Department of Computer & Information Science

Technical Reports (CIS)

University of Pennsylvania *Year* 1988

Combining Algebra and Higher-Order Types

Val Tannen
University of Pennsylvania, val@cis.upenn.edu

COMBINING ALGEBRA AND HIGHER-ORDER TYPES

Val Breazu-Tannen

MS-CIS-88-21
LINC LAB 107

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104

March 1988

Acknowledgements: This research was supported in part by DARPA grant NOOO14-85-K-0018, NSF grants MCS-8219196-CER, IRI84-10413-AO2 and U.S. Army grants DAA29-84-K-0061, DAA29-84-9-0027.

Combining Algebra and Higher-Order Types

*Val Breazu-Tannen**

Department of Computer and Information Science
University of Pennsylvania

March 23, 1988

Abstract. We study the higher-order rewrite/equational proof systems obtained by adding the simply typed lambda calculus to algebraic rewrite/equational proof systems. We show that if a many-sorted algebraic rewrite system has the Church-Rosser property, then the corresponding higher-order rewrite system which adds simply typed β-reduction has the Church-Rosser property too. This result is relevant to parallel implementations of functional programming languages.

We also show that provability in the higher-order equational proof system obtained by adding the simply typed β and η axioms to some many-sorted algebraic proof system is effectively reducible to provability in that algebraic proof system. This effective reduction also establishes transformations between higher-order and algebraic equational proofs, transformations which can be useful in automated deduction.

To appear in the proceedings of the Symposium on LOGIC IN COMPUTER SCIENCE
Edinburgh, July 1988

*Supported in part by U.S. Army Research Office Grant DAAG29–84–K–0061

1 Introduction

This paper presents two results in the area of "comparative anatomy" of type disciplines. Both results describe relationships between first-order features which can be modeled with algebraic equations and the higher-order paradigm modeled by the simply typed lambda calculus.

Our first concern is with the interaction between "first-order computation" modeled by algebraic rewriting, and "copy-rule computation" modeled by β-reduction of lambda terms. If no type discipline is imposed on lambda terms, it is known that this interaction causes new, and negative, effects: Klop has shown that the *untyped lambda calculus* enriched with surjective pairing reduction does not have the Church-Rosser (CR) property (see [Klop 1980], or [Barendregt 1984], pp. 403–407; the proof uses Turing's fixed point combinator), even though the rewrite system consisting of the surjective pairing rules alone is Church-Rosser, and, of course, β-reduction in isolation is CR. Another such counterexample can be adapted from [Breazu-Tannen & Meyer 1987a]:

Consider the following algebraic rewrite system, call it γ:

$$\begin{aligned} cond\ 0\ x\ y &\longrightarrow x \\ cond\ 1\ x\ y &\longrightarrow y \\ minus\ x\ x &\longrightarrow 0 \\ minus\ (succ\ x)\ x &\longrightarrow 1\ . \end{aligned}$$

This system is clearly strongly normalizing (terminating, noetherian) since all rules are length decreasing. It is also easy to check that γ has the *weak* CR property (is locally, or weakly, confluent). Thus, by a well-known result of [Newman 1942] (for a simple proof see [Huet 1978]), γ is CR. However, the CR property *fails* for $\beta\gamma$-reduction on untyped lambda terms constructed from the constants *cond*, *minus*, *succ*, 0 and 1. Indeed, let $F \stackrel{\text{def}}{=} \lambda f.\lambda x.succ\ (f\ x)$ and let P be a fixed point of F (*i.e.*, , $P \stackrel{\beta}{\twoheadrightarrow} F\ P$, see [Barendregt 1984]. p. 131). We have $P \stackrel{\beta}{\twoheadrightarrow} \lambda x.\ succ\ (P\ x)$, and thus

$$\begin{aligned} &cond\ (minus\ (P\ 0)\ (P\ 0))\ 0\ 1 \stackrel{\beta}{\twoheadrightarrow} cond\ (minus\ ((\lambda x.\ succ\ (P\ x))\ 0)\ (P\ 0))\ 0\ 1 \stackrel{\beta}{\longrightarrow} \\ &\stackrel{\beta}{\longrightarrow} cond\ (minus\ (succ\ (P\ 0))\ (P\ 0))\ 0\ 1 \stackrel{\gamma}{\longrightarrow} cond\ 1\ 0\ 1 \stackrel{\gamma}{\longrightarrow} 1\ . \end{aligned}$$

But the same term also reduces

$$cond\ (minus\ (P\ 0)\ (P\ 0))\ 0\ 1 \stackrel{\gamma}{\longrightarrow} cond\ 0\ 0\ 1 \stackrel{\gamma}{\longrightarrow} 0\ ,$$

and, obviously, 0 and 1 are distinct $\beta\gamma$-normal forms.

Both these counterexamples exploit the capability of expressing fixed point combinators in the untyped lambda calculus. Because of the normalization property, no such combinators can be expressed in the simply typed lambda calculus ($\lambda^{\rightarrow}$). And, in fact, we make essential use of the normalization property to show, in section 2, that combining any many-sorted algebraic rewrite system which is CR with $\lambda^{\rightarrow}$ gives a rewrite system which is also CR.

Specifically, given a many-sorted signature Σ, we construct simply typed lambda terms with the sorts of Σ as base (ground) types and from the symbols in Σ seen, by currying, as higher-order constants. We will call these terms $\lambda^{\rightarrow}\Sigma$-terms, and the algebraic Σ-terms—$\alpha\Sigma$-terms. More generally, α-tagged "things" (*e.g.*, proofs) are algebraic. Note that any $\alpha\Sigma$-term can be seen, by currying, as a $\lambda^{\rightarrow}\Sigma$-term. In fact, when the prefixed Polish notation is used for algebraic terms and application is assumed to associate to the left in writing lambda terms, this passage goes unnoticed.

Then, given a set R of rewrite rules between $\alpha\Sigma$-terms, we show that if R is CR on $\alpha\Sigma$-terms, then βR is CR on $\lambda^{\rightarrow}\Sigma$-terms [1]. (Of course, R-rewriting on $\lambda^{\rightarrow}\Sigma$-terms is such that the variables occurring in the algebraic rules can be instantiated with any $\lambda^{\rightarrow}\Sigma$-terms, as long as they are of the same (ground) type as the variables they replace.)

We compare this result with those of [Toyama 1987] and [Klop 1980]. Toyama shows that the direct sum of two CR algebraic rewriting systems is also CR. For the direct sum, the two components are required to have disjoint signatures. In our case, note that while the symbols of the algebraic signature do not play any special role in defining β-reduction, there is one "operation" which is implicit in algebraic rewriting and which is therefore shared with β-reduction, namely *application*, and indeed, Toyama's methods do not seem to help in this situation. Our putting together of an algebraic rewrite system and a lambda calculus is more like Klop's direct sum of *combinatory reduction systems* for which, as shown in [Klop 1980], preservation of the CR property fails, in general, (see the examples above). Klop proves preservation of CR under certain restrictions, but he keeps the untyped lambda calculus as one of the components and imposes the restrictions on the algebraic reduction rules. In contrast, our algebraic reduction rules are totally arbitrary, but we restrict the lambda terms using the simple type discipline. [2]

Our result about CR preservation is relevant for the problem of parallel implementation of functional programming languages (see [Hudak 1986] for a survey). Since it guarantees that results are independent of the computational strategy, the Church-Rosser property is the theoretical foundation for parallel evaluation. For functional languages based on the untyped lambda calculus (such as SCHEME [Abelson & Sussman 1985]) CR depends on the choice of the first-order computational rules. Useful optimizations such as $(x - x) \longrightarrow 0$ and $(succ(x) - x) \longrightarrow 1$ (see the counterexample above) or $(\mathbf{if}\ b\ \mathbf{then}\ x\ \mathbf{else}\ x) \longrightarrow x$ (see [Klop 1980]) are ruled out. Our result shows that, in contrast, *strongly typed* functional languages (such as ML [Gordon *et al.,* 1979] and Miranda [Turner 1985]) are completely flexible from this point of view. Beware: even typed functional languages feature *recursion* which causes the failure of CR just like the untyped fixed points do. The difference is that in typed languges the use of recursion can be effectively isolated and one

[1] The reader may wonder what happens with η-reduction. Even though, of course, η-reduction is CR both in isolation, and together with β-reduction, combining it with CR algebraic rewrite systems does not always yield a CR system: consider the algebraic reduction rule $f\,x \xrightarrow{r} a$. Then, $f \xleftarrow{\eta} \lambda x.\, f\,x \xrightarrow{r} \lambda x.\, a$. Nonetheless, given the absence of a perspicuous computational interpretation for η-reduction we do not regard this failure as significant. However, η, regarded as an *equational axiom*, may be useful when reasoning about programs and thus we take it into consideration when we study the interaction between equational proof systems.

[2] In the presence of types, the surjective pairing rules must be postulated for every pair of types, which takes us out of the framework of algebraic rewrite systems. Nonetheless, it is still true that simply typed lambda calculus with product types and surjective pairing has the CR property [Pottinger 1981]. This result also follows from the fact that the typed lambda calculus with surjective pairing is strongly normalizing [Lambek & Scott 1986] (I also know of two unpublished proofs of this fact, obtained independently: [Bercovici 1984] and [Dougherty 1986]), as above, by Newman's result, since it is easy to check the weak CR property.

can identify the chunks of program for which CR holds and parallel-execute them. This is much more difficult in untyped languages where non-typable "hacks" may hide the failure of CR.

In section 3, we study again the interaction between algebra and higher-order types but from the point of view of equational theories. In the *pure* simply typed lambda calculus, every term has a unique $\beta\eta$-normal form and this implies the decidability of simply typed $\beta\eta$-conversion. Algebraic equational theories can, of course, be undecidable. However, it is reasonable to hope that when combined with the simply typed lambda calculus, the algebraic theories would be the sole potential source of undecidability in the resulting systems, that is, the interaction itself does not introduce new and negative effects. [3] We show that indeed this is the case, and, moreover, we establish effective translations between equational *proofs* in the combined higher-order system and corresponding algebraic equational proofs, and conversely.

Specifically, we are given a many-sorted signature Σ, and a set E of equations between $\alpha\Sigma$-terms, regarded as *axioms*, that is, we are given a presentation of a many-sorted algebraic theory. Then, we consider the higher-order equational proof system which proves equalities between arbitrary $\lambda^{\rightarrow}\Sigma$-terms using β, η and the equations in E as axioms. (Again, the variables in E can be instantiated by lambda terms, of course.) This determines a simply typed lambda theory. We describe an algorithm which computes for every pair (M, N) of $\lambda^{\rightarrow}\Sigma$-terms of the same type, a finite non-empty set $\{L_1, \ldots, L_n\}$, where each L_i is a finite non-empty set of equations between $\alpha\Sigma$-*terms* and we show that $M = N$ is provable in the higher-order proof system if and only if for some i, $1 \leq i \leq n$, all the equations in L_i are provable in the algebraic proof system. (Each of the L_i's corresponds to a possible way of proving $M = N$ using algebraic equations.) It follows that the combined lambda theory is Turing-reducible to (recursive in) the algebraic theory, and thus if the latter is decidable, so is the former[4]. Moreover, we show that we can effectively transform any proof of $M = N$ in the higher-order system, into a choice of some i and algebraic proofs for all the equations in L_i. Conversely, we show that we can effectively transform a given i and algebraic proofs for all the equations in L_i into a higher-order proof of $M = N$.

As a corollary, it follows that the combined lambda theory is a *conservative extension* of the algebraic theory. Such an "effective" proof of conservative extension was first mentioned in [Breazu-Tannen & Meyer 1987a] (a complete account is in [Breazu-Tannen 1987]) [5]. As another corollary, it follows that the combined lambda theory is a conservative extension of the pure simply typed lambda theory if and only if the algebraic theory does not contain any equations of the form $x = y$ with x and y distinct variables (*i.e.,* no "partial inconsistencies"; since the theory is many-sorted, this does not necessarily imply that every equation is provable) [6].

[3] This is not true in general: there are, of course, undecidable equational theories which are the combination of two decidable equational theories.

[4] The similar result about adding the untyped lambda calculus to algebraic theories is obviously false, since provable equality of pure untyped terms is already undecidable.

[5] There is an easier, model-theoretic but not effective, proof of this [Meyer & Reinhold 1986].

[6] Again, there is an easier, model-theoretic but not effective, proof of this [Breazu-Tannen 1986] which uses a completeness result from [Friedman 1975].

2 Combining rewrite systems

Our terminology and notation follows [Barendregt 1984]. In addition, we use $\overset{\beta}{\twoheadleftarrow\!\twoheadrightarrow}$ for β-conversion (several steps). Note that any simply typed term X has a β-normal form which decomposes as $\beta nf(X) \equiv \lambda\vec{v}.\, h\ Z_1 \cdots Z_n$ where h is a variable or a constant an each Z_i is a β-normal form.

Fix a many-sorted signature Σ and a set R of rewrite rules between $\alpha\Sigma$-terms [Huet & Oppen 1980], [Klop 1987]. Note that for each rule $s \longrightarrow t \in R$, s is not allowed to be a variable and $FV(s) \supseteq FV(t)$.

Lemma 2.1 *Let X and Y be $\lambda^{\rightarrow}\Sigma$-terms.*
(i) Let $r \in R$. If $X \overset{r}{\longrightarrow} Y$ then $\beta nf(X) \overset{r}{\twoheadrightarrow} \beta nf(Y)$.
(ii) If $X \overset{\beta R}{\twoheadrightarrow} Y$ then $\beta nf(X) \overset{R}{\twoheadrightarrow} \beta nf(Y)$.

Proof. (i) Let $r \equiv s \longrightarrow t$ and let $\{x_1, \ldots, x_n\} \equiv FV(s)$. Since $X \overset{r}{\longrightarrow} Y$, there exists a context $P[\ ]$ with exactly one hole of the same (base) type as s and t, and $\lambda^{\rightarrow}\Sigma$-terms $Q_1, \ldots, Q_n$ with Q_i of the same (base) type as x_i, such that $X \equiv P[s[\vec{x}:= \vec{Q}]]$ and $Y \equiv P[t[\vec{x}:= \vec{Q}]]$.

Let $X' \overset{\text{def}}{=} P[(\lambda\vec{x}.\, s)\, \vec{Q}]$ and $Y' \overset{\text{def}}{=} P[(\lambda\vec{x}.\, t)\, \vec{Q}]$. Also let z be a fresh variable of the same type as $\lambda\vec{x}.\, s$ and $\lambda\vec{x}.\, t$. Since both $\lambda\vec{x}.\, s$ and $\lambda\vec{x}.\, t$ are closed, we have $X' \equiv P[z\ \vec{Q}][z:= \lambda\vec{x}.\, s]$ and $Y' \equiv P[z\ \vec{Q}][z:= \lambda\vec{x}.\, t]$. Let $P' \overset{\text{def}}{=} \beta nf(P[z\ \vec{Q}])$. We claim that any occurrence of z in P' is at the head of a term of base type. Indeed, the property is true for $P[z\ \vec{Q}]$ and it is easy to show that it is preserved under β-reduction.

Finally, we show by induction on the length of Z that for any term Z in β-normal form, and in which every occurrence of z is at the head of a term of base type, we have $\beta nf(Z[z:= \lambda\vec{x}.\, s]) \overset{r}{\twoheadrightarrow} \beta nf(Z[z:= \lambda\vec{x}.\, t])$. This ends the proof of (i), since $X \overset{\beta}{\twoheadleftarrow\!\twoheadrightarrow} \beta nf(P'[z:= \lambda\vec{x}.\, s])$ and therefore $\beta nf(X) \equiv \beta nf(P'[z:= \lambda\vec{x}.\, s])$ and similarly for Y. (ii) is immediate from (i). **End of proof**

Lemma 2.2 *(i) If M is a β-normal form and $M \overset{R}{\twoheadrightarrow} N$ then N is a β-normal form.*
(ii) If R-reduction is CR on $\alpha\Sigma$-terms then it is also CR on $\lambda^{\rightarrow}\Sigma$-terms in β-normal form.

Proof. (i) Immediate.
(ii) We show, by induction on the length of M, that for any term M in β-normal form, R-confluence holds from M, that is, if $N \overset{R}{\twoheadleftarrow} M \overset{R}{\twoheadrightarrow} P$ then there exists a Q such that $N \overset{R}{\twoheadrightarrow} Q \overset{R}{\twoheadleftarrow} P$. This is trivial for length 1 since no rules apply. Assume the statement is true for all strictly shorter β-normal forms and let us show it holds for $M \equiv \lambda\vec{v}.\, h\ Z_1 \cdots Z_n$. There are several cases.

The case when h is a variable of base type is trivial: no rules apply. Next, consider the case when $h \equiv z$, a variable of non-base type. We then show that $M \overset{R}{\twoheadrightarrow} N$ iff $N \equiv \lambda\vec{v}.\, z\ W_1 \cdots W_n$ and $Z_i \overset{R}{\twoheadrightarrow} W_i$ for each i. From this, it immediately follows that R-confluence holds from M, using the induction hypothesis on the strictly shorter Z_i's. Moreover, the case when h is a constant from Σ

but $h\ Z_1 \cdots Z_n$ is not of base type is just like the previous case: indeed, h cannot occur in any R-redex.

The interesting case is when h is in Σ and $h\ Z_1 \cdots Z_n$ is of base type. Then, $h\ Z_1 \cdots Z_n$ can be decomposed as $t[x_1 := N_1, \ldots, x_k := N_k]$ where t is a $\alpha\Sigma$-term, the x_i's are fresh variable of base type, and the N_i's are β-normal forms of base type which have at head a variable of non-base type (this can be shown by induction on the lenghth of β-normal forms of base type). We then show that $M \overset{R}{\twoheadrightarrow} N$ iff $N \equiv \lambda\vec{v}.\, s[x_1 := P_1, \ldots, x_k := P_k]$, where $t \overset{R}{\twoheadrightarrow} s$ and $N_i \overset{R}{\twoheadrightarrow} P_i$ for each i. One direction is immediate, the other one follows by induction on the length of the R-reduction.

With this we can show that R-confluence holds from M, using the induction hypothesis on the strictly shorter N_i's and the fact that R-confluence holds from the algebraic t [7]. **End of proof.**

Finally, we have

Theorem 2.3
If R-reduction is CR on $\alpha\Sigma$-terms then βR-reduction is CR on $\lambda^{\rightarrow}\Sigma$-terms.

Proof. Suppose that $N \overset{\beta R}{\twoheadleftarrow} M \overset{\beta R}{\twoheadrightarrow} P$. By Lemma 2.1 we obtain $\beta nf(N) \overset{R}{\twoheadleftarrow} \beta nf(M) \overset{R}{\twoheadrightarrow} \beta nf(P)$. Then, by Lemma 2.2, there exists a Q such that $\beta nf(N) \overset{R}{\twoheadrightarrow} Q \overset{R}{\twoheadleftarrow} \beta nf(P)$. Thus $N \overset{\beta}{\twoheadrightarrow} \beta nf(N) \overset{R}{\twoheadrightarrow} Q \overset{R}{\twoheadleftarrow} \beta nf(P) \overset{\beta}{\twoheadleftarrow} P$. **End of proof.**

3 Combining equational theories

Fix a many-sorted signature Σ and a set E of equations between $\alpha\Sigma$-terms (no restrictions on the form of these equations).

The notion of equational proof we will work with is that of *chain of replacements of equals by equals.* Such proofs have the same power as deduction trees using axioms and inference rules such as congruence, transitivity, *etc.,* and there are effective translations between these two kinds of proofs, so we can proceed without loss of generality [Breazu-Tannen & Meyer 1987b], [Breazu-Tannen 1987]. Given $e \in E$ and two $\lambda^{\rightarrow}\Sigma$-terms X and Y, we define *one-step e-conversion,* $X \overset{e}{\longleftrightarrow} Y$ in the obvious type-preserving manner (free variables in e can be instantiated with any $\lambda^{\rightarrow}\Sigma$-term of the same type); this holds only when X and Y have the same type. This yields a notion of $\beta\eta E$-conversion. We will write $E \vdash^{\lambda^{\rightarrow}} X = Y$ and $X \overset{\beta\eta E}{\twoheadleftrightarrow} Y$ interchangeably. For s and t $\alpha\Sigma$-terms, we define proofs in the *algebraic theory*: $E \vdash^{\alpha} t = t'$ whenever there exist $\alpha\Sigma$-*terms* $t_0, \ldots, t_n$ $(n \geq 0)$ and $e_1, \ldots, e_n \in E$ such that $t \equiv t_0 \overset{e_1}{\longleftrightarrow} t_1 \overset{e_2}{\longleftrightarrow} \cdots \overset{e_{n-1}}{\longleftrightarrow} t_{n-1} \overset{e_n}{\longleftrightarrow} t_n \equiv t'$.

Since we are working with many-sorted/many-base-typed calculi, we should point out that this kind of equational reasoning is sound only in models with all sorts/types non-empty. As shown in

[7] This should have been used somewhere, and, indeed is needed here: while t is a β-normal form, it is not strictly shorter than M when M is actually algebraic.

[Breazu-Tannen & Meyer 1987b],[Breazu-Tannen 1987], proofs of conservative extension for reasoning about models with possibly empty sorts/types [Goguen & Meseguer 1982],[Lambek & Scott 1986] are actually simplified versions of the proofs for the reasoning described above [8]. The same is true for the result we give here. This paper presents only the version that applies to the more complicated flavor of equational reasoning.

A *long $\beta\eta$-normal form* is (recursively) a term of the form $\lambda\vec{v}.\, h\, Z_1 \cdots Z_n$ where h is a variable or a constant, each Z_i is a long $\beta\eta$-normal form, and $h\, Z_1 \cdots Z_n$ is of *base type*. While such a term is in general *not* in η-normal form, the name is justified by the fact that any $\lambda^{\rightarrow}\Sigma$-term, X, is $\beta\eta$-convertible to a *unique* long $\beta\eta$-normal form, $l\beta\eta nf(X)$; to effectively obtain it, take the term to β-normal form and then perform (if needed) some anti-η-reductions. With this, we have a result very similar to Lemma 2.1 (and the proof is essentially the same) which reduces everything to the analysis of E-conversion on long $\beta\eta$-normal forms:

Lemma 3.1 *Let X and Y be $\lambda^{\rightarrow}\Sigma$-terms.*
(i) Let $e \in R$. If $X \overset{e}{\longleftrightarrow} Y$ then $l\beta\eta nf(X) \overset{e}{\twoheadleftrightarrow} l\beta\eta nf(Y)$.
(ii) $X \overset{\beta\eta E}{\twoheadleftrightarrow} Y$ if and only if $l\beta\eta nf(X) \overset{E}{\twoheadleftrightarrow} l\beta\eta nf(Y)$.

We describe an effective procedure $\mathcal{B}$ which takes as input two terms M and N of the same type and in long $\beta\eta$-normal form and which returns a boolean expression $\mathcal{B}(M,N)$ built from atoms of the form $E \vdash^{\alpha} s = t$ or of the form **false**, by means of conjunctions and disjunctions. (The intention is that $M \overset{E}{\twoheadleftrightarrow} N$ iff $\mathcal{B}(M,N)$ is true). $\mathcal{B}$ is given as recursive procedure. The description of $\mathcal{B}$ will be interrupted with lemmas which are essential for its understanding.

Algorithm $\mathcal{B}$. Given M and N, since they are long $\beta\eta$-normal forms of the same type, they both decompose uniquely (and, of course, effectively) as $M \equiv \lambda\vec{v}.\, M'$ and $N \equiv \lambda\vec{v}.\, N'$ where M' and N' are long $\beta\eta$-normal forms of the same base type.

Lemma 3.2 *Any long $\beta\eta$-normal form of base type uniquely (and, of course, effectively) decomposes as $F[U_1,\ldots,U_m]$ where $F[\ ,\ldots,\ ]$ is an $\alpha\Sigma$-context (i.e., an $\alpha\Sigma$-term with holes of base type) and each U_i is a long $\beta\eta$-normal form of base type which has at head a variable of non-base type.*

The proof is by induction on the length of long $\beta\eta$-normal forms. Thus, $\mathcal{B}$ continues by decomposing $M' \equiv F[U_1,\ldots,U_m]$ and $N' \equiv G[V_1,\ldots,V_n]$. We need some *names* for all the holes in $F[\ ,\ldots,\ ]$ and $G[\ ,\ldots,\ ]$ and we choose to use numbers, as follows $F[1,\ldots,m]$, $G[m+1,\ldots,m+n]$. Let $\mathcal{H} \overset{\text{def}}{=} \{1,\ldots,m+n\}$ be the set of all these holes. Let $\mathcal{X}$ be a set of $m+n$ fresh variables, one for each hole in $\mathcal{H}$, and with same (base) type as it. We denote by $\Phi(\mathcal{H},\mathcal{X})$ the set of all type-preserving maps from $\mathcal{H}$ to $\mathcal{X}$ (there are at most $(m+n)^{m+n}$ such maps). We also rename the terms $U_1,\ldots,U_m,V_1,\ldots,V_n$ as follows: $Z_1 \overset{\text{def}}{=} U_1, \ldots, Z_m \overset{\text{def}}{=} U_m$, $Z_{m+1} \overset{\text{def}}{=} V_1, \ldots, Z_{m+n} \overset{\text{def}}{=} V_m$

[8] The simplification comes from the fact that in the reasoning of [Goguen & Meseguer 1982],[Lambek & Scott 1986] all the free variables that occur in a chain of replacements of equals by equals must also occur in one or the other of the two terms being proven equal.

and, for each $\phi \in \Phi(\mathcal{H},\mathcal{X})$, we define the $\alpha\Sigma$-terms $F[\phi] \stackrel{\text{def}}{=} F[\phi(1),\ldots,\phi(m)]$ and $G[\phi] \stackrel{\text{def}}{=} G[\phi(m+1),\ldots,\phi(m+n)]$. The following lemma is not necessary for the description of what $\mathcal{B}$ does, but, at this point, it may help understanding why $\mathcal{B}$ does it.

Lemma 3.3 *$F[Z_1,\ldots,Z_m] \overset{E}{\twoheadleftarrow\!\!\twoheadrightarrow} G[Z_{m+1},\ldots,Z_{m+n}]$ iff there exists a map $\phi \in \Phi(\mathcal{H},\mathcal{X})$ such that $E \vdash^{\alpha} F[\phi] = G[\phi]$ and, for all $i,j \in \{1,\ldots,m+n\}$ such that $\phi(i) = \phi(j)$, we have $Z_i \overset{prE}{\twoheadleftarrow\!\!\twoheadrightarrow} Z_j$.*

Here, "proper" E-conversion, *i.e.*, $\overset{prE}{\twoheadleftarrow\!\!\twoheadrightarrow}$, is defined only for long $\beta\eta$-normal forms of base type which have at head a variable of non-base type, as follows: $U \overset{prE}{\twoheadleftarrow\!\!\twoheadrightarrow} V$ whenever U and V have the *same* variable at head, thus $U \equiv z\ M_1 \ldots M_m$ and $V \equiv z\ N_1 \ldots N_m$, and, for all $i \in \{1,\ldots,m\}$ we have $M_i \overset{E}{\twoheadleftarrow\!\!\twoheadrightarrow} N_i$.

One direction in the proof of the lemma is immediate. The other one follows by induction on the length of the proof by E-conversion and we omit the details here.

Finally, we let

$$\mathcal{B}(M,N) := \bigvee_{\phi\in\Phi(\mathcal{H},\mathcal{X})} \left((\ E \vdash^{\alpha} F[\phi] = G[\phi]\ ) \wedge \bigwedge_{\{(i,j)|\phi(i)=\phi(j)\}} \mathcal{C}(Z_i,Z_j) \right) .$$

where $\mathcal{C}$ is a subroutine which takes as arguments two terms U and V of the same base type, in long $\beta\eta$-normal form, each of which has at head a variable of non-base type and returns boolean expressions of the same kind as the ones $\mathcal{B}$ returns. (The intention is that $U \overset{prE}{\twoheadleftarrow\!\!\twoheadrightarrow} V$ iff $\mathcal{C}(U,V)$).

Subroutine $\mathcal{C}$. Given U and V, since they are both long $\beta\eta$-normal forms of base type, they both decompose uniquely (and, of course, effectively) as $U \equiv u\ M_1\ \cdots\ M_m$ and $V \equiv v\ N_1\ \cdots\ N_n$ where the M_i's and N_j's are long $\beta\eta$-normal forms. If $u \not\equiv v$ let $\mathcal{C}(U,V) :=$ **false** . If $u \equiv v$, then $m = n$ and, for each i, M_i and N_i are long $\beta\eta$-normal forms of the same type so let

$$\mathcal{C}(U,V) := \mathcal{B}(M_1,N_1) \wedge \cdots \wedge \mathcal{B}(M_m,N_m) .$$

It is not hard to see that $\mathcal{B}$ terminates on all inputs. Indeed, if we replace all the calls of $\mathcal{C}$ in $\mathcal{B}$ by the corresponding code of $\mathcal{C}$ we obtain an equivalent presentation of $\mathcal{B}$, with simple recursive calls. Termination then follows from the observation that if the call $\mathcal{B}(M',N')$ occurs inside a call $\mathcal{B}(M,N)$ then $length(M') + length(N')$ is strictly less than $length(M) + length(N)$. Moreover, using Lemma 3.3, one can show by induction on the sum of the lenghts of M and N that

Lemma 3.4 *For any two long $\beta\eta$-normal forms, M and N, $M \overset{E}{\twoheadleftarrow\!\!\twoheadrightarrow} N$ iff $\mathcal{B}(M,N)$ is true.*

Note that the boolean expression $\mathcal{B}(M,N)$ can be simplified by using the distributivity of conjunction through disjunction and such facts as **false** $\wedge$ *exp* = **false**, **false** $\vee$ *exp* = *exp*, *etc.*, . In principle, we end up either with a disjunction of conjunctions of atoms of the form $E \vdash^{\alpha} s = t$, or, with just **false**. It is not hard to see that the second possibility never happens. (This corresponds to the fact that even two distinct, *pure* long $\beta\eta$-normal forms are E-convertible if the algebraic theory contains "partial inconsistency" equations of the form $x = y$, x and y distinct.) Thus, in fact, we have another algorithm $\mathcal{D}$, which calls $\mathcal{B}$, simplifies the resulting boolean expression and then produces a finite non-empty collection of finite non-empty sets (one for each conjunctive subexpression in the simplified boolean expression) of algebraic equations (where $s = t$ comes from $E \vdash^{\alpha} s = t$). Consider now the algorithm $\mathcal{A}$ which computes for any two $\lambda^{\rightarrow}\Sigma$-terms of the same type, X and Y,

$$\mathcal{A}(X,Y) := \mathcal{D}(l\beta\eta nf(X), l\beta\eta nf(Y)) .$$

Putting together all the results from above, we obtain:

Theorem 3.5
Let $\mathcal{A}(X,Y) = \{L_1,\ldots,L_n\}$. *Then,* $E \vdash^{\lambda^{\rightarrow}} X = Y$ *if an only if there exists an* $i \in \{1,\ldots,n\}$ *such that for all* $s = t \in L_i$ *we have* $E \vdash^{\alpha} s = t$.

In fact, Lemmas 3.3 and 3.4 can be easily strenghtened to yield effective transformations of proofs by higher-order E-conversion into algebraic proofs and conversely. Details are omitted here. This gives

Theorem 3.6
Let $\mathcal{A}(X,Y) = \{L_1,\ldots,L_n\}$. *Then, there is an effective procedure for transforming any proof of* $E \vdash^{\lambda^{\rightarrow}} X = Y$ *into an effective choice of an* $i \in \{1,\ldots,n\}$ *and proofs of* $E \vdash^{\alpha} s = t$ *for each* $s = t \in L_i$. *Conversely, there is an effective procedure which transforms any* $i \in \{1,\ldots,n\}$ *and proofs of* $E \vdash^{\alpha} s = t$ *for each* $s = t \in L_i$, *into a proof of* $E \vdash^{\lambda^{\rightarrow}} X = Y$.

For algebraic terms, $\mathcal{A}(s,t) = \{\{s = t\}\}$, which gives

Corollary 3.7 *("effective" conservative extension of algebraic theories by simple types)*
There is an effective procedure which transforms any proof of $E \vdash^{\lambda^{\rightarrow}} s = t$ *where* s *and* t *are* $\alpha\Sigma$*-terms, into a proof of* $E \vdash^{\alpha} s = t$.

Let M and N be *pure* $\lambda^{\rightarrow}$-terms and let $\mathcal{A}(M,N) = \{L_1,\ldots,L_n\}$. It is not hard to see that each L_i consists only of equations of the form $x = x$ or $x = y$ with x and y distinct. A finer analysis shows:

Corollary 3.8 *(conservative extension of the pure simply typed lambda calculus by consistent algebraic theories) The following statements are equivalent*
1) $\forall M,N$, *pure* $\lambda^{\rightarrow}$*-terms of the same type,* $E \vdash^{\lambda^{\rightarrow}} M = N \iff M \overset{\beta\eta}{\twoheadleftarrow\twoheadrightarrow} N$.
2) $\forall x,y$, *algebraic variables of the same sort,* $E \vdash^{\alpha} x = y \implies x \equiv y$.

4 Conclusions and directions for future investigation

In addition to serving as another piece of formal evidence for the intuition that the interaction between algebra and higher-order types has no new negative effects, we believe that our proof-theoretic reductions will be useful in future development of equational theorem provers, an area of active research (see for example [Siekmann (ed.) 1986]).

We have made no attempts to analyze the complexity of the algorithm presented in section 3. The algorithm proceeds in two phases, the first of which involves normalization, and is therefore non-elementary recursive [Statman 1979]. This does sound scary in theory, but experience with theorem-provers has shown that most normalizations that occur in practice can be done efficiently [Miller & Nadathur 1986]. With this in mind, the complexity of the second part of the algorithm becomes relevant, and we propose to look for efficient versions of it.

Another complexity-theoretic question regards the length of higher-order equational proofs vs. that of algebraic proofs. Intuitively, the latter should be shorter than the former. However, the proof transformations we establish give, at first glance, conflicting information. It would be nice to formally settle this problem.

Turning to the result about rewrite systems, an obvious question is what happens to the weak and strong normalization properties. Assuming that every $\alpha\Sigma$-term R-reduces to an R-normal form, it is not hard to show, in the spirit of the proofs in section 2, that any $\lambda^{\rightarrow}\Sigma$-term βR-reduces to an βR-normal form. However, the pure simply typed lambda calculus is not just weakly, but *strongly* normalizing. It is an open question whether strong normalization of the algebraic rewrite system implies strong normalization of the combined rewrite system. A plethora of counterexamples for similar questions about the direct sum of algebraic rewriting systems [Toyama 1986] suggests a cautious approach to any kind of conjecture here. We should point out, though, one fact which is specific to the interaction between algebraic rewriting and β-reduction: an algebraic reduction step cannot create "essentially new" β-redexes, at worst it will duplicate existing β-redexes.

In this first series of results we chose the simple type discipline for convenience reasons: simplicity of notation, no need for type assignments, *etc.*, . In fact we believe that our results and proofs generalize readily to the Girard-Reynolds polymorphic type discipline, as already shown for the conservative extension result (see [Breazu-Tannen & Meyer 1987a] and [Breazu-Tannen & Meyer 1987b] and, for detailed proofs, [Breazu-Tannen 1987]).

However, it seems to us that the most interesting case is that of *full dependent type disciplines* such as the impredicative type discipline of the Calculus of Constructions [Coquand & Huet 1985a], [Coquand & Huet 1985b], that of Martin-Löf's Type:Type calculus [Martin-Löf 1971] (or see [Meyer & Reinhold 1986]) or the predicative type discipline presented in [MacQueen 1986]. Indeed, in such a type discipline, terms can occur inside types, which makes even the problem of *type-checking* which is trivially decidable for the simple and the polymorphic type disciplines) at least as hard as the problem of provable equality of terms. For those dependent type disciplines which are normalizing (such as that of the Calculus of Constructions and that of [MacQueen 1986]) we expect to show, using proofs in the spirit of the present paper, that when combined with an arbitrary algebraic theory, the problem of provable equality, and, more importantly, that of type-checking in the higher-order theory, are both Turing-reducible to the problem of provable equality

in the algebraic theory, that is, that the interaction between algebra and even such complex higher-order type disciplines does not have new negative effects. This, of course, is false for Type:Type where even the problem of type-checking of *pure* terms is undecidable [Meyer & Reinhold 1986], [Howe 1987], [Coquand 1986].

As a side-product of these conjectured proof-theoretic reductions we expect to obtain useful heuristics for type-checking in such rich type disciplines. This expectation is based on the belief that while arbitrarily complex terms can occur inside dependent types, practical program type-checking will, in fact, not involve computations of universal power inside the types (for another line of research with similar goals see [Cardelli 1987]).

Acknowledgement. I am grateful to Jean Gallier for several suggestions on how to improve the presentation of the paper.

References

[Abelson & Sussman 1985] H. Abelson and G. J. Sussman. *Structure and interpretation of computer programs.* The MIT Press, 1985.

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

[Bercovici 1984] I. Bercovici. Strong normalization for typed lambda calculus with surjective pairing—Tait's method. Unpublished manuscript, Laboratory for Computer Science, MIT. July 1984.

[Breazu-Tannen & Meyer 1987a] V. Breazu-Tannen and A. R. Meyer. Computable values can be classical. In *Proceedings of the 14th Symposium on Principles of Programming Languages*, pages 238–245, ACM, January 1987.

[Breazu-Tannen & Meyer 1987b] V. Breazu-Tannen and A. R. Meyer. Polymorphism is conservative over simple types. In *Proceedings of the Symposium on Logic in Computer Science*, pages 7–17, IEEE, June 1987.

[Breazu-Tannen 1986] V. Breazu-Tannen. Some conservative extension situations between logics of program equivalence. Talk given at Bell Labs, IBM Yorktown Heights, and Brown University. February 1986.

[Breazu-Tannen 1987] V. Breazu-Tannen. *Conservative extensions of type theories.* Ph.D. thesis, Massachusetts Institute of Technology, February 1987. Supervised by A. R. Meyer.

[Cardelli 1987] L. Cardelli. Phase distinctions in type theory. Manuscript. March 1987.

[Coquand & Huet 1985a] T. Coquand and G. Huet. *A Calculus of Constructions.* Technical Report, INRIA, Rocquencourt, France, June 1985.

[Coquand & Huet 1985b] T. Coquand and G. Huet. *Constructions: A Higher Order Proof System for Mechanizing Mathematics.* Rapport de Recherche 401, INRIA, Domaine de Voluceau, 78150 Rocquencourt, France, May 1985. Presented at EUROCAL 85, Linz, Austria.

[Coquand 1986] T. Coquand. Communication in the TYPES electronic forum (types@theory.lcs.mit.edu). September 17, 1986.

[Dougherty 1986] D. Dougherty. Personal communication. September 1986.

[Friedman 1975] H. Friedman. Equality between functionals. In R. Parikh, editor, *Proceedings of the Logic Colloqium '73*, pages 22–37, *Lecture Notes in Mathematics*, Vol. 453, Springer-Verlag, 1975.

[Goguen & Meseguer 1982] J. Goguen and J. Meseguer. Completeness of many-sorted equational logic. *SIGPLAN Notes*, 17:9–17, 1982.

[Gordon *et al.,* 1979] M. J. Gordon, R. Milner, and C. P. Wadsworth. *Edinburgh LCF.* Volume 78 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1979.

[Howe 1987] D. J. Howe. The computational behaviour of Girard's Paradox. In *Proceedings of the Symposium on Logic in Computer Science*, pages 205–214, IEEE, June 1987.

[Hudak 1986] P. Hudak. Para-functional programming. *Computer*, 18:60–70, August 1986.

[Huet & Oppen 1980] G. Huet and D. Oppen. Equations and rewrite rules: a survey. In Ronald V. Book, editor, *Formal Language Theory: Perspectives and Open Problems*, pages 349–405, Academic Press, New York, 1980.

[Huet 1978] G. Huet. Confluent reductions: abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27:797–821, 1978.

[Klop 1980] J. W. Klop. *Combinatory reduction systems.* Tract 129, Mathematical Center, Amsterdam, 1980.

[Klop 1987] J. W. Klop. Term rewriting systems: a tutorial. *Bull. EATCS*, (32):143–182, June 1987.

[Lambek & Scott 1986] J. Lambek and P. J. Scott. *Introduction to higher-order categorical logic.* Volume 7 of *Cambridge studies in advanced mathematics*, Cambridge University Press, 1986.

[MacQueen 1986] D. B. MacQueen. Using dependent types to express modular structure. In *Conf. Record Thirteenth Ann. Symp. Principles of Programming Languages*, pages 277–286, ACM, January 1986.

[Martin-Löf 1971] Per Martin-Löf. *A Theory of Types.* Report 71-3, Department of Mathematics, University of Stockholm, February 1971.

[Meyer & Reinhold 1986] A. R. Meyer and M. B. Reinhold. 'Type' is not a type: preliminary report. In *Conf. Record Thirteenth Ann. Symp. Principles of Programming Languages*, pages 287–295, ACM, January 1986.

[Miller & Nadathur 1986] D. Miller and G. Nadathur. Higher-order logic programming. In *Proceedings of the Third International Logic Programming Conference, London*, pages 448–462, June 1986.

[Newman 1942] M. H. A. Newman. On theories with a combinatorial definition of "equivalence". *Ann. Math*, 43:223–243, 1942.

[Pottinger 1981] G. Pottinger. The Church-Rosser theorem for the typed λ-calculus with surjective pairing. *Notre Dame J. of Formal Logic*, 22:264–268, 1981.

[Siekmann (ed.) 1986] J. H. Siekmann (ed.). *Proceedings of the 8th International Conference on Automated Deduction.* Volume 230 of *Lecture Notes in Computer Science*, Springer-Verlag, 1986.

[Statman 1979] R. Statman. The typed λ-calculus is not elementary recursive. *Theoretical Computer Science*, 9:73–81, 1979.

[Toyama 1986] Y. Toyama. Counterexamples to terminating for the direct sum of term rewriting systems. Manuscript. 1986.

[Toyama 1987] Y. Toyama. On the Church-Rosser property for the direct sum of term rewriting systems. *Journal of the ACM*, 34(1):128–143, January 1987.

[Turner 1985] D. A. Turner. Miranda: a non-strict functional language with polymorphic types. In J.-P. Jouannaud, editor, *Proceedings of the Conference on Functional Programming Languages and Computer Architecture*, pages 1–16, *Springer Lecture Notes in Computer Science*, Vol. 201, Springer-Verlag, 1985.