RJ 10271 (A0211-090) November 19, 2002
Computer Science

IBM Research Report

A Model of the Untyped Lambda Calculus in Non-Wellfounded Set Theory

Tom Costello
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

A model of the untyped λ calculus in non-wellfounded set theory

Tom Costello

Abstract

We develop a model of the untyped lambda calculus in non-wellfounded set theory. Every term is modeled by a set of pairs. This corresponds to how partial functions are normally represented in set theory. We show that modeling the untyped lambda calculus terms by total functions, or omitting mentioning the undefined term(s) causes the model to collapse to triviality.

This model of the lambda calculus is simple and intuitive, while capturing the recent approaches to modeling recursive domains.

1 Statement of the Issues

A logical theory is not finished once its syntax and rules are fully described. Its *model theory* also plays an important role. Model theory is the study of the relationship between derivability in a syntactic system, and truth in a mathematical structure. Relating a purely formal system to a mathematical model gives us insight and intuitions that are not easily available directly.

Models are often created using the language of set theory. Even when other models exist, e.g. as automata or games, models based on set theory provide a lingua franca, or a common basis from which to view mathematical structures.

In contrast to the trend of creating set theoretic models, we have the example of the untyped lambda calculus. This was, perhaps, the purest of the syntactic systems, which lacked *any* model theory until the early 70's. Scott (1972) suggested a model theory, which has become known as Scott domains. This work was based on the observation that in a particular lattice the lattice of continuous functions on that lattice was isomorphic to the original lattice.

The idea of Scott domains provide a model of the lambda calculus, and deep insight into the important role continuity plays. However, it does not have the flavor of a set theoretic model. For instance, the terms of the lambda calculus are modeled by points in the lattice, when treated as arguments, and as continuous functions, when treated as functions. One would like a presentation where this type ambiguity was avoided.

Since Scott's seminal work, other models have been proposed. We discuss some of these in the related work section.

2 Summary of main results

We develop a natural model of the untyped lambda calculus in non-wellfounded set theory. We model terms in the untyped lambda calculus as partial functions. In the classical tradition, functions are typically modeled in set theory as sets of tuples[1]. This raises a problem when we try to model

[1] The tuple $\langle x, y\rangle$ is usually modeled as $\{x, \{x, y\}\}$, though this is not the only possible choice, nor always the most convenient choice.

self-applicative functions in set theory, as they would thus be modeled by sets that contained themselves. Standard set theory does not contain such sets. In standard set theory, which we take for concreteness as ZFC, all sets are wellfounded, that is, no set contains itself. Non-wellfounded set theory does contain such sets, and thus removes this obstacle.

For instance, consider the function, f, whose domain is itself, and which is the identity on its domain. This is modeled by the set[2] as $x = \{\langle x, x\rangle\}$.

Thus, using the machinery of non-wellfounded set theory to overcome the "type-mismatch" we develop a model of the untyped lambda calculus, where terms are represented as partial functions. We show it is strongly extensional.

Non-wellfounded set theories recently proposed have strong equality conditions. Two sets are said to be equivalent if they are *bisimilar*. Bisimilarity is a notion of equivalence, which captures the intuition that it suffices if the sets look indistinguishable to an observer that traverses the sets, without having any way to globally fix his bearings.

We show that some natural models of the untyped lambda calculus, based on total functions, with a new element $\perp$, are identified with trivial models by this bisimulation relation. That is, the models are identified with the single function that is the identity on itself. These problems may lend support to the form of bisimulation suggested by (Abramsky 1991).

3 Significance and Relevance

The untyped lambda calculus is one of the earliest approaches to computation, and surely one of the most successful, both in applicability and simplicity. Reflecting the purpose of its creators, to develop an alternative to typed systems of logic, it has a non-wellfounded structure. Functions act upon themselves. They may take themselves as arguments, and return themselves as results.

Given this strong bias towards non-wellfoundedness, a model of the lambda calculus reflecting this structure is natural, and agrees with our intuitions.

The lambda calculus is an important structure, due to its central position in computability, and thus provides an important test case for any modeling framework. The results which show that lambda calculus terms can not be modeled as total functions, without going to a new form of equivalence for sets, sheds new light on the role of urelements in non-wellfounded set theory

Recent works have considered the close connection between bisimulation, lambda calculus, domain theory, and non-wellfoundedness. This paper builds on our understanding of how the notions are inter-related. In particular this work provides a simple framework for comparing the notions, that is perhaps better suited for comparative expositions than the earlier category theoretic approaches.

4 Related Work

Non-wellfounded set theory, under the guise that we consider it in this paper, was developed by Forti and Honsell (1983). It reached prominence after Aczel's (1988) book provided a general framework. More recently Barwise and Moss (Barwise and Moss 1996) have greatly expanded the uses and applications of non-wellfounded set theory, as well as providing the notion of the *solution lemma* upon which we base our constructions. They also consider the importance of special final

[2]This set is equal to $y = \{y\}$ if we take the usual definition of pairing.

co-algebras (special refers to the property of using only a small part of the universe of urelements at any time) in constructing greatest fixed points of monotone operators. The relation between special final co-algebras and co-recursion is developed by Moss and Danner (1996).

Models of the lambda calculus were first developed by Scott, based on lattices, and later by Plotkin (1972), based on continuous partial orders, that is, partial orders where every directed sequence has a least upper bound. Our model, will be significantly closer to Scott's rather than Plotkin's, as Scott's model identifies more diverging terms than Plotkin's.

More recently Pitts (1994), and Fiore (1996) have developed models of recursively defined domains, based around the notion of a final co-algebra. This is something like a greatest fixed point, but expressed in category theoretic terms. A co-algebra is a in-equationally defined structure. The final co-algebra is the unique co-algebra that all other co-algebras embed in. We recall that co-algebra homomorphisms may un-identify points. This thus relates to how non-wellfounded sets are constructed. Indeed Fiore's endofunctor construction mirrors the notion of bisimulation as the identity relation on sets.

Pitts (1994) uses categories of domains, (complete algebraic lattices, or complete lattices where every infinite meet of a set S is equal to the meet of a finite subset of S), while Fiore (1996) uses arbitrary categories, but enriches them over continuous partial orders, to achieve the desired models.

The notion of bisimulation was developed independently by several people, at several different times. It has its roots in Ehrenfeucht Mostowski games, and was studied under the name p-morphisms by van Benthem (1976) in connection with modal logic. There it was shown to be the natural equivalence relation for Kripke models—the first order sentences over the accessibility relation that are preserved by bisimulations are those that are equivalent to modal sentences.

Park (1981) developed bisimulation as an equivalence notion for processes. Milner (1980) popularized the notion in connection with processes, in particular in association with the language CCS.

It was introduced as a possible equivalence relation for non-wellfounded sets by Aczel (1988). Abramsky (1991) developed the notions of bisimulation in relation to domains.

5 Bisimulation

We define the notion of bisimulation over a relational structure. Later we will instantiate these relational structures with particular structures that model non-wellfounded sets.

Definition: 1 (relational structure) *A* labeled relational structure *is a triple* $\langle A, R \subset A \times A, l : A \to X \rangle$ *consisting of a set* A*, a binary relation over that set* R*, and a labeling function,* l*, from* A *to a set of labels* X*.*

Definition: 2 *A bisimulation relation between a pair of labeled relational structures,* $\langle A, R \subset A \times A, l : A \to X \rangle$ *and* $\langle A', R \subset A' \times A', l' : A' \to X \rangle$ *is a binary relation,* B*, a subset of* $A \times A'$*, such that, if* $B(b, c)$ *then*

- $l(b) = l(c)$,
- *for every* $b' \in A$*, if* $R(b, b')$ *then there exists an* $c' \in A'$*, such that* $R'(c, c')$ *and* $B(b', c')$*.*
- *for every* $c' \in A'$*, if* $R'(c, c')$ *then there exists an* $b' \in A$*, such that* $R(b, b')$ *and* $B(b', c')$*.*

We say that two labeled relational structures are bisimilar, if there is some bisimulation relation, B, that relates them.

Examples of relational structures that have been studied include Kripke models, transition systems, and, as we see later, systems of equations(Barwise and Moss 1996).

6 Non-wellfounded set theory

We now present some axioms for set theory with urelements. These are stated in the language of first order logic with equality, with a binary predicate of membership $\in$, a unary predicate true of urelements, $\mathcal{U}$, and a function from pairs of sets to sets *new*, and the constant $\emptyset$.

URELEMENTS	$\forall p\ q.(\mathcal{U}(p) \rightarrow \neg(q \in p))$
EXTENSIONALITY	$\forall a\ b.(\forall p.p \in a \iff p \in b) \rightarrow a = b$
PAIRING	$\forall p\ q \exists a.p \in a \wedge q \in a$
UNION	$\forall a \exists b. \forall c\ p.c \in a \wedge p \in c \rightarrow p \in b$
POWER SET	$\forall a \exists b \forall c.(\forall d.d \in c \rightarrow d \in a) \rightarrow c \in b$
INFINITY	$\exists a.\emptyset \in a \wedge (\forall b.b \in a \rightarrow (\exists c.c \in a \wedge c = b \cup \{b\}))$
COLLECTION	$\forall a.\forall p \in a.\exists q.\phi(a,p,q) \rightarrow (\exists b.\forall p \in a.\exists q \in b.\phi(a,p,q))$
SEPARATION	$\forall a \exists b. \forall p.p \in b \iff p \in a \wedge \phi(p,a)$
CHOICE	$\forall a \exists r.r$ is a well order of a
STRONG PLENITUDE	$\forall ab.\mathcal{U}(new(a,b)) \wedge new(a,b) \notin b \wedge$ $(\forall c.c \neq b \rightarrow new(a,b) \neq new(c,b))$

There are various ways to add the Anti-Foundation Axiom (AFA) to this axiomatization. Forti and Honsell (1983) suggest that, for every relational structure $\langle A, R\rangle$, there is a homomorphism onto a transitive set. Aczel's(1988) formulation is close to this, it is stated in term of an operation on graphs and labels. Rather than use these approaches we use a formulation due to Barwise and Moss (1996) based on the *Solution Lemma*.

Definition: 3 *A* general *system of equations is a tuple $\mathcal{E} = \langle X, A, e\rangle$ consisting of a set $X \subseteq \mathcal{U}$, a set $A \subseteq \mathcal{U}$ disjoint from X, and a function $e : X \rightarrow V_{afa}[X \cup A]$, where $V_{afa}[X \cup A]$ is the class of all sets whose transitive closure contain urelements only from $X \cup A$. We present this function as a set of equations.*

X is called the set of indeterminates of $\mathcal{E}$, and A is called the set of atoms of $\mathcal{E}$. For each $v \in X$, the set $b_v = e(v) \cap X$ is called the set of indeterminates on which v immediately depends. Similarly, the set $c_v = e(v) \cap A$ is called the set of atoms on which v immediately depends.

A solution to $\mathcal{E}$ is a function S with domain X satisfying

$$S(x) = \{S(y) : y \in b_x\} \cup c_x,$$

for each $x \in X$.

Definition: 4 ANTI FOUNDATION AXIOM *Every general system of equations has a unique solution S.*

Definition: 5 *Let $A \subseteq \mathcal{U}$, and let $\mathcal{E} = \langle X, A, e\rangle$ and $\mathcal{E}' = \langle X', A, e'\rangle$ be two general systems of equations which use A as their set of atoms.*

1. *An* A-bisimulation relation *between* $\mathcal{E}$ *and* $\mathcal{E}'$ *is a relation* R *on* $X \times X'$ *such that the following conditions hold:*

   (a) *If* $R(x, x')$ *then for every* $y \in e(x) \cap X$ *there exists a* $y' \in e'(x')' \cap X'$ *such that* $R(y, y')$.

   (b) *If* $R(x, x')$ *then for every* $y' \in e'(x') \cap X'$ *there exists a* $y \in e(x) \cap X$ *such that* $R(y, y')$.

   (c) *If* $R(x, x')$, *then* $e(x)$ *and* $e'(x')$ *contain the same atoms. That is,* $e(x) \cap A = e'(x') \cap A$.

2. *We say that the systems are* A*-bisimilar, and write* $\mathcal{E} \equiv \mathcal{E}'$ *if there is an* A*-bisimulation relation between them obeying:*

$$\forall x \in X \exists x' \in X'.R(x, x') \text{ and}$$
$$\forall x' \in X' \exists x \in X.R(x, x').$$

We drop A *when it is clear which set we refer to.*

Proposition: 1 *Every bisimulation relation on general systems of equations* $\mathcal{E} = \langle X, A, e \rangle$ *and* $\mathcal{E}' = \langle X', A, e' \rangle$ *corresponds to an* A*-bisimulation relation.*

Thus the standard notion of bisimulation may be applied to sets, viewed as systems of equations.

Proposition: 2 (Barwise Moss (1996) 7.3) *We can view sets as labeled relational structures, where the points are the elements of the transitive closure, and the labels are the urelements in each element. Identity on sets is then the largest bisimulation relation on sets.*

7 The untyped lambda calculus

The untyped lambda calculus is due to Church (1932, 1941). (Barendregt 1981) is a modern exposition. It is a simple system consisting of a countable set of variables $x_i, i \in \omega$, and a point, ., parentheses $(,)$, and the symbol λ, as well as two relations, $=$ equality and $\rhd$ reduction.

It has, as terms, the smallest set containing the variables, and, if t and t' are terms, then, $(t\ t')$ and $(\lambda x_j.t)$ are terms. t and t' are the immediate sub-terms of $(t\ t')$, and t and x_j are the immediate sub-terms of $(\lambda x_j.t)$. The sub-terms of a term are the terms in the transitive reflexive closure of sub-term. Free variables of a term t, $FV(t)$ are defined as usual. We have relations, $\rhd$ reduction, and $=$ equality. If t and t' are terms then $t \rhd t'$ and $t = t'$ are formulas.

Three conversions on terms are defined, α, β and η conversion. We use i and j for indices of variables, and use $t[t'/x_i]$ to denote the replacement of every occurrence of x_i in t by the term t'.

$\lambda x_i.t \rhd \lambda x_j.t[x_j/x_i]$		α reduction
$\lambda x_i.(t\ t') \rhd t[t'/x_i]$		β reduction
$\lambda x_i.(t\ x_i) \rhd t$	provided $x \notin FV(t)$	η reduction

We also have the trivial reduction $t \rhd t$, and the rule that reduction is transitive, and the rules that reductions can be applied to sub-terms.

Two terms that differ only on the names of variables are said to be α equivalent. They can be reduced to each other by applying α conversions to sub-terms. A term is said to be in normal form if it cannot be reduced by β or η reduction.

If a term t can be reduced to a term t' in normal form by a series of reductions, then every term t'' in normal form that t can be reduced to, is α equivalent to t'. This is the Church Rosser property.

The standard rules for equality of terms are obeyed. In particular, if one term reduces to another, then they are equal.

$t \rhd t'$ implies $t = t'$	$t = t$
$t = t'$ implies $t' = t$	$t = t', t' = t''$ implies $t = t''$
$t = t'$ implies $t\ t'' = t'\ t''$	$t = t'$ implies $t''\ t = t''\ t'$
if $t = t'$ is derivable then so is, $\lambda x.t = \lambda x.t'$	

We write $\lambda \vdash t = t'$ if $t = t'$ is derivable the above rules.

Definition: 6 (applicative system) *An applicative system is a structure $\mathfrak{M} = \langle X, \cdot \rangle$, where $\cdot$ is a binary operation on X, called application.*

The set of terms $T(\mathfrak{M})$ over $\mathfrak{M}$, on variables $a_i, i \in \omega$, is defined as the smallest set containing the variables, distinguished constants c_a for each element $a \in X$, and which is closed under juxtaposition, $A, B \in T(\mathfrak{M})$ implies $(AB) \in T(\mathfrak{M})$. Juxtaposition denotes application, thus we write $A \cdot B$ as (AB), and parentheses are omitted to denote left association.

Definition: 7 (combinatory algebra) *A* combinatory algebra *is an applicative system $\mathfrak{M}$ such that $\mathfrak{M}$ is not trivial, (i.e. $\|X\| > 1$), and for each term A over $\mathfrak{M}$, with variables among $y_1 \ldots y_n$, we have in $\mathfrak{M}$,*

$$\exists f\ \forall y_1 \ldots, y_n.\ f y_1 \ldots y_n = A \quad \textit{(Combinatory Completeness)}$$

By equality of terms, we mean equality of all substitutions for variables, by elements of X. A combinatory algebra $\mathfrak{M}$ is extensional if in addition in $\mathfrak{M}$,

$$\forall x (fx = f'x) \to f = f' \ \textit{(Extensionality)}$$

We write $\mathfrak{M} \models t = t'$ if $t = t'$ is true in the structure $\mathfrak{M}$.

Proposition: 3 *Let $\mathfrak{M} = \langle X, \cdot \rangle$ be an applicative system such that for some $k, s \in X$ one has in $\mathfrak{M}$,*

$$k \neq s, \quad k\ x\ y = x, \quad s\ x\ y\ z = x\ z(y\ z)$$

then $\mathfrak{M}$ is a combinatory algebra.

Definition: 8 (pre-λ-algebra) *A* pre-λ-algebra *is a combinatory algebra $\mathfrak{M}$ together with an association of terms $A \in T(\mathfrak{M})$ with terms $\lambda^* x.A \in T(\mathfrak{M})$, such that*

- *x does not occur in $\lambda^* x.A$,*
- *$\mathfrak{M} \models (\lambda^* x.A)x = A$.*

Definition: 9 *A λ-algebra, or* model of the λ-calculus *is a pre-λ-algebra $\mathfrak{M}$ such that $\lambda \vdash M = N \Rightarrow \mathfrak{M} \models M = N$.*

A λ-algebra $\mathfrak{M}$ is extensional *iff $\mathfrak{M}$ satisfies $\forall x (f\ x = g\ x) \to f = g$.*

Proposition: 4 *A combinatory algebra satisfying extensionality is an extensional λ-algebra.*

There is only one way to define abstraction.

A λ-algebra is called *sensible* iff it equates all unsolvable terms, there is a unique maximal sensible λ-algebra. These are the models that we consider later. We have now defined what a model of the lambda calculus is.

8 Models of the untyped lambda calculus

Functions are often modeled in set theory as a set of tuples. The function f, is then identified with the set $\{\langle x, f(x)\rangle\}$, where $\langle x, y\rangle$ denotes some kind of pairing operation, and the x's range over the domain of f. The most normal pairing operation is, $\{x, \{x, y\}\}$, which we use here, but others are possible. Using this idea we now model the lambda calculus.

8.1 Partial Functions

We consider the system of equations T_T defined as follows. We take an enumeration of all the lambda calculus terms in normal form, plus a lambda calculus term without a normal form denoted $\perp$, which for concreteness we take as $(\lambda x.(x\ x))(\lambda y.(y\ y))$. We associate a urelement v_t with each normal form t, and we associate $v_\perp$ with $\perp$.

If the term $(t\ t')$, has a normal form, we write $V(t\ t')$, for the urelement associated with that normal form, $V(t\ t')$ is undefined if $t\ t'$ does not have a normal form. $V(\perp\ t)$ is always undefined, but $V(t\ \perp)$ may not be.

We can now define a system of equations T_T as follows. It has as indeterminates the urelements $\mathcal{V} = \{v_t | t \text{ is a term in normal form or } \perp\}$. Its set of atoms is the empty set. Its function e, is defined by the following equations:

$$e(v_\perp) = \{\}, \quad e(v_{t_i}) = \{\langle v_{t_j}, V(t_i\ t_j)\rangle | V(t_i\ t_j) \text{ is defined. } \}$$

Let $T_T = \langle V, \emptyset, e\rangle$. Let $S(T_T)$ be the solution to T_T, viewed as a set.

Lemma: 1 *T_T is a system of equations.*

Proof: We need to show that the atoms and indeterminates of T_T are disjoint. This is immediate as the atoms are empty. We need to show that the equations uniquely specify a function e, from X to $V_{\text{afa}}[X \cup A]$. We have two cases to check. $e(v_\perp)$ is immediate as it is the $\emptyset$.

The second case involves showing that $\{\langle t_j, V(t_i\ t_j)\rangle | V(t_i\ t_j) \text{ is defined. } \}$ is a set, whose transitive closure has all its urelements in X. This is immediate as v_{t_j} and $V(t_i\ t_j)$ are always in X, when $V(t_i\ t_j)$ is defined. ∎

Lemma: 2 *$S(T_T)$ is not bisimilar to $S(x = \{\langle x, x\rangle\})$, or $S(x = \{x\})$, in particular $v_\perp \neq v_{\lambda x.x}$.*

Proof: We note that $v_\perp = \{\}$, while $v_{\lambda x.x}$ is non-empty. ∎

Theorem: 1 *$\langle S(T_T), \lambda x\ y.$ if $\exists z.\langle y, z\rangle \in x$ then z else $\{\}\rangle$ is a model of the untyped lambda calculus.*

Proof Sketch: We show that $\langle S(T_T), \lambda x\ y.$ if $\exists z.\langle y, z\rangle \in x$ then z else $\{\}\rangle$ is an applicative structure, and we use proposition 3 to show that it is a combinatory algebra. Finally we note that bisimulation gives us extensionality, and thus by proposition 4 it is a model of the λ calculus.

It is an applicative structure as it is a set and a binary operation. We now show it is a combinatory algebra by showing combinatory completeness.

Let $k = S((\lambda x.(\lambda y.x)))$ and $s = S((\lambda x.(\lambda y.(\lambda z.((xz)(yz))))))$. We show that these s and k behave as required. We show that $k\ x\ y = x$ and $s\ x\ y\ z = ((xz)(yz))$, where application is defined by our binary operation. This is immediate as V is defined in terms of application, and diverging terms are identified.

We need to show that $s \neq k$, consider $\lambda x.x$, which we call i and $\perp$ our diverging term. Then $k\ i\ \perp \not\equiv s\ i\ \perp\ i$, as $k\ i \perp = i$ and $s\ i\ \perp\ i = i\ i(\perp\ i) = i\ \perp = \perp$, and $i \not\equiv \perp$, as $\perp$ is the empty set, while by lemma 2 $\lambda x.x$ is not. Thus we have shown that $s\ i\ \perp\ i \neq k\ i\ \perp$. Further, $k\ i\ \perp\ i = k\ i\ \perp$, as $k\ i\ \perp$ is the identity, so $s\ i\ \perp\ i \neq k\ i\ \perp\ i$, and thus s and k are distinct.

Finally bisimulation identifies extensionally equivalent structures, as extensionally equivalent structures are bisimilar. Consider two terms t and t'. Suppose that $t = t'$. By this we mean that for all assignment of closed terms to the variables of t and t', $t = t'$. We show under the same interpretation that $(\lambda x.t)x = (\lambda x.t')x$. This follows immediately as V is defined by application. ∎

8.2 Without $\perp$

We now consider what happens if we attempt to remove $\perp$ from the model. We build a model that omits all mention of divergence, and we show that it collapses.

We take a countable set of urelements, which we associate with the lambda calculus terms in normal form. We call these urelements, u_t, where u_t is associated with the normal form term t. For purposes of enumeration, we subscript the terms in normal form with indices, and let the set of these urelements be $\mathbf{U}$.

If the term $(t\ t')$, has a normal form, we write $U(t\ t')$, for the urelement associated with that normal form, $U(t\ t')$ is undefined otherwise.

We consider the system of equations $T_{\tilde{\perp}}$ defined as follows. We take the set of indeterminates as the u_t's, we have no atoms, and we define e' by, $e'(u_{t_i}) = \{\langle u_{t_j}, U(t_i\ t_j)\rangle | U(t_i\ t_j) \text{ is defined}\}$. Thus $T_{\tilde{\perp}} = \langle \mathbf{U}, \emptyset, e'\rangle$.

Lemma: 3 *$T_{\tilde{\perp}}$ is a system of equations.*

Proof: Each of the requirements are met. ∎

Proposition: 5 *$S(T_{\tilde{\perp}})$ is bisimilar to $S(x = \{x\})$.*

Proof: We note that all elements can be identified, as there are no urelements to clash, and every set corresponding to a non-diverging term is non-empty. ∎

Proposition: 6 *$\langle S(T_{\tilde{\perp}}), \lambda x\ y.$ if $\langle y, z\rangle \in x$ then $z\rangle$ is the trivial applicative structure.*

Proof: Immediate from lemma 5. ∎

Thus, if we do not mention the diverging terms, then our model collapses.

8.3 With $\perp$

We saw that if we drop all mention of the diverging terms our model collapses. We now attempt to model the untyped lambda calculus by total functions, so that we can model application in the simple way. That is, we now wish to model application by $\lambda xy.$ if $\exists z.\langle y, z\rangle \in x$ then z, as opposed to our previous method, which never included pairs with an undefined second element. Rather, we treated the case where second element would have been undefined separately. Thus we now wish to model all diverging terms by the set $x = \{\langle t, x\rangle |$ for all $t\}$.

We consider the system of equations $T_{\perp}$ defined as follows. We take an enumeration of all the lambda calculus terms in normal form, plus a diverging lambda calculus term denoted $\perp$. We define

$\perp$, for concreteness as $\lambda x.(\lambda y.(yy))(\lambda z.(zz))$. This term diverges on all inputs. This will act as our undefined term. We call the set of urelements, including $w_\perp$, $\mathcal{W}$.

If the term $(t\ t')$, has a normal form, we write $W(t\ t')$, for the urelement associated with that normal form, $W(t\ t')$ is $\perp$ otherwise. Let e'' be, $e''(v_{t_i}) = \{\langle v_{t_j}, W(t_i\ t_j)\rangle\}$. Let $T_\perp = \langle \mathcal{W}, \emptyset, e''\rangle$.

Lemma: 4 *$T_\perp$ is a system of equations.*

Proof: All conditions are met. However, note that W is not r.e. for the first time. ∎

Proposition: 7 *$S(T_\perp)$ is bisimilar to $S(x = \{x\})$.*

Proof: Exactly as lemma 5. ∎

Proposition: 8 *$\langle S(T_\perp), \lambda x\ y.$ if $\exists z.\langle y, z\rangle \in x$ then $z\rangle$ is the trivial applicative structure.*

Proof: Immediate from lemma 7. ∎

Thus we see that if we model terms by total functions, or if we model terms by partial functions without including the totally undefined function, our model collapses to the trivial non-wellfounded set.

A possible solution, that would prevent this collapse, is to look for a form of bisimulation that treats divergence in a special way. *Partial Bisimulation* has this notion, but it seems difficult to relate to set theoretic approaches. Approaches to bisimulation that consider only the "finitely observable part" of a bisimulation have been considered by Abramsky (1991), and may be more appropriate here.

9 Conclusion

Our model of the untyped lambda calculus is simple, perhaps even naive. This is as it should be. The functions definable in the lambda calculus are modeled by sets of pairs, exactly as functions are usually represented in set theory.

Church (1932) developed the lambda calculus as a replacement for the typed systems of logic and set theory that Russell (1935) and his followers had proposed as a solution to the paradoxes. This pursuit also fell victim to paradoxes (Klenne and Rosser 1935). The restricted version we now use was the fragment that Church and Rosser later proved free from contradiction.

This work suggests that the novelty that Church was pursuing was non-wellfoundedness. At the very least it seems that non-wellfoundedness was part of his vision. An interesting possibility for further study is thus the relationship between non-wellfoundedness and illative logic (Barendregt et al. 1993).

References

Abramsky, S. 1991. A Domain Equation for Bisimulation. *Information and Computation* 92:161–218.

Aczel, P. 1988. *Non-Well-Founded Sets.* Vol. 14 of *Lecture Notes.* CSLI.

Barendregt, H. 1981. *The Lambda Calculus: Its syntax and semantics.* North-Holland Publishing Company.

Barendregt, H., M. Bunder, and W. Dekkers. 1993. Systems of illative combinatory logic complete for first order propositional and predicate calculus. *Journal of Symbolic Logic* 58(3):769–788.

Barwise, J., and L. Moss. 1996. *Vicious Circles.* CSLI.

Benthem, J. v. 1976. *Modal Correspondence theory.* PhD thesis, Mathematical Institute, University of Amsterdam. See also *Correspondence Theory*, Handbook of philosophical logic, Vol. II, Reidel, Dordrecht 1984.

Church, A. 1932. A set of postulates for the foundation of logic. *Annals of Math. (2)* 33:346–366.

Church, A. 1941. *The Calculi of Lambda Conversion.* Princeton Univ. Press.

Fiore, M. 1996. A coinduction principle for recursive data types based on bisimulation. *Information and Computation* 186–198.

Forti, F., and F. Honsell. 1983. Set theory with Free Construction Principles. Technical Report 10:493-522, Annali Scuola Normale Supeiore di Pisa.

Klenne, S., and B. Rosser. 1935. The inconsistency of certain formal logics. *Annals of Mathematics* 36(2):630–636.

Milner, R. 1980. *A Calculus of Communicating Systems, LNCS 92.* Springer-Verlag.

Moss, L., and N. Danner. 1996. On the Foundations of Corecursion. Technical report, Indiana University, Bloomington.

Park, D. 1981. Concurrency and automata on infinite sequences. In *Proc. Theoretical Computer Science, LNCS 104*, 167–183. Springer-Verlag.

Pitts, A. 1994. A co-induction principle for recursively defined domains. *Theoretical Computer Science* 124:195–219.

Plotkin, G. 1972. A set theoretical definition of application. Scholl of A.I. Memo MIP-R-95, University of Edinburgh.

Russell, B., and A. Whitehead. 1935. *Principia Mathematica.* Cambridge University Press. Third edition.

Scott, D. 1972. Continuous lattices. In *Proc. 1971 Dalhousie Conference: LNM 274*, 97–136. Springer-Verlag.