_{\tau:\tau} M = M} \qquad (\Lambda^{\tau:\tau}7)$$

Symmetric

$$\frac{\mathcal{A} \vdash_{\tau:\tau} M = N}{\mathcal{A} \vdash_{\tau:\tau} N = M} \qquad (\Lambda^{\tau:\tau}8)$$

Transitive

$$\frac{\mathcal{A} \vdash_{\tau:\tau} M = N \quad \mathcal{A} \vdash_{\tau:\tau} N = O}{\mathcal{A} \vdash_{\tau:\tau} M = O} \qquad (\Lambda^{\tau:\tau}9)$$

Left Congruence

$$\frac{\mathcal{A} \vdash_{\tau:\tau} M : \sigma \quad \mathcal{A} \vdash_{\tau:\tau} f : \Pi x \in \sigma \,.\, \tau(x) \quad \mathcal{A} \vdash_{\tau:\tau} f = f'}{\mathcal{A} \vdash_{\tau:\tau} fM = f'M} \qquad (\Lambda^{\tau:\tau}10)$$

Right Congruence

$$\frac{\mathcal{A} \vdash_{\tau:\tau} M : \sigma \quad \mathcal{A} \vdash_{\tau:\tau} f : \Pi x \in \sigma \,.\, \tau(x) \quad \mathcal{A} \vdash_{\tau:\tau} M = M'}{\mathcal{A} \vdash_{\tau:\tau} fM = fM'} \qquad (\Lambda^{\tau:\tau}11)$$

β-reduction

$$\frac{\mathcal{A} \vdash_{\tau:\tau} M : \sigma \quad \mathcal{A} \vdash_{\tau:\tau} \lambda x : \sigma \,.\, N(x) : \Pi x \in \sigma \,.\, \tau(x)}{\mathcal{A} \vdash_{\tau:\tau} (\lambda x : \sigma \,.\, N(x))M = N(M)} \qquad (\Lambda^{\tau:\tau}12)$$

ξ-λ

$$\frac{\mathcal{A} \vdash_{\tau:\tau} \sigma : \mathbf{type} \quad \mathcal{A}[x:\sigma] \vdash_{\tau:\tau} M(x) : \tau(x) \quad \mathcal{A}[x:\sigma] \vdash_{\tau:\tau} M(x) = N(x)}{\mathcal{A} \vdash_{\tau:\tau} \lambda x : \sigma \,.\, M(x) = \lambda x : \sigma \,.\, N(x)} \qquad (\Lambda^{\tau:\tau}13)$$

ξ-Π

$$\frac{\mathcal{A} \vdash_{\tau:\tau} \sigma : \mathbf{type} \quad \mathcal{A}[x:\sigma] \vdash_{\tau:\tau} \tau(x) : \mathbf{type} \quad \mathcal{A}[x:\sigma] \vdash_{\tau:\tau} \tau(x) = \tau'(x)}{\mathcal{A} \vdash_{\tau:\tau} \Pi x \in \sigma \,.\, \tau(x) = \Pi x \in \sigma \,.\, \tau'(x)} \qquad (\Lambda^{\tau:\tau}14)$$

η-λ

$$\frac{\mathcal{A} \vdash_{\tau:\tau} f : \Pi y \in \sigma \,.\, \tau(y)}{\mathcal{A} \vdash_{\tau:\tau} \lambda x \in \sigma \,.\, (fx) = f} \quad \text{if } x \notin FV(f). \qquad (\Lambda^{\tau:\tau}15)$$

Binding type conv. λ

$$\frac{\mathcal{A} \vdash_{\tau:\tau} \sigma : \mathbf{type} \quad \mathcal{A}[x:\sigma] \vdash_{\tau:\tau} M(x) : \tau(x) \quad \mathcal{A} \vdash_{\tau:\tau} \sigma = \sigma'}{\mathcal{A} \vdash_{\tau:\tau} \lambda x \in \sigma \,.\, M(x) = \lambda x \in \sigma' \,.\, M(x)} \qquad (\Lambda^{\tau:\tau}16)$$

Binding type conv. Π

$$\frac{\mathcal{A} \vdash_{\tau:\tau} \sigma : \mathbf{type} \quad \mathcal{A}[x:\sigma] \vdash_{\tau:\tau} \tau(x) : \mathbf{type} \quad \mathcal{A} \vdash_{\tau:\tau} \sigma = \sigma'}{\mathcal{A} \vdash_{\tau:\tau} \Pi x \in \sigma \,.\, \tau(x) = \Pi x \in \sigma' \,.\, \tau(x)} \qquad (\Lambda^{\tau:\tau}17)$$

A.2 The equality theory of $\Lambda^{\exists|\cdot|}$

Reflexive

$$\frac{\mathcal{A} \vdash M : \sigma}{\mathcal{A} \vdash M = M} \qquad (\Lambda^{\exists|\cdot|}16)$$

Symmetric

$$\frac{\mathcal{A} \vdash M = N}{\mathcal{A} \vdash N = M} \qquad (\Lambda^{\exists|\cdot|}17)$$

Transitive

$$\frac{\mathcal{A} \vdash M = N \quad \mathcal{A} \vdash N = O}{\mathcal{A} \vdash M = O} \qquad (\Lambda^{\exists|\cdot|}18)$$

Reflexive

$$\frac{\mathcal{A} \vdash \sigma : \mathbf{type}}{\mathcal{A} \vdash \sigma = \sigma} \qquad (\Lambda^{\exists|\cdot|}19)$$

Symmetric

$$\frac{\mathcal{A} \vdash \sigma = \tau}{\mathcal{A} \vdash \tau = \sigma} \qquad (\Lambda^{\exists|\cdot|}20)$$

Transitive

$$\frac{\mathcal{A} \vdash \sigma = \tau \quad \mathcal{A} \vdash \tau = \rho}{\mathcal{A} \vdash \sigma = \rho} \qquad (\Lambda^{\exists|\cdot|}21)$$

Left congruence λ

$$\frac{\mathcal{A} \vdash M : \sigma \quad \mathcal{A} \vdash f : \Pi x \in \sigma \,.\, \tau(x) \quad \mathcal{A} \vdash f = f'}{\mathcal{A} \vdash fM = f'M} \qquad (\Lambda^{\exists|\cdot|}22)$$

Right congruence λ

$$\frac{\mathcal{A} \vdash M : \sigma \quad \mathcal{A} \vdash f : \Pi x \in \sigma \,.\, \tau(x) \quad \mathcal{A} \vdash M = M'}{\mathcal{A} \vdash fM = fM'} \qquad (\Lambda^{\exists|\cdot|}23)$$

Left congruence Λ

$$\frac{\mathcal{A} \vdash \sigma : \mathbf{type} \quad \mathcal{A} \vdash f : \forall t \,.\, \tau(t) \quad \mathcal{A} \vdash f = f'}{\mathcal{A} \vdash f\sigma = f'\sigma} \qquad (\Lambda^{\exists|\cdot|}24)$$

Right congruence Λ

$$\frac{\mathcal{A} \vdash \sigma : \mathbf{type} \quad \mathcal{A} \vdash f : \forall t \,.\, \tau(t) \quad \mathcal{A} \vdash \sigma = \sigma'}{\mathcal{A} \vdash f\sigma = f\sigma'} \qquad (\Lambda^{\exists}|\cdot|_{25})$$

Left congruence $\mathcal{L}_1$

$$\frac{\mathcal{A} \vdash N : \Sigma x \in \sigma \,.\, \tau(x) \quad \mathcal{A} \vdash M : \Pi x \in \sigma \,.\, \Pi y \in \tau(x) \,.\, \rho(x,y) \quad \mathcal{A} \vdash M = M'}{\mathcal{A} \vdash \mathcal{L}MN = \mathcal{L}M'N} \qquad (\Lambda^{\exists}|\cdot|_{26})$$

Right congruence $\mathcal{L}_1$

$$\frac{\mathcal{A} \vdash N : \Sigma x \in \sigma \,.\, \tau(x) \quad \mathcal{A} \vdash M : \Pi x \in \sigma \,.\, \Pi y \in \tau(x) \,.\, \rho(x,y) \quad \mathcal{A} \vdash N = N'}{\mathcal{A} \vdash \mathcal{L}MN = \mathcal{L}MN'} \qquad (\Lambda^{\exists}|\cdot|_{27})$$

Left congruence $\mathcal{L}_2$

$$\frac{\mathcal{A} \vdash N : \exists t \,.\, \sigma(t) \quad \mathcal{A} \vdash M : \forall t \,.\, \Pi x \in \sigma(t) \,.\, \tau(t,x) \quad \mathcal{A} \vdash M = M'}{\mathcal{A} \vdash \mathcal{L}MN = \mathcal{L}M'N} \qquad (\Lambda^{\exists}|\cdot|_{28})$$

Right congruence $\mathcal{L}_2$

$$\frac{\mathcal{A} \vdash N : \exists t \,.\, \sigma(t) \quad \mathcal{A} \vdash M : \forall t \,.\, \Pi x \in \sigma(t) \,.\, \tau(t,x) \quad \mathcal{A} \vdash N = N'}{\mathcal{A} \vdash \mathcal{L}MN = \mathcal{L}MN'} \qquad (\Lambda^{\exists}|\cdot|_{29})$$

Congruence carrier

$$\frac{\mathcal{A} \vdash M : \exists t \,.\, \sigma(t) \quad \mathcal{A} \vdash M = M'}{\mathcal{A} \vdash |M| = |M'|} \qquad (\Lambda^{\exists}|\cdot|_{30})$$

β-λ

$$\frac{\mathcal{A} \vdash M : \sigma \quad \mathcal{A} \vdash \lambda x \in \sigma \,.\, N(x) : \Pi x \in \sigma \,.\, \tau(x)}{\mathcal{A} \vdash (\lambda x \in \sigma \,.\, N(x))M = N(M)} \qquad (\Lambda^{\exists}|\cdot|_{31})$$

β-Λ

$$\frac{\mathcal{A} \vdash \sigma : \mathbf{type} \quad \mathcal{A} \vdash \Lambda t \in \sigma \,.\, N(t) : \forall t \,.\, \tau(t)}{\mathcal{A} \vdash (\Lambda t \,.\, N(t))\sigma = N(\sigma)} \qquad (\Lambda^{\exists}|\cdot|_{32})$$

Pair red. 1
$$\frac{\mathcal{A} \vdash \langle N_1, N_2 \rangle_{\Sigma x \in \sigma . \tau(x)} : \Sigma x \in \sigma \, . \, \tau(x) \quad \mathcal{A} \vdash M : \Pi x \in \sigma \, . \, \Pi y \in \tau(x) \, . \, \rho(x,y)}{\mathcal{A} \vdash \mathcal{L} M \langle N_1, N_2 \rangle_{\Sigma x \in \sigma . \tau(x)} = M N_1 N_2} \qquad (\Lambda^{\exists} | \cdot | 33)$$

Pair red. 2
$$\frac{\mathcal{A} \vdash \langle \tau, N \rangle_{\exists t . \sigma(t)} : \exists t \, . \, \sigma(t) \quad \mathcal{A} \vdash M : \forall t \, . \, \Pi x \in \sigma(t) \, . \, \rho(t,x)}{\mathcal{A} \vdash \mathcal{L} M \langle \tau, N \rangle_{\exists t . \sigma(t)} = M \tau N} \qquad (\Lambda^{\exists} | \cdot | 34)$$

Carrier red.
$$\frac{\mathcal{A} \vdash \langle \tau, N \rangle_{\exists t . \sigma(t)} : \exists t \, . \, \sigma(t)}{\mathcal{A} \vdash |\langle \tau, N \rangle_{\exists t . \sigma(t)}| = \tau} \qquad (\Lambda^{\exists} | \cdot | 35)$$

ξ-λ
$$\frac{\mathcal{A} \vdash \sigma : \mathbf{type} \quad \mathcal{A}[x : \sigma] \vdash M(x) : \tau(x) \quad \mathcal{A}[x : \sigma] \vdash M(x) = N(x)}{\mathcal{A} \vdash \lambda x \in \sigma \, . \, M(x) = \lambda x \in \sigma \, . \, N(x)} \qquad (\Lambda^{\exists} | \cdot | 36)$$

ξ-Λ
$$\frac{\mathcal{A}[s : \mathbf{type}] \vdash M(s) : \tau(s) \quad \mathcal{A}[s : \mathbf{type}] \vdash M(s) = N(s)}{\mathcal{A} \vdash \Lambda s \, . \, M(s) = \Lambda s \, . \, N(s)} \qquad (\Lambda^{\exists} | \cdot | 37)$$

ξ-Π
$$\frac{\mathcal{A} \vdash \sigma : \mathbf{type} \quad \mathcal{A}[x : \sigma] \vdash \tau(x) : \mathbf{type} \quad \mathcal{A}[x : \sigma] \vdash \tau(x) = \tau'(x)}{\mathcal{A} \vdash \Pi x \in \sigma \, . \, \tau(x) = \Pi x \in \sigma \, . \, \tau'(x)} \qquad (\Lambda^{\exists} | \cdot | 38)$$

ξ-$\forall$
$$\frac{\mathcal{A}[s : \mathbf{type}] \vdash \sigma(s) : \mathbf{type} \quad \mathcal{A}[s : \mathbf{type}] \vdash \sigma(s) = \sigma'(s)}{\mathcal{A} \vdash \forall s \, . \, \sigma(s) = \forall s \, . \, \sigma'(s)} \qquad (\Lambda^{\exists} | \cdot | 39)$$

ξ-Σ
$$\frac{\mathcal{A} \vdash \sigma : \mathbf{type} \quad \mathcal{A}[x : \sigma] \vdash \tau(x) : \mathbf{type} \quad \mathcal{A}[x : \sigma] \vdash \tau(x) = \tau'(x)}{\mathcal{A} \vdash \Sigma x \in \sigma \, . \, \tau(x) = \Sigma x \in \sigma \, . \, \tau'(x)} \qquad (\Lambda^{\exists} | \cdot | 40)$$

ξ-$\exists$
$$\frac{\mathcal{A}[s : \mathbf{type}] \vdash \sigma(s) : \mathbf{type} \quad \mathcal{A}[s : \mathbf{type}] \vdash \sigma(s) = \sigma'(s)}{\mathcal{A} \vdash \exists s \, . \, \sigma(s) = \exists s \, . \, \sigma'(s)} \qquad (\Lambda^{\exists} | \cdot | 41)$$

ξ-$\mathcal{L}_1$
$$\frac{\mathcal{A} \vdash \langle M, N\rangle_{\Sigma x \in \sigma . \tau(x)} : \Sigma x \in \sigma . \tau(x) \quad \mathcal{A} \vdash M = M' \quad \mathcal{A} \vdash N = N'}{\mathcal{A} \vdash \langle M, N\rangle_{\Sigma x \in \sigma . \tau(x)} = \langle M', N'\rangle_{\Sigma x \in \sigma . \tau(x)}} \qquad (\Lambda^{\exists}|\cdot|_{42})$$

ξ-$\mathcal{L}_2$
$$\frac{\mathcal{A} \vdash \langle \tau, M\rangle_{\exists t . \sigma(t)} : \exists t . \sigma(t) \quad \mathcal{A} \vdash \tau = \tau' \quad \mathcal{A} \vdash M = M'}{\mathcal{A} \vdash \langle \tau, M\rangle_{\exists t . \sigma(t)} = \langle \tau', M'\rangle_{\exists t . \sigma(t)}} \qquad (\Lambda^{\exists}|\cdot|_{43})$$

Binding type conv. λ
$$\frac{\mathcal{A} \vdash \lambda x \in \sigma . M(x) : \Pi x \in \sigma . \tau(x) \quad \mathcal{A} \vdash \sigma = \sigma'}{\mathcal{A} \vdash \lambda x \in \sigma . M(x) = \lambda x \in \sigma' . M(x)} \qquad (\Lambda^{\exists}|\cdot|_{44})$$

Binding type conv. Π
$$\frac{\mathcal{A} \vdash \Pi x \in \sigma . \tau(x) : \textbf{type} \quad \mathcal{A} \vdash \sigma = \sigma'}{\mathcal{A} \vdash \Pi x \in \sigma . \tau(x) = \Pi x \in \sigma' . \tau(x)} \qquad (\Lambda^{\exists}|\cdot|_{45})$$

Binding type conv. $\mathcal{L}$
$$\frac{\mathcal{A} \vdash \langle M, N\rangle_{\Sigma x \in \sigma . \tau(x)} : \Sigma x \in \sigma . \tau(x) \quad \mathcal{A} \vdash \sigma = \sigma'}{\mathcal{A} \vdash \langle M, N\rangle_{\Sigma x \in \sigma . \tau(x)} = \langle M, N\rangle_{\Sigma x \in \sigma' . \tau(x)}} \qquad (\Lambda^{\exists}|\cdot|_{46})$$

Binding type conv. Σ
$$\frac{\mathcal{A} \vdash \Sigma x \in \sigma . \tau(x) : \textbf{type} \quad \mathcal{A} \vdash \sigma = \sigma'}{\mathcal{A} \vdash \Sigma x \in \sigma . \tau(x) = \Sigma x \in \sigma . \tau(x)} \qquad (\Lambda^{\exists}|\cdot|_{47})$$

η-λ
$$\frac{\mathcal{A} \vdash f : \Pi y \in \sigma . \tau(y)}{\mathcal{A} \vdash \lambda x \in \sigma . (f x) = f} \quad \text{if } x \notin FV(f). \qquad (\Lambda^{\exists}|\cdot|_{48})$$

η-Λ
$$\frac{\mathcal{A} \vdash f : \forall t . \sigma(t)}{\mathcal{A} \vdash \Lambda t . (f t) = f} \quad \text{if } t \notin FV(f). \qquad (\Lambda^{\exists}|\cdot|_{49})$$

Surjective Pairing 1
$$\frac{\mathcal{A} \vdash N : \Sigma x \in \sigma . \tau(x)}{\mathcal{A} \vdash N = \langle \pi_1 N, \pi_2 N\rangle_{\Sigma x \in \sigma . \tau(x)}} \qquad (\Lambda^{\exists}|\cdot|_{50})$$

Surjective Pairing 2
$$\frac{\mathcal{A} \vdash N : \exists t . \sigma(t)}{\mathcal{A} \vdash N = \langle |N|, \pi_2^2 N\rangle_{\exists t . \sigma(t)}} \qquad (\Lambda^{\exists}|\cdot|_{51})$$