The Computational Behaviour of Girard's Paradox

Douglas J. Howe

TR 87-820
March 1987

Department of Computer Science
Cornell University
Ithaca, NY 14853

[0]This paper is to appear in *The Proceedings of the Second Annual Symposium on Logic in Computer Science*, IEEE, 1987.

The Computational Behaviour of Girard's Paradox*

Douglas J. Howe
Department of Computer Science
Cornell University

Abstract

In their paper *"Type" Is Not a Type*, Meyer and Reinhold argued that serious pathologies can result when a type of all types is added to a programming language with dependent types. Central to their argument is the claim that by following the proof of Girard's paradox it is possible to construct in their calculus $\lambda^{\tau\tau}$ a term having a fixed-point property. Because of the tremendous amount of formal detail involved, they were unable to establish this claim. We have made use of the Nuprl proof development system in constructing a formal proof of Girard's paradox and analysing the resulting term. We can show that the term does not have the desired fixed-point property, but does have a weaker form of it that is sufficient to establish some of the results of Meyer and Reinhold. We believe that the method used here is in itself of some interest, representing a new kind of application of a computer to a problem in symbolic logic.

1 Introduction

For the purpose of studying the effect of the type-of-all-types assumption on programming languages with dependent types, Meyer and Reinhold defined λ^{Π}, a polymorphically typed λ-calculus with dependent types, and

*This work was supported, in part, by NSF grant no. MCS-81-04018.

$\lambda^{\tau\tau}$, the calculus obtained from λ^{Π} by adding the axiom $\tau \in \tau$ (where τ represents the type of all types). They asserted that by following the proof of Girard's paradox [4] they could construct in $\lambda^{\tau\tau}$ a polymorphic fixed-point combinator

$$Y \in \Pi\ T{:}\tau\,.\,(T \to T) \to T$$

such that for any type T and any $F \in T \to T$

$$YTF = F(YTF).$$

They then showed that the existence of such a Y had some important implications for $\lambda^{\tau\tau}$:

- A non-normalizing term is derivable.
- Equality of terms is undecidable.
- $\lambda^{\tau\tau}$ does not conservatively extend base-type theories, so that classical reasoning about base-type objects is not valid in the programming language formed by $\lambda^{\tau\tau}$ together with base-types.

However, they were not able to establish the existence of Y. It appears that the problem may involve so much formal detail that a solution without mechanical assistance is infeasible. A non-trivial proof must be carried out in a formal system in which even the basic notions of logic need to be encoded, and the behaviour under reduction of a very large term must be characterized. (A printout, with no spaces, of the term constructed in our effort is about 40 pages long.)

In [8], Reinhold considered an extension of $\lambda^{\tau\tau}$ that included the natural numbers (together with a recursion combinator) and a dependent sum (or sigma) type constructor. He outlined a proof that by following Martin-Löf's simplification of the proof of the paradox [6], it was possible to obtain in the extension a term Y, of the proper type, such that there exist terms Y_n for $n \geq 0$ with $Y_0 \equiv Y$ and for all n

$$Y_n\,TF = F(Y_{n+1}\,TF).$$

He called a term satisfying this property a *looping combinator*, and conjectured that an analogous construction in $\lambda^{\tau\tau}$ would give a fixed-point combinator.

In this paper, we show that at least one particular proof of Girard's paradox yields a term which is a looping combinator and *not* a fixed-point combinator. The computer plays an essential role in the argument. The Nuprl proof development system [2] was used to deal with the vast amounts of syntactic detail that an analysis of the computational behaviour of the paradox seems to entail. It was straightforward (taking less than two days) to formalize a proof of the paradox in a subset of the Nuprl type-theory that is similar to $\lambda^{\tau\tau}$, using the proof-development tools provided by the system. We show how the proof that the resulting term is a looping but not a fixed-point combinator can be reduced to the verification of a (small) finite number of machine-checkable assertions about reduction. These verifications were carried out using Nuprl's term-manipulation facilities. What we present, then, is not a complete proof, in that the argument depends on the correctness of the Nuprl implementation. More specifically, we prove that if the formal argument carried out in the system constitutes a valid proof in the Nuprl type theory, and if the implementation of the programming language ML [5] is correct insofar as it was used to perform certain reduction sequences and to verify certain simple properties of some Nuprl proofs, then the resulting term has the properties we claim.

Several features of Nuprl played an important role in the construction of the term by allowing us to formalize the paradox at a high level of abstraction. Nuprl's *definition* mechanism permits highly readable notations for basic concepts. The *tactic* mechanism provides for the sound addition of derived inference rules and for the construction of proof-finding programs. Finally, the *extractor* makes it possible to construct terms implicitly; one can represent a mathematical proposition P by a Nuprl type T, construct a proof which resembles a conventional proof of P, and then have the system *extract* from the proof a term $t \in T$ that embodies the computational content of the proposition. Because of these features, the formal proof we constructed, although providing the construction of a massive term, bears a strong resemblance to the conventional proof on which it is based.

Of course, the negative half of our result is based on one particular proof of the paradox. Although it is possible that some other approach would give a different result, it seems that the failure of the fixed-point property results from the essential content of the paradox. It should be straightforward, though, to apply our techniques to alternate proofs. If the fixed-point

property cannot be obtained, then the argument in [7] establishing the failure of conservative extension for $\lambda^{\tau\tau}$ is no longer valid. However, the looping property is enough to prove both the undecidability of equality and the existence of a non-normalizing term.

In Section 2 we present the variant of $\lambda^{\tau\tau}$ that corresponds to the subset of the Nuprl type theory we used. In Section 3, we discuss Girard's paradox and its formalization. In Section 4 we analyze the computational behaviour of the resulting term. At end we make a few concluding remarks.

2 $\nu^{\tau\tau}$

In this section we present a polymorphic λ-calculus $\nu^{\tau\tau}$ that has dependent types and a type of all of types. It is a simple matter to check that the actual Nuprl proof of Girard's paradox that we constructed yields a proof in $\nu^{\tau\tau}$; it seems certain it also yields a proof in $\lambda^{\tau\tau}$ but this has not been verified. Details are contained in an appendix that we have omitted. The chief difference between $\lambda^{\tau\tau}$ and $\nu^{\tau\tau}$ is that λ-abstractions in $\lambda^{\tau\tau}$ contain the type for the bound variable.

The set of terms of $\nu^{\tau\tau}$ is the smallest set containing an infinite set of variables, the constant τ, $a(b)$ whenever a and b are terms, and $\lambda x.\, B$ and $\Pi\, x : A\,.\, B$ whenever x is a variable and A and B are terms. We will write $t[a/x]$ to denote the result of substituting a for x in t. The rules of $\nu^{\tau\tau}$ deal with *sequents*, of the form:

$$x_1 : A_1, \ldots, x_n : A_n \vdash t \in T,$$

where the variables x_i are all distinct, for each i the set of free variables of A_i is contained in $\{x_1, \ldots, x_{i-1}\}$, and the set of free variables of $t \in T$ is contained in $\{x_1, \ldots, x_n\}$. We will call the portion of a sequent to the left of the turnstile a *context*. The extension of a context $\mathcal{A}$ by an assumption $x : A$ or by another context $\mathcal{A}'$ will be denoted by $\mathcal{A}, x : A$ and $\mathcal{A}, \mathcal{A}'$ respectively. We will write $t \rightarrow t'$ if t' can be obtained from t by a sequence of β-reductions.

Following are the the rules of $\nu^{\tau\tau}$. We omit an axiomatization of a theory of equality.

τ-formation $\qquad \mathcal{A} \vdash \tau \in \tau$

Π-*formation*
$$\frac{\mathcal{A} \vdash A \in \tau \quad \mathcal{A}, x{:}A \vdash B \in \tau}{\mathcal{A} \vdash \Pi\, x{:}A\,.\,B \in \tau}$$

λ-*intro*
$$\frac{\mathcal{A} \vdash A \in \tau \quad \mathcal{A}, x{:}A \vdash b \in B}{\mathcal{A} \vdash \lambda x.\, b \in \Pi\, x{:}A\,.\,B}$$

function-elim
$$\frac{\mathcal{A} \vdash f \in \Pi\, x{:}A\,.\,B \quad \mathcal{A} \vdash a \in A}{\mathcal{A} \vdash f(a) \in B[a/x]}$$

assumption
$$\mathcal{A}, x{:}A, \mathcal{A}' \vdash x \in A$$

reduction-1
$$\frac{\mathcal{A} \vdash t \in T' \quad \mathcal{A} \vdash T \in \tau \quad T \to T'}{\mathcal{A} \vdash t \in T}$$

reduction-2
$$\frac{\mathcal{A}, x{:}A', \mathcal{A}' \vdash t \in T \quad A \to A'}{\mathcal{A}, x{:}A, \mathcal{A}' \vdash t \in T}$$

3 Girard's Paradox

Girard's paradox is an adaptation by Girard [4] of the well-known Burali-Forti paradox. The proof of the paradox that we use is somewhat different from Girard's.

Informally, the argument proceeds as follows. Define an ordering (a type together with a transitive binary relation) to be well-founded if it has no infinite descending chains of elements. If the collection of all types is itself a type, then we can form the collection of all well-founded orderings. We make this collection into an ordering using the relation of embedding, and show that this ordering is well-founded. Any well-founded ordering can be embedded in the collection, and so the collection is embedded in itself. This gives an infinite descending chain, contradicting well-foundedness.

Formalizing this argument in $\nu^{\tau\tau}$ involves the *propositions-as-types* correspondence, whereby a proposition is associated with a type of $\nu^{\tau\tau}$ in such a way that the members of the type correspond to constructive proofs of the proposition. In this scheme, universal quantification corresponds to the Π type; most of the other concepts of logic must be coded. Falsity is represented by the type $\Pi\, t{:}\tau\,.\,t$; a member of this type would be a function

taking any type and producing a member of that type. A formalization of the paradox in $\nu^{\tau\tau}$ yields a term that is a member of this type; the outermost reduction sequence starting with this term does not terminate.

What was noticed by Meyer and Reinhold was that given a type T and a function $F \in T \to T$, one could slightly modify the proof of the paradox so that F would be inserted into the looping process that gives the infinite reduction sequence. Using Nuprl [2], a computer system exploiting the propositions-as-types correspondence, we took a (modified) proof of the paradox, made formal definitions for the basic notions of logic and for the other concepts involved, and then used the reasoning facilities of Nuprl to construct a formal version of the informal argument. Nuprl is designed in such a way that the system could then *extract* from the formal derivation the required term, which can be shown in $\nu^{\tau\tau}$ to have the type T. We then analysed this term with the help of Nuprl's symbolic computation facilities. An important point here is that Nuprl made it possible to quickly construct, in a natural way, a formal argument that looks very much like a detailed version of the informal one and that produces, via extraction, the term to be analysed. The complete Nuprl proof comprises eleven lemmas, with an average of about ten steps per lemma. (An example of a step is given later.)

We now give a detailed version of the informal argument above, which can be rather directly translated into Nuprl. We point out the two modifications that are made to the proof of the paradox in order to obtain the looping combinator. After presenting this detailed version, we will very briefly discuss its translation into Nuprl. This translation into Nuprl is sufficiently direct that the interested reader should be able to determine the correspondence between the detailed argument below and the terms discussed in the next section. We assume at the outset that we are given a type $\mathtt{T}$ and a function $\mathtt{F} \in \mathtt{T} \to \mathtt{T}$; in the Nuprl proof, the function and its type are represented by variables, and it is assumed as an axiom that $\mathtt{F} \in \mathtt{T} \to \mathtt{T}$. Also taken as an axiom in the Nuprl proof is the $\tau \in \tau$ assumption.

The definitions of embedding, U, and $<_U$ given below are variants of those given by Coquand [3]. We will not present the (well-known) encodings for logic; as an example, existential quantification is defined by

$$\exists\, x \in A\,.\,B \;\equiv\; \Pi\, C{:}\tau\,.(\Pi\, x{:}A\,.\,B \to C) \to C$$

where we use $A \to B$ to abbreviate $\Pi\, x : A \,.\, B$ when x does not occur free in B.

An *ordering* is a type A together with propositional functions $r \in A \to A \to \mathit{Prop}$ and $d \in A \to \mathit{Prop}$. (We use *Prop* and *Type* as synonyms for τ.) r is to be understood as a relation over A, and d as a subset of A. Typical names for orderings will be (A, r, d) and (B, s, e). We will be somewhat sloppy about notation for function application, writing, e.g., $f(x, y)$ for $f(x)(y)$. We will also abbreviate by writing R for (A, r, d) and S for (B, s, e).

If $f \in A \to B$ and $b \in B$, and

$$\begin{aligned} & e(b) \;\&\; \forall\, x \in A \,.\, d(x) \Rightarrow e(f(x)) \\ & \;\&\; \forall\, x, y \in A \,.\, d(x) \Rightarrow d(y) \\ & \qquad \Rightarrow r(x, y) \Rightarrow s(f(x), f(y)) \\ & \;\&\; \forall\, x \in A \,.\, d(x) \Rightarrow s(f(x), b) \end{aligned}$$

then we will say that (f, b) *embeds* R in S. We will say that the first ordering is *embedded* in the second, and write $R < S$, if such an f and b exist. We will refer to f and b as the *order-preserving map* and the *bound* of the embedding, respectively.

Define transitivity, *trans*(R), in the obvious way. R is *well-founded* if there are no non-empty subsets P of A such that for every y in P there is an x in P with $r(x, y)$. We call such subsets *chains*. More formally, $P \in A \to \mathit{Prop}$ is a chain in R if

$$\begin{aligned} & \exists\, x \in A \,.\, P(x) \;\&\; d(x) \\ & \;\&\; \forall\, y \in A \,.\, P(y) \Rightarrow \exists\, x \in A \,.\, P(x) \;\&\; r(x, y). \end{aligned}$$

Well-foundedness is defined by:

$$\mathit{wf}(R) \;\equiv\; \forall\, P \in A \to \mathit{Prop} \,.\, (P \;\mathit{chain\ in}\; R \to \mathit{false}).$$

The definition of $\mathit{wf}(R)$ is the only one that needs to be changed in order to obtain the looping combinator; we replace *false* by the assumed type `T`.

We now define *Uord*, the collection of all well-founded orderings. Define the type U to be

$$\begin{aligned} & (\Pi\, B : \mathit{Type} \,.\, (B \to B \to \mathit{Prop}) \\ & \qquad \to (B \to \mathit{Prop}) \to \mathit{Prop}) \to \mathit{Prop} \end{aligned}$$

and the injection i by $i(R) \equiv \lambda x\,.\,x(R)$. U can be viewed as the collection of all sets of sets of orderings, and i as the function which associates to each ordering the set of all sets of orderings that contain it. The mapping i is injective with respect to intensional equality, defined by

$$a = b \equiv \forall P \in U \to \mathit{Prop}\,.\,P(a) \Rightarrow P(b).$$

If $i(R) = i(S)$ then we can prove that the two orderings have the same properties. The ordering on U is defined by: $u <_U v$ *iff* there are R and S such that $u = i(R)$, $v = i(S)$, and $R < S$. Finally, define: $d_U(u)$ *iff* there is an R such that $u = i(R)$, $\mathit{trans}(R)$, and $\mathit{wf}(R)$. We will refer to the ordering $(U, <_U, d_U)$ as *Uord*.

Our proof of the paradox consists of eight lemmas, to each of which except the first corresponds a lemma in the Nuprl formalization. The parenthesized numbers in lemma statements below give the number of steps in the corresponding Nuprl proof. There are three additional lemmas in the Nuprl development; they contain a total of ten steps. We will abuse notation by writing R (possibly with subscripts) for (A, r, d). We will also omit type information when it is clear from the context. The proofs of the lemmas below are for the most part straightforward, and so they are omitted or sketched.

Lemma 1 *If $i(R_1) = i(R_2) \in U$ and P is such that $P(R_1)$, then $P(R_2)$.*

Lemma 2 *(9). If $R_1 < R_2$ and (f_1, b_1) embeds R_2 in R_3 then there are f_2 and b_2 which embed R_1 in R_3 and such that $r_3(b_2, b_1)$.*

Lemma 3 *(4). If $d_U(i(R))$ then $\mathit{trans}(R)$ (4) and $\mathit{wf}(R)$.*

Lemma 4 *(8). $\mathit{trans}(\mathit{Uord})$.*

Lemma 5 *(31). $\mathit{wf}(\mathit{Uord})$.*

Proof. Suppose P is a chain in *Uord*. Then there is an R_0 such that $P(i(R_0))$ and $d_U(i(R_0))$. Let Q be

$$\lambda a\,.\,\exists R\,.\,\exists f\,.\,P(i(R))\ \&\ (f, a)\ \mathit{embeds}\ R\ \mathit{in}\ R_0.$$

Q is a chain in R_0. □

The previous two lemmas, together with the assumption $\tau \in \tau$, imply that $d_U(i(Uord))$.

It is the proof of Lemma 5 that is modified for the construction of the looping combinator. The basic idea of this proof is to assume that there is a chain in *Uord* and obtain from it a chain in a *member* of *Uord* (this member is the element asserted to exist in the definition of chain). With the *modified* definition of well-foundedness, to prove this lemma we must give a function which takes chains in *Uord* to members of `T`. Given a chain P in *Uord*, we can obtain a chain Q in some R as in the unmodified proof. Since $wf(R)$, we can get a member x of `T`. Instead of using x directly, we first apply `F` to it.

Denote by R_a the ordering

$$(A, r, \lambda x\,.\, d(x)\ \&\ r(x, a))$$

Lemma 6 *(9). If $d_U(i(R))$ and $a \in A$ is such that $d(a)$, then $d_U(i(R_a))$.*

Lemma 7 *(16). If $d_U(i(R))$ then $R < Uord$.*

Proof. $\lambda a\,.\, i(R_a)$ and $i(R)$ give an embedding. □

Lemma 8 *(13). Contradiction.*

Proof. $\lambda u\,.\,(u = i(Uord))$ is a chain in *Uord*, contradicting 5. □

In the Nuprl version of Lemma 8, instead of obtaining a contradiction, we obtain a member of the type `T`. That is, the complete Nuprl proof of the lemma yields a term t and a proof in $\nu^{\tau\tau}$ that $\vdash t \in$ `T`. We have space here only for a few examples that illustrate the Nuprl version of the preceding argument. For more information on Nuprl, see the Nuprl book [2].

The definitions given above can be directly transcribed into Nuprl. For example, the (modified) well-foundedness predicate is the term

```
λ A r d. ∀P:A->Type. P chain in (A,r,d) => T
```

(which uses other definitions). Also defined is a display form for applications of this predicate. For example, the term which consists of the above

```
1. A: Type
2. q: A->A->Type
3. c: A->Type
4. dU(i(A, q, c))
5. a1: A
6. c(a1)
>> (A,q,(λa. c(a) & q(a,a1))) < (A,q,c)

BY (DIntro ['λw. w'; 'a1'] [] [] ...)
   THEN (DIntro [][][] ...)
   THEN (Reduce ...)

1* 7. x: A
   8. c(x) & q(x,a1)
   >> c(x)

2* 7. x: A
   8. c(x) & q(x,a1)
   >> (q(x))(a1)
```

Figure 1: A step from Lemma 7.

lambda-term applied to three arguments `A`, `r`, and `d` appears on the computer screen as `wf(A,r,d)`.

In Figure 1 is a step from the Nuprl proof of Lemma 7. From top to bottom, the components are: a vertically presented sequent (using >> as a turnstile) with numbered hypotheses; an ML program, called a *tactic*, (the text following the BY) that was entered by the author; and two *subgoal* sequents that were generated by the system as the result of the tactic execution (the subgoals inherit all the hypotheses of the first sequent). We will not explain the operation of the tactic shown in the figure except to note that the effect is roughly to make progress toward proving the asserted embedding by supplying an order-preserving map and an embedding bound.

4 Analysis of the Extracted Term

The Nuprl formalization of lemma 8 yields a term t such that $\vdash t \in \mathtt{T}$. The term we will examine, call it L, is a reduced (via β-reductions) form of t. The term L actually contains constants which denote extractions from other Nuprl theorems. Since there is a linear ordering of theorem dependencies, these constants can be eliminated by successively replacing them by the terms they denote. The actual Nuprl terms to which we apply symbolic computation will retain these constants, since they provide convenient labels for certain subterms of interest, although they are replaced by their denotations when necessary for computation.

We will write $t \xrightarrow{h} t'$ if t reduces to t' via a sequence of *head* β-reductions, and $t \rightarrow t'$ if t reduces to t' via *some* sequence of β-reductions. We write $t \equiv t'$ if t and t' are the same except for the names of bound variables. Two terms are equal if they are identical up to β-conversion.

In what follows, we will be interested in the results of computations in which certain subterms are irrelevant. We will therefore consider terms which contain a special variable ι. We will say that t is an *instance* of t', and write $t \leq t'$, if t can be obtained from t' by replacing some occurrences of ι by another term (possibly different terms for different occurrences). A term will be called *ground* if it does not contain ι. We also generalize the notion of reduction (but equality will only apply to ground terms). When we say that t reduces to t' in a certain way, we mean that for any instance of t there is an instance of t' such that the statement is true in the sense previously defined. To verify $t \rightarrow t'$ using Nuprl's computation facilities, we can proceed as follows. Replace by some distinct Nuprl variable all occurrences of ι in t, apply some Nuprl reduction steps to obtain t'', and check that t'' is an instance of t'.

The Looping Property

The basic idea, due to Meyer and Reinhold, behind the proof that L has the looping property is as follows. The proof of the paradox involves constructing a chain in *Uord* and deriving a contradiction (member of T)

from *Uord*'s well-foundedness. This is reflected in the form of L, which is

$$wf(P^0)(pf^0)$$

where wf is the term extracted from the Nuprl proof of Lemma 5, P^0 is the chain in *Uord* from the proof of Lemma 8, and pf^0 is the term which constitutes a proof that P^0 is a chain. *Uord*'s well-foundedness gives a procedure which when applied to P^0 (together with pf^0), produces a chain P^1 (with proof pf^1) in a *member* of the chain P^0, which, by construction of P^0, is again *Uord*, and then applies this member's well-foundedness proof. This is reflected in the reduction sequence

$$wf(P^0)(pf^0) \rightarrow \mathtt{F}(wf(P^1)(pf^1)).$$

P^1 and pf^1 are similar to P^0 and pf^0, so we can repeat the above, obtaining P^k and pf^k for $k > 1$, and thus obtaining the required L_k.

The problem in turning the above idea into a proof is that one must characterize the reduction behaviour of a massive term. It is likely that the task is not feasible for a human without the aid of a computer. We have solved the problem by reducing it to a small set of machine-checkable assertions.

The proof we give below involves verifying certain properties of certain terms via symbolic computation. Whenever we state that a computation has a certain result, then unless stated otherwise the computation was carried out using utilities of the Nuprl system. We will not completely describe the strategies for reduction that were used; the two main strategies were head reduction, and normalization while holding constant a small number of terms that were extractions from theorems. (A transcript of the session in which these computations were done is available from the author.) In what follows, when confusion might otherwise arise, object variables, i.e., Nuprl variables that are part of the terms being discussed, will have typewriter typeface (e.g., $\mathtt{n}$ instead of n).

Theorem 1 *For each k there is a ground term L_k such that $L_0 \equiv L$ and for all k, $L_k \rightarrow \mathtt{F}(L_{k+1})$.*

Proof. We first make some simple definitions. The first definitions are for pairing and projection: for terms a and b, let $\langle a, b\rangle$ denote $\lambda tf.f(a)(b)$,

and let $\pi_1(a)$ and $\pi_2(a)$ denote $a(\iota)(\lambda xy.\, x)$ and $a(\iota)(\lambda xy.\, y)$, respectively. We assume that the variables in the pairing definition are always named so as to avoid capturing free variables of a and b. We extend the pairing notation to n-tuples by right-associating. Next, denote by eq, id, pf_1, pf_2, and pf_3 the terms $\lambda P.\, \lambda x.\, x$, $\lambda x.\, x$, $\lambda xy.\, \pi_1(y)$, $\lambda uvwxy.\, y$, and $\lambda xy.\, \pi_2(y)$, respectively. Finally, let d and wf denote the terms extracted from the Nuprl proof that $i(Uord)$ is in d_U, and from the Nuprl proof of Lemma 5 (that $Uord$ is well-founded), respectively (actually, these terms are just Nuprl constants denoting other terms, as discussed earlier).

We inductively define (non-ground) terms f^k, p_n^k and s_n such that the following three properties hold for all $k \geq 1$ and $n \geq 0$:

$$wf(\iota)(\langle\langle \iota, p_0^k, d\rangle, f^k\rangle) \rightarrow \mathtt{F}(wf(\iota)(\langle\langle \iota, p_0^{k+1}, d\rangle, f^{k+1}\rangle)),$$

$$f^k(\iota)(p_n^k) \rightarrow \langle \iota, p_{n+1}^k, s_n\rangle,$$

and

$$L \rightarrow \mathtt{F}(wf(\iota)(\langle\langle \iota, p_0^1, d\rangle, f^1\rangle)).$$

Intuitively, the tuple $\langle \iota, p_0^k, d\rangle$ corresponds to the proof that the chain P^k is non-empty (with ι taking the place of $Uord$ and with p_0^k corresponding to the proof that $Uord$ is in P^k), and f^k is a function which takes a member of P^k and returns another member which is smaller in the embedding ordering. In the tuple $\langle \iota, p_{n+1}^k, s_n\rangle$, ι takes the place of this other member, p_{n+1}^k corresponds to the proof that this member is a member of P^k, and s_n corresponds to a proof that this member is smaller than the first member. The three properties above suffice to prove the theorem, since we can apply the third property, and then repeatedly apply the first to obtain L_k, for $k \geq 1$, as a ground instance of

$$wf(\iota)(\langle\langle \iota, p_0^k, d\rangle, f^k\rangle).$$

Define, for t a term,

$$\begin{aligned} \mathcal{S}_0[t] &\equiv \langle \iota, \iota, \iota, \iota, \iota, \iota, eq, eq, id, \iota, t, pf_1, pf_2, pf_3\rangle \\ \mathcal{S}[t] &\equiv \langle \iota, \iota, \iota, \iota, \iota, \iota, eq, eq, id, \iota, \langle \iota, t\rangle, \\ &\qquad \lambda xp.\, \langle \pi_1(p), \iota\rangle, pf_2, pf_3\rangle \end{aligned}$$

Define the terms s_n inductively by $s_0 = S_0[d]$, $s_{n+1} = S[s_n]$. Next, for t, t' terms, denote by $\mathcal{P}_1[t, t']$ the term

$$\langle \iota, \iota, \iota, \iota, eq, \iota, \iota, t, t' \rangle.$$

We define by induction on $k \geq 1$ the terms f^k and p_n^k, and simultaneously show that they satisfy the required properties. For the case $k = 1$, we define by induction on $n \geq 0$ terms

$$p_n^1 \quad \equiv \quad \mathcal{P}_1[op_n, bpf_n],$$

where op_n and bpf_n are terms such that for each $j \geq 0$

$$op_n(\iota)(\iota)(\iota)(\iota)(s_j) \rightarrow s_{n+j+1}$$

and

$$bpf_n(\iota)(d) \rightarrow s_n.$$

Computing, we get

$$L \rightarrow \mathtt{F}(wf(\iota)(t))$$

where t is ground and

$$t \rightarrow \langle \langle \iota, \mathcal{P}_1[op_0, bpf_0], d \rangle, f \rangle$$

for some op_0 bpf_0, and f. Take p_0^1 to be $\mathcal{P}_1[op_0, bpf_0]$, and f^1 to be f. We now verify the required properties of op_0 and bpf_0 with two computations ($\mathtt{sj}$ is a single Nuprl variable):

$$op_0(\iota)(\iota)(\iota)(\iota)(\mathtt{sj}) \rightarrow \mathcal{S}[\mathtt{sj}]$$

and

$$bpf_0(\iota)(\mathtt{d}) \rightarrow \mathcal{S}_0(\mathtt{d}).$$

Suppose that we have defined p_n^1, and that op_n and bpf_n have the required properties. We have

$$\begin{array}{l} f^1(\iota)(\mathcal{P}_1[\mathtt{opn}, \mathtt{bpfn}]) \\ \rightarrow \langle \iota, \mathcal{P}_1[op'_{n+1}, bpf'_{n+1}], \mathtt{bpfn}(\iota)(d) \rangle \end{array}$$

for some terms op'_{n+1} and bpf'_{n+1}. Define p^1_{n+1} to be $\mathcal{P}_1[op'_{n+1}, bpf'_{n+1}]$ with op_n and bpf_n substituted for $\mathtt{opn}$ and $\mathtt{bpfn}$ respectively. By the induction hypothesis, $bpf_n(\iota)(d)$ reduces to s_n, so if we can verify the required properties for p_{n+1} then we will have completed the construction of the terms p^1_n and shown that for all $n \geq 0$

$$f^1(\iota)(p^1_n) \rightarrow \langle \iota, p^1_{n+1}, s_n \rangle.$$

The properties for p^1_{n+1} are verified by computing

$$op'_{n+1}(\iota)(\iota)(\iota)(\iota)(\mathtt{sj}) \rightarrow (\mathtt{opn})(\iota)(\iota)(\iota)(\iota)(\mathcal{S}[\mathtt{sj}])$$

and

$$bpf'_{n+1}(\iota)(\mathtt{d}) \rightarrow (\mathtt{opn})(\iota)(\iota)(\iota)(\iota)(\mathcal{S}_0[\mathtt{d}])$$

and applying the induction hypothesis.

Suppose now that $k \geq 1$ and that the terms f^k and p^k_n have been defined and satisfy

$$f^k(\iota)(p^k_n) \rightarrow \langle \iota, p^k_{n+1}, s_n \rangle. \quad (*)$$

For t, t' terms, denote by $\mathcal{P}[t, t']$ the term

$$\langle \iota, \iota, \iota, id, t, t', pf_1, pf_2, pf_3 \rangle.$$

Define, for all $n \geq 0$, p^{k+1}_n to be

$$\mathcal{P}[p^k_{n+1}, u],$$

where u is d if $n = 0$, and ι otherwise. For any term t denote by $\sigma(t)$ the term

$$t[p^k_0, f^k / \mathtt{pk0}, \mathtt{fk}]$$

Now

$$wf(\iota)(\langle\langle \iota, \mathtt{pk0}, d\rangle, \mathtt{fk}\rangle) \xrightarrow{h} \mathtt{F}(w(\iota)(u))$$

where $w \xrightarrow{h} wf$ and $u \xrightarrow{h} \langle t, f \rangle$. Define f^{k+1} to be $\sigma(f)$. t is

$$\mathtt{fk}(\iota)(\mathtt{pk0})(v)(v').$$

This term, under σ, contains a subterm to which, by induction, $(*)$ can be applied. We have

$$(\langle \iota, \mathtt{pk1}, s_0 \rangle)(v)(v') \rightarrow \langle \iota, \mathcal{P}[\mathtt{pk1}, d], d \rangle.$$

This proves that

$$\sigma(t) \rightarrow \langle \iota, p_0^{k+1}, d \rangle.$$

We must now show that f^{k+1} has the required property. Suppose $n \geq 0$. Let r be $\mathcal{P}[\mathtt{p}, u]$ where u is d if $n = 0$ or ι if $n > 0$, and let r' be $\mathcal{S}[\mathtt{sn}]$. For t a term, let $\tau(t)$ be

$$t[p_0^k, f^k, p_{n+1}^k, s_n / \mathtt{pk0}, \mathtt{fk}, \mathtt{p}, \mathtt{sn}].$$

We must show that

$$\tau(f(\iota)(r)) \rightarrow \langle \iota, p_{n+1}^{k+1}, s_n \rangle.$$

Computing, we get

$$f(\iota)(r) \rightarrow \mathtt{fk}(\iota)(\mathtt{p})(a)(a')$$

for some a and a'. Now by induction

$$\tau(\mathtt{fk}(\iota)(\mathtt{p})) \rightarrow \langle \iota, p_{n+2}^k, s_{n+1} \rangle.$$

By computation,

$$(\langle \iota, \mathtt{q}, r' \rangle)(a)(a') \rightarrow \langle \iota, \mathcal{P}[\mathtt{q}, d], \mathtt{sn} \rangle.$$

□

Not a Fixed-Point

We will now show that L is not an application of a fixed-point combinator. In doing so, we only make one use of Nuprl symbolic computation to establish a fact about reduction; this use will be indicated explicitly. Let L_k, for $k \geq 0$, be the terms constructed as the proof of Theorem 1. Define u^k to be

$$\langle \langle \iota, p_0^k, d \rangle, f^k \rangle.$$

For $k \geq 1$, L_k is an instance of $wf(\iota)(u^k)$. We consider T and F to be constants in what follows.

Figure 2 contains part of a printout of the term wf with some subterms elided, and containing names of theorems (which refer to the corresponding extractions). From it we obtain the term wf' by replacing the (unique) subterm of the form $\lambda \mathtt{y}.\mathtt{F}(t)$ by $\mathtt{X}(\lambda \mathtt{y}.\mathtt{F}(t))$, (X is a fresh variable). Note

```
(λ P v0.
  (v0
    . . .
    (λ x v9.
      (v9
        . . .
            . . .
                (λ d0 y.
                  (F
                    (y
                      (and (trans A0 r0 d0)
                           (wf A0 r0 d0))
                      (λ v3 v4. v4)
                      (wf A0 r0 d0)
                      (λ v3 v4. v4)
                      . . .
                      . . .)))))))))))
```

Figure 2: *wf* with some subterms elided.

that t is headed by y. The idea here is to block certain reductions in order to show that all reductions from L must repeatedly satisfy a certain property. We will call a term t F-*reducible* if it has no subterm of the form $\mathtt{F}(t')$ such that $t \xrightarrow{h} \mathtt{F}(t')$. We will write $\mathtt{F}^{(n)}(t)$ for the term which is the n-fold application of F to t. In the remainder of this section, *all* terms will be *ground* unless indicated otherwise.

Lemma 9 *If* $w \xrightarrow{h} wf$, $u \to u^k$, $k \geq 1$, *and for some terms* c *and* t, $w(c)(u) \to \mathtt{F}^{(i)}(t)$, *where* $i \geq 0$ *and* t *is* F*-reducible, then* $t \xrightarrow{h} \mathtt{F}((\iota)(\iota)(u'))$ *for some* u' *equal to an instance of* u^{k+i+1} *(i.e., where there exists* $u'' \leq u^{k+i+1}$ *with* $u' = u''$*).*

Proof. The proof is by induction on i. We will do the induction step first. Suppose, then, that $i > 0$. By standardization[1] we have

$$w(c)(u) \xrightarrow{h} wf(c)(u) \xrightarrow{h} \mathtt{F}(t')$$

[1]Roughly, for every reduction sequence, there is a left-to-right reduction sequence with the same initial and final terms. For details, see [1].

for some t' with $t' \rightarrow \mathtt{F}^{(i-1)}(t)$. Therefore, to prove the induction step it suffices to show that t' has the form $w'(c')(u')$ where $w' \xrightarrow{h} wf$ and $u' \rightarrow u^{k+1}$.

Nuprl's symbolic computation gives

$$wf'(\iota)(\langle\langle\iota, \mathtt{pk0}, d\rangle, \iota\rangle) \xrightarrow{h} \mathtt{X}(\lambda \mathtt{y}.\, \mathtt{F}(w_0(c_0)(u_0)))(v_0)$$

for some terms w_0, c_0, u_0, and v_0 that are not ground terms, are minimal with respect to $\leq$ (i.e., are obtained by treating ι as a variable, doing the computation, and not making any replacements, in the reduct, of subterms by ι), and do not contain X. It follows that

$$wf'(\iota)(u^k) \xrightarrow{h} \mathtt{X}(\lambda \mathtt{y}.\, \mathtt{F}(w_1(\iota)(u_1)))(v_1)$$

for some terms w_1, u_1, and v_1 that are not ground terms, do not contain X, and are minimal with respect to $\leq$. Note that, since w_1, u_1, and v_1 do not contain X, for any c', w', u', v' and any $u \leq u^k$ such that

$$wf'(c')(u) \xrightarrow{h} \mathtt{X}(\lambda \mathtt{y}.\, \mathtt{F}(w'(\iota)(u')))(v'),$$

w', u' and v' do not contain X. Removing the X and doing the final β-conversion in the reduction defining w_1, etc., we get

$$wf(\iota)(u^k) \xrightarrow{h} \mathtt{F}(w_1[v_1/\mathtt{y}](\iota)(u_1[v_1/\mathtt{y}])).$$

In the proof of Theorem 1, we showed that

$$wf(\iota)(u^k) \xrightarrow{h} \mathtt{F}(w_2(\iota)(u_2))$$

where w_2 and u_2 are not ground, $w_2 \xrightarrow{h} wf$ and $u_2 \rightarrow u^{k+1}$. By the minimality of w_1, u_1, and v_1, we have

$$w_1[v_1/\mathtt{y}] \leq w_2 \quad \textit{and} \quad u_1[v_1/\mathtt{y}] \leq u_2.$$

Now,

$$wf'(c)(u) \rightarrow wf'(c)(u_0^k) \xrightarrow{h} \mathtt{X}(\lambda \mathtt{y}.\, \mathtt{F}(w_3(\iota)(u_3)))(v_3)$$

where u_0^k, w_3, u_3, and v_3 are instances of u^k, w_1, u_1, and v_1, respectively. Standardizing, we get

$$wf'(c)(u) \xrightarrow{h} \mathtt{X}(\lambda \mathtt{y}.\, \mathtt{F}(w_4(\iota)(u_4)))(v_4) \quad (*)$$

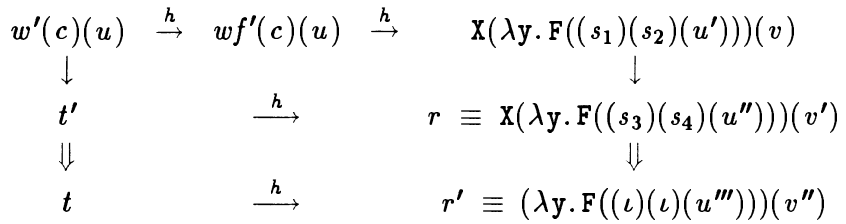

Figure 3: Reductions in the base case argument.

where, since w_4 must be in head normal form by the construction of wf', $w_4 \rightarrow w_3$, $u_4 \rightarrow u_3$, and $v_4 \rightarrow v_3$. Therefore,

$$w_4[v_4/\mathtt{y}] \rightarrow w_3[v_3/\mathtt{y}] \leq w_1[v_1/\mathtt{y}] \leq w_2 \xrightarrow{h} wf.$$

Note that wf was actually defined to be a constant denoting an extraction, so we must have

$$w_4[v_4/\mathtt{y}] \xrightarrow{h} wf.$$

Also,

$$u_4[v_4/\mathtt{y}] \rightarrow u_3[v_3/\mathtt{y}] \leq u_1[v_1/\mathtt{y}] \leq u_2 \rightarrow u^{k+1},$$

so $u_4[v_4/\mathtt{y}] \rightarrow u^{k+1}$. Removing the $\mathtt{X}$ in $(*)$, then, we have

$$w(c)(u) \xrightarrow{h} wf(c)(u) \xrightarrow{h} (\lambda \mathtt{y}.\mathtt{F}(w_4(\iota)(u_4)))(v_4)$$
$$\xrightarrow{h} \mathtt{F}(w_4[v_4/\mathtt{y}](\iota)(u_4[v_4/\mathtt{y}])).$$

This completes the proof of the induction case.

To prove the base case, we must show that if $w \xrightarrow{h} wf$, $u \rightarrow u^k$, and $w(c)(u) \rightarrow t$ where t is $\mathtt{F}$-reducible, then $t \xrightarrow{h} \mathtt{F}((\iota)(\iota)(u'))$, where u' is equal to an instance of u^{k+1}. For v a term, let $\varepsilon(v)$ be the term obtained by repeatedly replacing subterms of the form $\mathtt{X}(a)$ by a.

Figure 3 contains a diagram which summarizes the reductions discussed in the following. Let w' be w with all occurrences of wf replaced by wf'. Using the argument in the induction case, we have

$$w'(c)(u) \xrightarrow{h} wf'(c)(u) \xrightarrow{h} \mathtt{X}(\lambda \mathtt{y}.\mathtt{F}((s_1)(s_2)(u')))(v)$$

where

$$u'[v/\mathtt{y}] \to u^{k+1},$$

and u' and v do not contain $\mathtt{X}$. Now consider the reduction sequence from $w(c)(u)$ to t. Note that if b is such that $w'(c)(u) \to b$, and if for some a and a' $\mathtt{X}(a)(a')$ is a subterm of b, then $a(a')$ is a redex and has no free variables, and if this redex is contracted, the resulting term is head normal (see Figure 2). It follows that we can obtain a reduction sequence from $w'(c)(u)$ to some t' where t can be obtained from t' by replacing maximal subterms of the form $\mathtt{X}(a_1)(a_2)$ or $\mathtt{X}(a_1)$ by some term a_3 where $\varepsilon(a_1(a_2)) \to a_3$ or $\varepsilon(a_1) \to a_3$, respectively. (Such a set of replacements is indicated in the diagram in Figure 3 by a double arrow.) We must also have

$$t' \xrightarrow{h} \mathtt{X}(\lambda \mathtt{y}.\mathtt{F}((s_3)(s_4)(u'')))(v')$$

for some u'' and v' where, using standardization and the fact that s_1 is head-normal, $u' \to u''$ and $v \to v'$. Call the reduct above r. The head reduction sequence from t' to r gives, step for step, a sequence $t \xrightarrow{h} r'$ for some r'. The replacements taking t' to t give rise to a replacement taking r to r'. Suppose that the replacement involves contracting the topmost redex

$$\lambda \mathtt{y}.\mathtt{F}((s_3)(s_4)(u''))(v').$$

Then, since also any subterm of t' of the form $\mathtt{X}(a)(a')$ must be closed, we must have that

$$\mathtt{X}(\lambda \mathtt{y}.\mathtt{F}((s_3)(s_4)(u'')))(v')$$

is a subterm of t', and that the replacements which take t' to t involve replacing this subterm by r'. This contradicts the assumption that t is $\mathtt{F}$-reducible. Therefore the term r' is of the form

$$(\lambda \mathtt{y}.\mathtt{F}((\iota)(\iota)(u''')))(v'')$$

for some u''' and v''. Since u'' and v' do not contain $\mathtt{X}$, and since s_3 is head-normal, we have $u'' \to u'''$ and $v' \to v''$. Now

$$u'[v/\mathtt{y}] \to u''[v'/\mathtt{y}] \to u'''[v''/\mathtt{y}].$$

Since also $u'[v/\mathtt{y}]$ reduces to an instance of u^{k+1}, we are done. □

Lemma 10 *Any term t can be reduced to a term $\mathtt{F}^{(i)}(t')$ where t' is* F*-reducible.*

Proof. The proof is a simple induction on the size of terms. Write t as $\mathtt{F}^{(i)}(t')$ with i maximal, and suppose t' is not F-reducible. Then t' reduces to a term $\mathtt{F}(t'')$ which appears in t'. This term is smaller than t'. □

Theorem 2 *If $1 \leq k < k'$, then $L_k \neq L_{k'}$.*

Proof. Suppose not. Then L_k and $L_{k'}$ have a common reduct t. By Lemma 10 we may assume that t is of the form $F^{(i)}(t')$ where t' is F-reducible. Applying Lemma 9, we get

$$t' \xrightarrow{h} \mathtt{F}((\iota)(\iota)(u)),$$

where u is equal to an instance of u^{k+i+1} and equal to an instance of $u^{k'+i+1}$. This means that there are instances of u^{k+i+1} and $u^{k'+i+1}$ that have a common reduct, which implies that there are instances of p_0^{k+i+1} and $p_0^{k'+i+1}$ that have a common reduct. From the definition of these latter two terms, it follows that there are instances of p_{k+i}^1 and $p_{k+i}^{k'-k+1}$ that have a common reduct. But this is impossible, since the first of these terms is a nine-tuple whose fifth component is *eq*, while the second is a nine-tuple whose fifth component is also a nine-tuple. □

The following is immediate.

Theorem 3 *There is no Z such that $Z = L$ and $Z = \mathtt{F}(Z)$.*

5 Conclusion

We formalized one particular proof of Girard's paradox. It is possible that another proof would yield a term which is a fixed-point combinator, but it seems unlikely. The focus of the looping structure of the paradox seems to be in the proof that *Uord* is well-founded, where, to show that a particular entity does not exist (a chain in our case, a proof that some member is less than itself in the case of Girard's proof), one assumes that it does, transforms it into an analogous entity in a *member* of *Uord*, and

applies that member's well-foundedness property. It seems certain that if the transformation adds any structure, then the fixed-point cannot be obtained.

Nuprl was used in an essential way in this effort. The formal proof of Girard's paradox was carried out at a fairly high level and in a reasonably short time (about two working days), and involved very little explicit reasoning about the components of the term that was being constructed. Nuprl's collection of tools for term manipulation were then used to verify certain properties of the huge resulting term. Nuprl also played a valuable role in the discovery of parts of the argument presented in this paper, since the straightforward formalization of the informal proof yielded a term in whose correctness one could have confidence, and on which one could quickly test certain kinds of conjectures using the various tools provided by the system.

The result of this effort indicates a possible role for mechanized formal problem solving environments such as Nuprl in mathematical research. There may arise other problems such as the one addressed in this paper, where the computer plays an indispensable role in the solution. If this is the case, then systems like Nuprl, that can support high-level formal reasoning in a variety of problem domains, and that provide a large set of facilities for manipulation and analysis of proofs and terms, will prove to be valuable tools.

Acknowledgements

The author is grateful to Bob Constable, Todd Knoblock, and Scott Smith for their suggestions concerning the presentation of this work.

References

[1] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* Volume 103 of *Studies in Logic and the Foundations of Mathematics*, North-Holland, Amsterdam, revised edition, 1984.

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

[3] T. Coquand. An analysis of Girard's paradox. In *First Annual Symposium on Logic in Computer Science*, IEEE, 1986.

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

[5] M. J. Gordon, R. Milner, and C. P. Wadsworth. *Edinburgh LCF: A Mechanized Logic of Computation.* Volume 78 of *Lecture Notes in Computer Science*, Springer-Verlag, 1979.

[6] P. Martin-Löf. An intuitionistic theory of types. 1972. Mimeographed manuscript.

[7] A. R. Meyer and M. 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.

[8] M. Reinhold. Looping with Girard's paradox. 1986. Unpublished manuscript.