A Combinatory Logic Approach to Higher-order E-unification [*]

Daniel J. Dougherty
Department of Mathematics
Wesleyan University
Middletown, CT 06457 USA
ddougherty@eagle.wesleyan.edu

Patricia Johann
Department of Mathematics and Computer Science
Hobart and William Smith Colleges
Geneva, NY 14456 USA
johann@hws.bitnet

Abstract

Let E be a first-order equational theory. A translation of typed higher-order E-unification problems into a typed combinatory logic framework is presented and justified. The case in which E admits presentation as a convergent term rewriting system is treated in detail: in this situation, a modification of ordinary narrowing is shown to be a complete method for enumerating higher-order E-unifiers. In fact, we treat a more general problem, in which the types of terms contain type variables.

1 Introduction

Investigation of the interaction between first-order and higher-order equational reasoning has emerged as an active line of research. The collective import of a recent series of papers, originating with [Bre88] and including (among others) [Bar90], [BG91a], [BG91b], [Dou92], [JO91] and [Oka89], is that when various typed λ-calculi are enriched by first-order equational theories, the validity problem is well-behaved, and furthermore that the respective computational approaches to verifying equations (β-reduction and term rewriting) interact in a "modular" fashion.

This paper is concerned with the satisfiability problem in such a combined system, that is, with higher-order E-unification. The main novelty in our approach lies in the use of typed combinatory logic ($\mathcal{CL}$) rather than typed λ-calculus ($\mathcal{LC}$) as a formalization of higher-order logic. The claim herein is that, as an algebraic treatment of higher-order

[*]This paper is part of the second author's dissertation for the Ph.D. at Wesleyan University in 1991, under the direction of Daniel J. Dougherty; it was presented at the Eleventh International Conference on Automated Deduction in Saratoga Springs, NY, June 1992 and an extended abstract appears in the proceedings of that conference.

reasoning, $\mathcal{CL}$ provides a congenial setting for incorporating first-order unification into the higher-order problem, eliminating the complexities (both conceptual and practical) involved with the presence of bound variables.

We provide evidence for this claim by analyzing the situation in which the equational theory E admits a presentation as a convergent (confluent and terminating) term rewriting system. We develop the completeness of an algorithm which is essentially a normalized narrowing algorithm described in terms of transformations on systems, and which enumerates a complete set of higher-order E-unifiers for any system of terms. Future work will explore a combinatory approach to preunification and will address specific equational theories such as associativity and commutativity.

Any of the standard effective translations between $\mathcal{LC}$ and $\mathcal{CL}$ facilitate a natural approach to solving higher-order unification problems. A solution strategy might attempt to unify in $\mathcal{CL}$ the translations of the $\mathcal{LC}$-terms specifying a higher-order unification problem, rather than performing the unifications in $\mathcal{LC}$. Since the basic axioms for $\mathcal{CL}$ admit presentation as a convergent reduction relation, and so support narrowing as a unification procedure, this program is particularly appealing. Encouraged by the well-known techniques for combining unification algorithms for *first-order* theories ([Sch89], [Bou90]), we can hope for a modular solution to the problem of combining the first-order and higher-order problems.

But the reader familiar with combinatory logic will immediately recognize a difficulty: even when E is empty, the equality generated by the basic $\mathcal{CL}$-equations is *not* that induced by the translation from $\mathcal{LC}$ to $\mathcal{CL}$. Furthermore, no convergent term rewriting system is known for this induced equality.

This difficulty is solved here — generalizing the techniques of [Dou93] — by defining a certain notion of reduction on *systems* of $\mathcal{CL}$-terms. When E has a convergent presentation, this reduction captures the induced equality described above and supports the syntactic unification strategy of narrowing (and so yields a solution to the unification problem in its usual formalization). The situation turns out to be particularly pleasant when the term rewriting system underlying E is left-linear.

Classical higher-order unification concerns unification of terms of the explicitly simply typed λ-calculus; here we treat a more general problem in which the types of terms contain type variables which are eligible for instantiation by our answer substitutions.

Compiling functional programs into combinators has become a standard technique, motivated by the inefficiencies inherent in instantiating terms in the presence of bound variables (see [Pey87] for a discussion); implementations of higher-order unification problems should enjoy similar benefit from the passage to combinators.

Moreover, reasoning about and implementing substitution is simpler in an algebraic setting than in the traditional framework, and the use of combinators spares us the usual manipulation of long η-normal forms of $\mathcal{LC}$-terms. There is more than notational simplicity at stake: as pointed out by Snyder ([Sny90]), the presence of bound variables during paramodulation steps causes significant technical complications.

The incorporation of type-variables represents a significant generalization of the classical problem and yet presents no difficulties for the approach described here, since our algorithm is driven by the shapes of the terms (as usual in algebraic unification) rather than by the shape of the types. Huet's classical algorithm, by contrast, requires that the types of terms are completely specified (see the discussions in [Nip90] and especially [Dou92]).

Finally, the use of type-variables allows a finite axiomatization of typed combinatory logic, which in turn supports a complete unification procedure with a finitely branching search space.

Applications of pure higher-order unification in automated deduction are varied and plentiful: higher-order logic, specification of higher-order logics, machine learning, type inference in polymorphic lambda calculus, and extensions of logic programming are but some areas employing higher-order unification methods. Unification in the presence of a first-order equational theory E has been surveyed in [Sie89] and [JK91]; the use of narrowing as an algebraic unification procedure originates with Fay ([Fay79]).

The seminal work in higher-order unification is [Hue75]; Gallier and Snyder ([GS89b]) have presented Huet's algorithm in a transformational setting. Snyder ([Sny90]) has given transformations for higher-order unification in the presence of an *arbitrary* equational theory E; Nipkow and Qian ([NQ91]) refined these to allow a modular approach (to pre-unification as well as unification) when a unification algorithm for E is known. The methods in [NQ91] are complete when E satisfies certain (strong) constraints.

Unification in the presence of type variables has applications in program transformation ([PE88], [Ell90]) and higher-order logic programming ([EP89]), and would be useful in theorem proving in higher-order logic (see [NPS??] for a more complete discussion). Other attempts to extend classical higher-order unification to allow more flexible typing schemes include treatment of λ-calculi with type-variables in [Nip90] and [Hus??], and the presentation of an algorithm for unification in the presence of dependent function types in [Ell89]. Nipkow's algorithm has been incorporated into the generic theorem prover Isabelle, and Elliott's is designed as the basis for a generalization of the programming language λ-Prolog. All of these algorithms are based on Huet's method; only Hustadt's is a complete unification procedure.

Preliminaries

We will often draw upon classical results about the lambda calculus and combinatory logic (see, for example [HS86]) and use the basic results on the combination of lambda calculus and first-order rewriting ([Bre88], [BG91a], [BG91b], [Dou92]). We will assume familiarity with the use of transformations on systems to study unification ([MM82], [GS89a]). For definitions and notations here not given explicitly, the reader is referred to [DJ90].

Fix a set of equations E.

Terms and equalities

The *types* are formed by closing a set of *base types* and *type-variables* under the operation $(\alpha_1 \to \alpha_2)$ for types α_1 and α_2. Base types and type-variables are called *atomic types.*

It will be convenient to arrange that distinct term-variables do not become identical by virtue of a type-substitution; we therefore require a precise notion of type-erasure for term-variables. Fix an infinite well-ordered set of *indeterminates* and an infinite well-ordered set of *parameters.* A *term-variable* is an ordered pair consisting of an indeterminate and a type; a *constant* is an ordered pair consisting of a parameter and a type; an *atom* is either a term-variable or a constant. The *type-erasure* of an atom is the first element of the pair. Note that distinct atoms are not required to have distinct type-erasures, although we will assume, for convenience, that distinct atoms in E do.

We will denote by Σ the set of all constants whose types are of the form $\alpha_1 \to \alpha_2 \to \ldots \to \alpha_k$, where $k > 0$ and α_i is a base type for $i = 1, \ldots, k$. When discussing combinatory logic we assume that the parameters include the symbols I, K, and S (although it is possible to postulate only K and S, for technical reasons we will need to have I as a primitive parameter). In the course of testing equality or unifiability of terms we will find it convenient to introduce constants not occurring in any terms under consideration; we therefore assume that the set of parameters has a distinguished infinite set *Args* disjoint from $\{I, K, S\}$ and the type erasures of all symbols in E.

$\mathcal{LC}$ is the set of explicitly simply typed lambda terms over the atoms excluding I, K, and S; $\mathcal{CL}$ is the set of explicitly simply typed combinatory logic terms over these atoms together with the various I, K, and S typed as usual (the typed I, K, and S are called *redex* atoms). The *support* of a term T, $Supp(T)$, is the set of type-variables occurring in T together with the indeterminates occurring among the type-erasures of the atoms in T; a *pure* term is a term in which no constant occurs whose erasure is in *Args*. A *fresh* indeterminate or parameter is one not occurring in any term currently under consideration; we will often refer to a choice of a term T with fresh variables, by which we mean that $Supp(T)$ is disjoint from all type-variables and indeterminates currently under consideration.

Syntactic equality between terms or types is denoted by $\equiv$. We will not explicitly indicate the types of terms unless it is necessary.

If $T \equiv hT_1 \ldots T_n$ is a term and h is an atom, then h is called the *head* of T; if furthermore $h \in \Sigma$, then T is said to be a Σ-term. An *algebraic term* is either a variable of base type or a term of the form $hT_1 \ldots T_n$, where $h \in \Sigma$ and each term T_i is algebraic.

Each term T can be identified with a labeled binary tree, and when this is done, the set of 0-1 sequences indexing its symbols is called the set of *occurrences* of T and denoted $O(T)$. As usual, we write T/u for the subterm of T determined by the occurrence u, and $T[u \leftarrow U]$ for the term obtained by replacing T/u by U. Write $Vars(T)$ for the set of variables which are T/u for some $u \in O(T)$.

On $\mathcal{CL}$, *weak equality* is generated by *weak reduction*, determined by the rules $Ix \longrightarrow x$, $Kxy \longrightarrow x$, and $Sxyz \longrightarrow xz(yz)$; given a term rewriting system R, *wR-reduction* and *$\beta\eta R$-reduction* are the reduction relations generated by R together with the rules for weak reduction or $\beta\eta$-reduction, respectively. Weak reduction and $\beta\eta$-reduction are both convergent on typed terms, and so we may speak of the *weak normal form* of a $\mathcal{CL}$-term and the *$\beta\eta$-normal form* of a $\mathcal{LC}$-term. The *long β-normal form* of a $\mathcal{LC}$-term is obtained in the usual way. If R is convergent, then so are wR and $\beta\eta^{-1}R$ ([Bre88], [BG91a], [BG91b]); we therefore similarly refer to the *wR-normal form* or *long βR-normal form* of a $\mathcal{CL}$- or $\mathcal{LC}$-term as appropriate.

Fix any of the standard $\Lambda : \mathcal{CL} \to \mathcal{LC}$ and $\mathcal{H} : \mathcal{LC} \to \mathcal{CL}$ such that $\Lambda(\mathcal{H}(M)) =_{\beta\eta} M$. Given a first-order equational theory E, define *extensional combinatory E-equality* by $X =_{CE} Y$ iff $\Lambda(X) =_{\beta\eta E} \Lambda(Y)$; it follows that for any $\mathcal{LC}$-terms M and N, $M =_{\beta\eta E} N$ iff $\mathcal{H}(M) =_{CE} \mathcal{H}(N)$.

Substitutions and unification

A *type-substitution* is an ordinary algebraic substitution over the algebra of types; a type-substitution θ_0 induces a mapping on terms in an obvious way, and we shall denote this map by θ_0 as well. A *term-substitution* θ_1 is an ordinary (type-preserving) substitution on $\mathcal{LC}$- or $\mathcal{CL}$-terms, as appropriate. A *substitution* θ is a pair consisting of a type-substitution θ_0 and a term-substitution θ_1; such a pair induces a mapping on $\mathcal{LC}$ and on $\mathcal{CL}$, also denoted θ, by the rule $\theta T \equiv \theta_1(\theta_0 T)$ (application of a substitution to a term, as well as composition of substitutions, will be indicated by juxtaposition). A *pure* substitution is one whose range is a set of pure terms. It will be notationally convenient to allow a term-substitution θ_1 to act as the identity on types, so that we may refer to $\theta\alpha$ when α is a type.

Our dual substitutions behave in most ways just as ordinary substitutions. A substitution θ is idempotent iff both θ_0 and θ_1 are idempotent, for example, and if two ($\mathcal{LC}$- or $\mathcal{CL}$-) terms are syntactically unifiable they possess a most general unifier, or *mgu*. Details are worked out in [Dou93].

An instance of the *higher-order E-unification* problem is a two-element multiset $\langle M, N\rangle$ of $\mathcal{LC}$-terms; a solution is a substitution θ such that $\theta M =_{\beta\eta E} \theta N$.

Suppose $\mathcal{W}$ is a set of type-variables and indeterminates. Whenever $=_*$ is a notion of equality on terms the notation $\theta =_* \theta'[\mathcal{W}]$ means that

1. for every type-variable $t \in \mathcal{W}$, $\theta_0(t) \equiv \theta'_0(t)$, and
2. for every term-variable x whose type-erasure is in $\mathcal{W}$, $\theta_1(x) =_* \theta'_1(x)$.

A notion of equality $=_*$ on terms induces an order $\leq_*$ on substitutions defined by: $\theta \leq_* \theta'[\mathcal{W}]$ if there is a substitution μ with $\mu\theta =_* \theta'[\mathcal{W}]$.

The transfer to combinatory logic

If θ is an $\mathcal{LC}$-substitution, let the $\mathcal{CL}$-substitution $(\mathcal{H} \circ \theta)$ be defined by $(\mathcal{H} \circ \theta)X \equiv (\mathcal{H} \circ \theta_1)(\theta_0 X)$; if θ is a $\mathcal{CL}$-substitution, define $(\Lambda \circ \theta)$ analogously. The justification for our strategy of translating the unification problem from $\mathcal{LC}$ to $\mathcal{CL}$ is embodied in the following lemma, adapted from [Dou93], which follows from the facts that for any $\mathcal{LC}$-term M and substitution θ, $\mathcal{H}(\theta M) \equiv (\mathcal{H} \circ \theta)\mathcal{H}(M)$, and for any $\mathcal{CL}$-term X and substitution θ, $\Lambda(\theta X) \equiv (\Lambda \circ \theta)\Lambda(X)$.

Lemma 1.1 *Let M and N be $\mathcal{LC}$-terms. The $\beta\eta E$-unifiers of M and N are (up to pointwise $\beta\eta$-conversion) those of the form $(\Lambda \circ \theta)$, where θ ranges over the $\mathcal{CL}$-substitutions such that $\theta\mathcal{H}(M) =_{CE} \theta\mathcal{H}(N)$.*

If we define *extensional combinatory E-unification* as the problem of unifying $\mathcal{CL}$-terms with respect to extensional combinatory E-equality, the above discussion shows how a method for extensional combinatory E-unification yields a method for higher-order E-unification as originally presented.

The rest of this paper will be concerned with extensional combinatory E-equality, henceforth *CE-equality*, and extensional combinatory E-unification, henceforth *CE-unification*. The unqualified word "term" will mean "combinatory logic term".

When E is presented by a term rewriting system R we will often abuse notation by writing, e.g., $X =_{CR} Y$ for $X =_{CE} Y$, or by referring to "CR-validity" instead of "CE-validity." When E is empty we will refer to "C-validity" and "C-equality," and write $X =_C Y$.

Systems

A *pair* is either a *term-pair* or a *type-pair*, where a term-pair is a two-element multiset of $\mathcal{CL}$-terms and a type-pair is a two-element multiset of types. A pair is *trivial* if its elements are identical, and CE*-valid* if its elements are CE-equal (it will be convenient to consider trivial type-pairs to be CE-valid).

A *system* is a set of pairs in which no two distinct atoms have the same type-erasure; this of course implies that no two distinct terms appearing as subterms of terms in a system have the same type-erasure. A system is *trivial* if each of its pairs is trivial, and CE*-valid* if each of its pairs is CE-valid. If the symmetric difference between the systems Γ and Γ' (i.e., the set of pairs occurring in exactly one of Γ and Γ') is trivial, we write $\Gamma \cong \Gamma'$. As is customary, we use $\Gamma, \langle X, Y \rangle$ to abbreviate $\Gamma \cup \{ \langle X, Y \rangle \}$. Since this is ambiguous as a decomposition of the system in question (Γ may or may not contain $\langle X, Y \rangle$), we introduce the notation $\Gamma; \langle X, Y \rangle$ to refer to $\Gamma \cup \{ \langle X, Y \rangle \}$ with the understanding that $\langle X, Y \rangle$ is *not* a pair in Γ.

A consequence of the fact that terms are explicitly typed (rather than typable under a type-inference system) is that a pair will not be CE-valid unless its terms have the same type; this restriction is not built into the definition of system since terms with different types may still be unifiable. Type-pairs will play no role until Section 4.

A substitution θ is a *unifier* of a system Γ if $\theta\Gamma$ (obtained by applying θ to each type and term occurring in Γ) is trivial. A substitution θ is a CE*-unifier* of a system Γ if $\theta\Gamma$ is CE-valid. When E is empty or admits presentation as a term rewriting system, we refer to "C-unifiers" or "CR-unifiers," respectively.

The restriction on type-erasures of the atoms in a system is designed to avoid the technical complications which would result if distinct variables could become identical after a type-substitution, and to avoid unneccessary identification of other terms in systems via substitution application as well.

Note that if Γ is a unification problem presented as a set of pairs in which some indeterminates appear with more than one type, then successive applications of syntactic unification on the types assigned to these indeterminates will result in a system with the same CE-unifiers.

Let Γ be a system. If $\langle t, \alpha \rangle$ is a type-pair in Γ and there are no occurrences (in type- or term-pairs) of t in Γ other than the one indicated, then t is *solved* in Γ and $\langle t, \alpha \rangle$ is a *solved type-pair.* If $\langle x, X \rangle$ is a term-pair in Γ, x and X have the same type, and there are no occurrences of x in Γ other than the one indicated, then x is *solved* in Γ and $\langle x, X \rangle$ is a *solved term-pair.*

If each non-trivial term- or type-pair in Γ is solved, then Γ is a *solved system*, and its pairs determine an idempotent substitution in the usual way, although a pair consisting of two solved variables requires a choice as to which is in the domain of the substitution. Conversely, any idempotent substitution can be represented as a solved system; if θ is an idempotent substitution, write $[\theta]$ for any solved system which represents it.

2 The Validity Problem

In this section, we give the transformations for CR-validity which will comprise the main analytical tool in investigating the completeness properties of our unification algorithm.

2.1 Transformations for C-validity

It will be helpful to have a brief discussion of the combinator-based approach to simple higher-order unification (without first-order equations).

Certainly C-equality is decidable; we may simply pass to $\mathcal{LC}$ and use (convergent) $\beta\eta$-reduction to test for equality of terms there. We might hope for a corresponding rewrite relation defined directly over $\mathcal{CL}$-terms, since narrowing is a well-understood unification technique for equational theories admitting a convergent presentation. Unfortunately no such relation is known (the classical *strong reduction* relation in $\mathcal{CL}$, which does capture C-equality, is clearly not suitable as foundation for unification — it is not finitely axiomatizable, and indeed even recognizing the set of rules is non-trivial).

However, there is defined in [Dou93] a well-behaved notion of reduction on *systems* which decides C-equality. C-equality can be obtained from weak equality by the addition of the extensionality rule:

$$\text{Infer } X =_C Y \text{ from } Xz =_C Yz, \text{ when } z \text{ is not free in } X \text{ or in } Y.$$

Using this rule, deciding C-equality reduces to deciding C-equality between atomic-type terms. The weak normal form of any atomic-type term has a non-redex atom at the head. But two terms $hX_1 \cdots X_k$ and $h'Y_1 \cdots Y_{k'}$ are C-equal iff $h \equiv h'$, $k \equiv k'$, and $X_i =_C Y_i$ for each i. This motivates the following set of transformations.

Definition 2.1 The set VT consists of the following three transformations:

- WEAK REDUCE
$$\Gamma;\ \langle X, Y\rangle \longrightarrow \Gamma,\ \langle X', Y\rangle\,,$$
when X weakly reduces to X'.

- ADD ARGUMENT
$$\Gamma;\ \langle X, Y\rangle \longrightarrow \Gamma,\ \langle Xd, Yd\rangle\,,$$
when X and Y have the same type, at least one of X and Y is of one of the forms: I, K, KA, S, SA, or SAB, and d is built from the first parameter in *Args* not occurring in $\langle X, Y\rangle$, and given the appropriate type.

- DECOMPOSE
$$\Gamma;\ \langle hX_1 \cdots X_k, hY_1 \cdots Y_k\rangle \longrightarrow \Gamma,\ \langle X_1, Y_1\rangle, \ldots, \langle X_k, Y_k\rangle\,,$$
when h is a non-redex atom and $k \geq 1$.

Here and elsewhere, we will observe the convention that no transformation is to be applied to a trivial pair. Observe the use of ";" on the left-hand sides of transformations, so that the effect of the transformation is unambiguous, and the use of "," on the right-hand sides, to preclude repetition of identical pairs.

The notation for WEAK REDUCE exploits the fact that pairs are unordered; we intend of course that either element of a pair may be reduced. A similar remark applies in several places below. The use, in ADD ARGUMENT, of new constants rather than new variables will serve to remind us in unification that the new arguments are not part of the original term and should not be instantiated. The necessity for the restriction on d in ADD ARGUMENT may be seen by considering the non-C-valid pair $\langle Kd, I\rangle$, which can be reduced by an improper application of ADD ARGUMENT to the C-valid pair $\langle Kdd, Id\rangle$.

It is tempting to think of VT as a term rewriting system, but the analogy is not perfect, since ADD ARGUMENT can only be applied at the heads of terms (it changes their types) and a system admitting a DECOMPOSE step need no longer do so after instantiation (consider the case when a redex atom is substituted for a head variable). Fortunately, the facts that term rewriting systems are closed under context application and stable under substitution are not necessary for supporting a unification procedure, as will be demonstrated in the remainder of this paper.

The key facts about VT (proved in [Dou93]) are the following:

Theorem 2.2 *Let Γ be any system.*

1. *(Soundness) Suppose $\Gamma \longrightarrow \Gamma'$. Then Γ is C-valid if Γ' is C-valid.*
2. *(Sufficiency) Suppose Γ is* VT*-irreducible. Then Γ is C-valid iff it is trivial.*
3. *(Termination) Every sequence of* VT*-steps terminates.*

Here, in the absence of algebraic equations, part (1) of Theorem 2.2 can be strengthened to read "if and only if". Thus, a simple algorithm for deciding C-equality between terms X and Y can apply VT transformations in any order to the system originally comprising $\langle X, Y\rangle$. Since every sequence of VT-steps terminates, an irreducible system will be obtained; $X =_C Y$ iff this system is trivial. The VT transformations can in fact be restricted to head applications; of course, we may halt and report non-equality if ever we generate a pair of terms whose heads are different non-redex atoms.

The combinator-based higher-order unification method can be characterized simply as the notion of narrowing on systems induced by VT as a reduction relation; Theorem 2.2 is the foundation of the completeness proof.

2.2 Transformations for CR-validity

In the move to higher-order E-unification, the difficult step is that of constructing a set of transformations analogous to VT which capture CE-validity. The rest of this section is devoted to such a construction.

We must first understand how first-order equality, especially when presented in terms of a term rewriting system R, interacts with C-equality. Known results concerning rewriting and the λ-calculus are encouraging, but do not apply directly. If R is terminating, termination of wR-reduction follows from termination of βR-reduction, but of course wR-reduction will not capture CR-equality. Furthermore, we are committed to unification relative to *extensional* combinatory equality, and yet R-reduction and η-reduction will not be jointly confluent in general — for example, if R consists of the single rule $fx \longrightarrow a$, then $\lambda x.fx$ η-reduces to f and R-reduces to $\lambda x.a$. The ADD ARGUMENT transformation on systems can be thought of as a way of "recovering" confluence.

We begin our investigation of CR-validity by considering the following set RVT of transformations obtained from VT by naively incorporating R-reduction.

Definition 2.3 Let R be a first-order term rewriting system.

1. The transformation R-REDUCE is defined by

$$\Gamma;\, \langle X, Y\rangle \longrightarrow \Gamma,\, \langle X', Y\rangle\,,$$

where $X \longrightarrow X'$ is an instance of R-reduction.

2. The set RVT consists of R-REDUCE and the transformations from VT, with the proviso that DECOMPOSE may be applied only to a pair of terms which are wR-irreducible.

The restriction on DECOMPOSE will prevent the perhaps unsound step of decomposing an actual or potential head wR-redex.

We might hope for an analogue of Theorem 2.2, but unfortunately, sufficiency can fail:

Example 2.4 Let R be the rule $fzz \longrightarrow a$ and Γ be the system $\langle f(x(SK))(x(KI)), a\rangle$, where x is a variable taking terms of the appropriate functional type to base type terms. Then Γ is non-trivial and CR-valid (since $SK =_C KI$) and yet allows no application of a rule from RVT.

On the other hand, the existence of repeated variables in the rewrite rule is the only difficulty:

Theorem 2.5 *Let R be convergent and left-linear. Then the results of Theorem 2.2 hold for RVT reduction and CR-validity.*

Proof. This is a special case of the main result of this section, Theorem 2.10 below. □

An important consequence of Theorem 2.5 will be that when R is left-linear, naively adding R-narrowing to the higher-order narrowing transformations induced by VT allows enumeration of all CR-unifiers.

The difficulty in Example 2.4 is that the terms in the "potential R-redex" corresponding to the two occurrences of z in $fzz \longrightarrow a$ are not at the "top level," and so cannot be determined CR-equal using RVT. That this obstacle can be overcome by "pre-processing" the pair $\langle x(SK), x(KI)\rangle$ suggests incorporating arbitrary convergent R into our computational paradigm by making such subterms of potential R-redexes immediately accessible. A formalism suitable for this task is that of conditional rewriting systems. This proof-theoretic tool has found use in demonstrating syntactic properties of unconditional term rewriting systems and extensions of the λ-calculus ([deV87], [deV90]), as well as in establishing consistency and uniqueness of normal forms for certain term rewriting systems whose non-left-linearity precludes confluence, so that the usual proofs of these results break down ([Klo80]).

Note that for any term rewriting system R there is a natural conditional term rewriting system determining the same equality relation, in which the unconditional parts of the conditional rules are left-linear. We may define such a conditionalization R^L of R by renaming *repeated* variable occurrences in the left-hand sides of R-rules (so that $fzz \longrightarrow a$ becomes $fz'z \longrightarrow a$ or $fzz' \longrightarrow a$ for a new variable z', for instance, rather than $fxy \longrightarrow a$ for new variables x and y) and recording the necessary constraints among the variables as a collection of equations between variables. By observing that for each rule this collection of constraints is naturally a *system* Δ, conditional rewriting can be accommodated by transformational methods as follows.

If we denote by R^0 the collection of unconditional parts of the new conditional rules (called the *unconditional part* of R^L), we may describe a potential R^L-reduction in transformational terms as an R^0-reduction together with a *witnessing system* of conditions $\sigma\Delta$, where Δ is the system of variable-equations associated with the rule from R^L and σ is the matching substitution for the R^0-reduction.

Example 2.6 If R comprises the rule $fzz \longrightarrow a$, then a conditionalization R^L of R has unconditional part $fz'z \longrightarrow a$ and variable condition $z = z'$. The potential reduction of $f(x(SK))(x(KI))$ to a using R^L is captured in transformational terms by trading the pair $\langle f(x(SK))(x(KI)), a\rangle$ for the pair $\langle a, a\rangle$ obtained by R^0-reduction, together with the witnessing system $\sigma\Delta \equiv \langle x(SK), x(KI)\rangle$, where $\sigma = \{z' \mapsto x(SK), z \mapsto x(KI)\}$.

Such a transformation of systems corresponds to a reduction by the conditional rewrite system R^L precisely when the associated witnessing system is CR-valid, so that CR-validity — checked before committing to the computational step, at a later time, or even in a piecemeal fashion — "justifies" the application of R^0-REDUCE for the linear term rewriting system R^0. Of course, an ordinary R-reduction corresponds to such a transformation of systems in which the witnessing system is trivial.

The issue of how to define the rewrite relation associated with a conditional term rewriting system does not arise in this presentation. The equational conditional systems used in our presentation could be traded in favor of conditional reduction systems or normalized conditional reduction systems ([BK86]). Although only the equational systems will in general capture the original equational theory, all three kinds of conditional term rewriting systems can be modeled by our transformational approach. It is worth remarking that the different presentations lead to different R^L-confluence results for terms, and that we circumvent this issue entirely by passing to systems.

Definition 2.7 Let R, R^L, R^0, and Δ be as in the preceding discussion.

1. The transformation R^L-REDUCE is defined by

$$\Gamma;\ \langle X, Y\rangle \longrightarrow \Gamma,\ \langle X', Y\rangle, \sigma\Delta,$$

where $X \longrightarrow X'$ is an instance of R^0-reduction with matching substitution σ and witnessing system of conditions $\sigma\Delta$.

2. The set R^LVT consists of R^L-REDUCE and the transformations from VT, with the provisos that
   - DECOMPOSE may be applied only to a pair of terms which are wR-irreducible, and
   - the witnessing system associated with an R^L-REDUCE step is non-trivial iff the redex term X is wR-irreducible.

The constraints on R^LVT-steps are computationally natural and are designed to minimize non-determinism; we will not want to perform an R^L-REDUCE step out of a term when there are wR-reductions available, and, as with the RVT transformations, neither will we wish to perform the unsound decomposition of a head R-redex. Clearly any RVT step is also an R^LVT-step, although the converse does not hold; we will treat R^LVT as an extension of RVT by identifying R-REDUCE with "trivially-conditional R^L-REDUCE."

We would like an analogue of Theorem 2.2 for the R^LVT transformations, but unfortunately, applications of R^L-REDUCE do not necessarily preserve CR-unifiers, since the witnessing system for such a step need not be CR-valid. Fortunately, we can restrict our attention to certain R^LVT-steps which do preserve CR-unifiers: *validity preserving* R^L-REDUCE steps, defined below, have witnessing systems satisfying a property both stronger and less intuitive than mere CR-validity, but the class of R^LVT-steps they determine is shown in Theorem 2.10 to capture CR-equality.

A $\mathcal{CL}$-term X is said to be in *CR-normal form* if $X \equiv \mathcal{H}(M)$ for some $\mathcal{LC}$-term M in long β-normal form. For each $\mathcal{CL}$-term X, define a new term $\widehat{X}$ to be $CRnf(X)$ if X is not a Σ-term, and $f\widehat{X_1}...\widehat{X_n}$ if $X \equiv fX_1...X_n$ for $f \in \Sigma$. For a substitution θ, let $\widehat{\theta}$ denote the substitution whose value on any variable x is $\widehat{\theta x}$.

Definition 2.8 1. A VT-step is *validity preserving* if it is an instance of TRIVIAL or ADD ARGUMENT, or an instance of DECOMPOSE such that the system obtained is CR-valid.

2. An R^L-REDUCE step with witnessing system of conditions $\sigma\Delta$ is *validity preserving* if $\widehat{\sigma}\Delta$ is trivial.

If $\Gamma \longrightarrow \Gamma'$ by a validity preserving DECOMPOSE step and Γ is CR-valid, then Γ' is CR-valid by definition. That Γ' is CR-valid when Γ is CR-valid and $\Gamma \longrightarrow \Gamma'$ by a validity preserving R^L-REDUCE step follows from the observation that, for all X and Y, $\widehat{X} \equiv \widehat{Y}$ implies $X =_{CR} Y$ (although not necessarily conversely).

The requirement that R^LVT-steps are validity preserving is somewhat stronger than what is actually needed to insure that Γ' is CR-valid when $\Gamma \longrightarrow \Gamma'$ and Γ is CR-valid, namely that $\sigma\Delta$ is CR-valid. But, as explained in the discussion following Definition 3.7 below, this stronger condition is the key to our proof of termination for sequences of R^LVT-steps.

We now revisit the earlier example which defeated RVT:

Example 2.9 Let R and Γ be as in Example 2.4. A conditionalization R^L of R has unconditional part $fz'z \longrightarrow a$ and variable condition $z' = z$. Application of R^L-REDUCE yields $\langle a, a\rangle\,, \langle x(SK), x(KI)\rangle$. DECOMPOSE and two applications of ADD ARGUMENT further give $\langle a, a\rangle\,, \langle SKpq, KIpq\rangle$. Several WEAK REDUCE steps yield $\langle a, a\rangle\,, \langle q, q\rangle$, and, anticipating the next lemma, we conclude that Γ is CR-valid. □

The following theorem embodies the facts about R^LVT critical to our program and justifies our use of validity preserving R^LVT-steps to study CR-validity of systems; it recalls Theorem 2.2.

Theorem 2.10 *Let R be convergent.*

1. *(Soundness) Suppose $\Gamma \longrightarrow \Gamma'$. Then Γ is CR-valid if Γ' is CR-valid; when the given step is validity preserving then Γ is CR-valid iff Γ' is CR-valid.*

2. *(Sufficiency) Suppose Γ is irreducible with respect to validity preserving R^LVT-steps. Then Γ is CR-valid iff it is trivial.*

3. *(Termination) Every sequence of validity preserving R^LVT-steps terminates.*

The proof of (1) is straightforward. The proofs of (2) and (3) are more delicate; the former is the focus of the remainder of this section, and the latter of the next.

We will find it convenient for the proofs of sufficiency and termination of R^LVT to consider the effects of these transformations on the individual terms appearing in a system. We define below the relation $\widehat{R}$ on terms, which describes the relationship between the redex term X of a system Γ and the term which replaces X in the system obtained by applying a validity preserving R^L-REDUCE step to Γ. Investigation of $\widehat{R}$ forms the basis of our analysis of the R^LVT transformations' expressiveness; in addition, $\widehat{R}$ is an important component of a new relation on terms whose termination will imply the termination of sequences of validity preserving R^LVT-steps.

Definition 2.11 Let R be a term rewriting system, and R^0 the unconditional part of a linearization R^L of R. The relation $\widehat{R}(X, X')$ holds for terms X and X' provided

$$\Gamma;\ \langle X, Y\rangle \longrightarrow \Gamma,\ \langle X', Y\rangle, \sigma\Delta$$

via a validity preserving R^L-REDUCE step, and X is wR-irreducible whenever $R(X, X')$ does not hold.

Note that the relation R is properly contained within the relation $\widehat{R}$, and that if R is left-linear, then $\widehat{R}$ coincides with R (throughout we will maintain a notational distinction between the term rewriting system R and the reduction relation $\xrightarrow{R}$ which R induces). Moreover, if X is algebraic, then $\widehat{R}(X, Y)$ implies $X \xrightarrow{R} Y$, so that Y is also algebraic. The condition on X corresponds exactly to the second proviso of Definition 2.7(2), and insures that $\widehat{R}$ is indeed the "term version" of validity preserving R^L-REDUCE steps from R^LVT. As per our convention, we denote by $\xrightarrow{\widehat{R}}$ the reduction relation induced by $\widehat{R}$.

Throughout the remainder of this section, we will let R denote a term rewriting system, R^0 denote the unconditional part of a linearization of R, and $\widehat{R}$ denote the relation derived from R as defined above. By *$w\widehat{R}$-reduction* we will mean the reduction relation obtained by adding $\widehat{R}$-reduction to the rules for weak reduction.

We begin the proof of Theorem 2.10(2) with a few preliminary notions and an investigation of the relationship between $\widehat{X}$ and X. The Σ-boundary of a term separates the parts of the term "near the root" (the Σ-part) from the rest of the term.

Definition 2.12 1. The *Σ-boundary* of a term T, denoted $B_\Sigma(T)$, is defined to be the empty set of occurrences if T is a variable, $\{u \in O(T) \mid \exists u_1, u_2 \text{ such that } u \equiv u_1u_2, T/u_1 \equiv T_i, u_2 \in B_\Sigma(T_i)\}$ if $T \equiv hT_1...T_n$ for $h \in \Sigma$, and the singleton set containing the empty occurrence otherwise.

2. The *Σ-part* of T, denoted $P_\Sigma(T)$, is defined by $P_\Sigma(T) = \{u \in O(T) \mid \neg\exists v \in B_\Sigma(T) \text{ such that } v \leq u\}$.

In particular, if T is not of the form $hT_1...T_n$ with $h \in \Sigma$, then $P_\Sigma(T) = \emptyset$.

Example 2.13 We have $B_\Sigma(Sxy) = \{??\}$, $B_\Sigma(xT_1...T_n) = \{??\}$, and $B_\Sigma(f(gx)(Kxy)) = \{??\}$. In addition, $P_\Sigma(Sxy) = \{??\}$, $P_\Sigma(xT_1...T_n) = \{??\}$, and $P_\Sigma(f(gx)(Kxy)) = \{??\}$.

Lemma 2.14 *1. If X is algebraic, then $\widehat{X} \equiv X$.*

2. If $u \in P_\Sigma(X)$, then $\widehat{X/u} \equiv \widehat{X}/u$.

3. Suppose X is in weak normal form. If X is not a Σ-term, then neither is $\widehat{X} \equiv CRnf(X)$.

Proof. The proofs of (1) and (2) are routine. To prove (3), notice that if X is not a Σ-term, then the translation under Λ of X is either an abstraction term or of the form $hM_1...M_n$ with $h \notin \Sigma$. The long βR-normal form of $\Lambda(X)$ is thus also of this form, and therefore its translation under $\mathcal{H}$ does not have a symbol from Σ at the head. □

Lemma 2.15 *If X is in weak normal form and $\widehat{X} \equiv \sigma S$, where S is algebraic, then there exists a substitution θ such that $X \equiv \theta S^0$ and $\widehat{X} \equiv \widehat{\theta} S$.*

Proof. It suffices to exhibit a substitution θ such that $X \equiv \theta S^0$ and $\sigma \equiv \widehat{\theta}[Vars(S)]$. Since X is in weak normal form, Lemma 2.14(3) guarantees that $B_\Sigma(X) = B_\Sigma(\widehat{X})$, so that every occurrence in S^0 is also an occurrence in X, and therefore in $\widehat{X}$. Define the substitution θ by $\theta(x) \equiv X/u$ when $S^0/u \equiv x$. Clearly $X \equiv \theta S^0$. To see that $\widehat{\theta} \equiv \sigma[Vars(S)]$, choose v such that $x \equiv S/v \equiv S^0/v$. Then

$$\sigma(x) \equiv \widehat{X}/v \equiv \widehat{X/v} \equiv \theta(\widehat{S^0/v}) \equiv \theta(\widehat{S/v}) \equiv \widehat{\theta} x$$

as desired. □

So if X is not an R^0-redex, then $\widehat{X}$ cannot be an R-redex. In fact, for wR-normal forms, a stronger restriction holds.

Lemma 2.16 *Let X be in wR-normal form. If X is not an $\widehat{R}$-redex, then $\widehat{X}$ is not an R-redex.*

Proof. If $\widehat{X} \equiv \sigma S$ for some rule $S \longrightarrow T$ in R, then by the previous lemma, there exists a substitution θ with $X \equiv \theta S^0$ and $\widehat{X} \equiv \widehat{\theta} S$, and so X is an R^0-redex. Since X is in wR-normal form, it is immediate that X is in fact an $\widehat{R}$-redex. □

Consequently, the CR-normal forms of these wR-normal forms have a particularly nice form.

Lemma 2.17 *Let X be in wR-normal form. If X is not an $\widehat{R}$-redex, then $CRnf(X) \equiv \widehat{X}$.*

Proof. By induction on X. If X is not a Σ-term, then the result holds by definition of $\widehat{X}$. Otherwise, $X \equiv fX_1...X_n$, and by the induction hypothesis, $\widehat{X} \equiv f\, CRnf(X_1)...CRnf(X_n)$. By Lemma 2.16, $\widehat{X}$ is not an R-redex, and so we easily see that $CRnf(X) \equiv \widehat{X}$. □

In particular, if X is an atomic type $w\widehat{R}$-normal form, then $CRnf(X) \equiv \widehat{X}$.

Lemma 2.18 *If $X \equiv hX_1...X_n$ is an atomic type $w\widehat{R}$-normal form, then $CRnf(X) \equiv h\, CRnf(X_1)...CRnf(X_n)$.*

Proof. In fact any atomic type $w\widehat{R}$-normal form looks like $hX_1...X_n$, where h is a constant or a variable. The proof is a routine induction on X using Lemma 2.17 and considering cases according as X is or is not a Σ-term. □

Proof of Theorem 2.10(2): One direction is immediate. For the other, it suffices to see that if some CR-valid and non-trivial pair $\langle X, Y\rangle$ from Γ is irreducible with respect to ADD ARGUMENT, WEAK REDUCE, and validity preserving R^L-REDUCE steps, then a validity preserving DECOMPOSE step applies out of $\langle X, Y\rangle$. The proof proceeds by induction on X.

For such a pair $\langle X, Y\rangle$, $X \equiv hX_1...X_n$ and $Y \equiv h'Y_1...Y_m$, and both are atomic-type terms in $w\widehat{R}$-normal form. By Lemma 2.18, $CRnf(X) \equiv h\, CRnf(X_1)...CRnf(X_n)$ and $CRnf(Y) \equiv h'\, CRnf(Y_1)...CRnf(Y_m)$, and since Γ is CR-valid, we must have $CRnf(X) \equiv CRnf(Y)$. Then $h \equiv h'$, $n = m$, and $CRnf(X_i) \equiv CRnf(Y_i)$ for $i = 1, ..., n$, so that $\langle X, Y\rangle$ admits a validity preserving DECOMPOSE step. □

In the next section we prove part (3) of Lemma 2.10.

3 Termination of validity preserving R^LVT-steps

Our approach to proving termination is to identify every system with the multiset of terms occurring in it, and define a new relation on *terms* with the property that each validity preserving R^LVT-step replaces one or two terms in a system with terms which are derived from them by this new relation. The proof then proceeds by the standard technique of multiset induction ([Der87]), provided we show that whenever R is convergent, this new relation is terminating on all terms. The relation which will support this strategy is given by

Definition 3.1 The relation $X \xrightarrow{CR^+} Y$ holds if either

- X weakly reduces to Y (write $X \xrightarrow{w} Y$),
- $Y \equiv Xd$ for a constant d (write $X \xrightarrow{aa} Y$),
- $X \equiv aX_1 ... X_n$, a is a non-redex atom, and $Y \equiv X_i$ for some i (write $X \xrightarrow{sel} Y$), or
- $X \xrightarrow{\widehat{R}} Y$.

If $X \xrightarrow{sel} Y$, we will say that Y is obtained from X by an application of Select, and if $X \xrightarrow{aa} Y$, we will say that Y is obtained from X by an application of Add Argument.

Observe that adding an argument may induce wR-reductions, extracting a subterm may increase the length of the type of a term, and wR-reduction can increase the size of a term. So it seems that no simple proof of termination is available based on an ordering of terms with respect to size, type, or R-reduction size.

Although CR^+-reduction will be seen below to fit our proof specification exactly, we consider in the next several paragraphs the pitfalls of a more naive reduction relation with which we might hope would support our strategy of multiset induction. This relation also arises by examining the effects of the various R^LVT-steps on individual terms.

Say that an R^0-reduction using rule $S^0 \longrightarrow T$ and matching substitution σ is *sound* if the redex subterm $X \equiv \sigma S^0$ satisfies $X =_{CR} \sigma S$. Write $X \hookrightarrow Y$ if Y is obtained from X via the relation which is defined exactly as in Definition 3.1, except that the last clause there is replaced by: $X \longrightarrow Y$ via a sound R^0-reduction.

An obvious candidate for a measure proving termination of $\hookrightarrow$-reduction is $\mu(X) = \langle \mu_1(X), \mu_2(X), \mu_3(X) \rangle$, where $\mu_1(X)$ is the length of the type of X, $\mu_2(X)$ is the length of a longest sequence of wR^0-reductions out of X, and $\mu_3(X)$ is the size of X. But even if R is convergent, R^0-reduction need not be terminating: if R consists of the rule $fx(gx) \longrightarrow fxx$, then R is terminating, but the linearization $fx(gy) \longrightarrow fxx$ is not, since the term $f(ga)(ga)$ reduces to itself.

The R^0-reductions in this example are not sound, but unfortunately, even sequences of sound R^0-reductions need not be terminating, as can been seen by adding to this R the rule $gx \longrightarrow x$. Then $f(ga)(ga)$ reduces to itself by a sound R^0-reduction.

Since we must in fact consider sound R^0-reductions which are not R-reductions, the restriction to R-normal forms in properly conditional R^L-REDUCE steps — and therefore in $\widehat{R}$-reductions — is important. For example, permitting sound R^0-reductions only in case the redex term is in R-normal form gives the essentially unique reduction sequence $f(ga)(ga) \longrightarrow faa$ for our problematic term, and this sequence is clearly terminating.

Of course, the restriction to R-normal forms is also natural, since we will not want to introduce unnecessary non-determinism into our computations.

Insight into the weak normal form restriction in the definitions of $\widehat{R}$ and R^L-REDUCE will emerge as we consider the structure of the proof of termination of CR^+-reduction. Our proof uses a variation on the technique of logical relations, a fundamental tool in the study of simply typed λ-calculi; see [Mit90] for an introduction and references.

Let $\mathcal{T}$ denote the set of all terms which are terminating under CR^+-reduction.

Definition 3.2 Define by induction on types the sets

- $\mathcal{S}_0 = \{X | X$ is an atomic type term in $\mathcal{T}\}$, and
- $\mathcal{S}_{\alpha\to\beta} = \{X |$ for all $Y \in \mathcal{S}_\alpha, XY \in \mathcal{S}_\beta\}$.

Let $\mathcal{S}$ denote the union of the sets $\mathcal{S}_\alpha$. We may prove that a term X of type $\alpha_1 \to \alpha_2 \to \ldots \to \alpha_n \to \alpha_0$, where α_0 is an atomic type, belongs to $\mathcal{S}$ by supplying it with argument terms Y_i, $i = 1, \ldots, n$, belonging to the appropriate sets $\mathcal{S}_{\alpha_i}$ and showing that $XY_1 \ldots Y_n$ is in $\mathcal{T}$.

We will show that $\mathcal{S} \subseteq \mathcal{T}$, and that all terms are contained in $\mathcal{S}$, thereby proving termination of CR^+-reduction. Since $\mathcal{S}$ is closed under application by definition, it suffices to see that it contains all atoms. We begin with a technical fact, and then prove the required properties of $\mathcal{S}$.

Lemma 3.3 *If $X \xrightarrow{CR^+} Y$ and $X \in \mathcal{S}$, then $Y \in \mathcal{S}$.*

Proof. By induction on types. □

Lemma 3.4 *1. Every non-redex atom not in Σ is in $\mathcal{S}$.*

2. $\mathcal{S} \subseteq \mathcal{T}$.

Proof. We prove both statements simultaneously by induction on types.

- We consider first a term X of atomic type.
  - (1) If X is a non-redex atom not in Σ, then no reductions are available out of X.
  - (2) If $X \in \mathcal{S}_0$, then it is in $\mathcal{T}$ by definition.
- Consider a term X of type $\alpha \equiv \alpha_1 \to \alpha_2 \to \ldots \to \alpha_n \to \alpha_0$, with α_0 of atomic type.

- (1) We must show that if $Y_i \in \mathcal{S}_{\alpha_i}$ for $i = 1, ..., n$, then $Y \equiv XY_1...Y_n \in \mathcal{T}$. By Lemma 3.3 and the induction hypothesis, any CR^+-reduction sequence out of Y involving no head reduction must be finite. So there must be a head reduction in the sequence, and the first such is then necessarily an application of Select. Without loss of generality, the head reduction may be promoted so that it is the first reduction in the sequence: otherwise the sequence is $XY_1...Y_n \twoheadrightarrow XY_1'...Y_n' \xrightarrow{sel} Y_i' \twoheadrightarrow ...$ and Lemma 3.3 applies so that $Y_i' \in \mathcal{T}$ by the induction hypothesis. We may therefore assume that the sequence is $hY_1...Y_n \xrightarrow{sel} Y_i \twoheadrightarrow ...$ But note that $Y_i \in \mathcal{T}$ by the induction hypothesis.
- (2) Let $X \in \mathcal{S}$ and consider an infinite CR^+-reduction sequence

$$\rho : X \equiv Z_0 \longrightarrow Z_1 \longrightarrow Z_2 \longrightarrow ...$$

If ρ consists of some number of $w\widehat{R}$-reductions and applications of Select followed by an application of Add Argument, then let d be the constant introduced in the application of Add Argument; otherwise, let d be an arbitrary argument. Consider the sequence

$$\rho d : Xd \longrightarrow Z_1 d \longrightarrow Z_2 d \longrightarrow ...$$

where each reduction performed in ρd is precisely the reduction performed in ρ. Then ρd either eventually coincides with ρ (if the sequence uses an application of Add Argument or Select), or else exactly mirrors ρ. In either case, ρd is infinite, contradicting the induction hypothesis on Xd.

□

Lemma 3.5 $\{I, K, S\} \subseteq \mathcal{S}$.

Proof. The proofs for all three redex atoms proceed according to the following scheme using induction over types: consider the atomic-type term Y obtained by applying the redex atom in question to an appropriate number of argument terms from $\mathcal{S}$, and suppose that Y admits an infinite CR^+-reduction sequence. If no head reduction were done in this sequence, then some one of these argument terms would also admit an infinite CR^+-reduction sequence, and this would contradict Lemma 3.4(2). So there must be a head reduction in any infinite CR^+-reduction sequence out of Y. Without loss of generality we may assume that the first reduction in the sequence is a head reduction, since otherwise the first head reduction may be promoted (the sequence $IZ_1 \twoheadrightarrow IZ_1' \longrightarrow Z_1' \twoheadrightarrow ...$, for example, may be traded for the sequence $IZ_1 \longrightarrow Z_1 \twoheadrightarrow Z_1' \twoheadrightarrow ...$, and similarly for sequences involving K and S). It is easy to check that in all three cases, the fact that the argument terms are all in $\mathcal{S}$ implies that the head reduct, and therefore Y, is also in $\mathcal{S}$. □

The remainder of the section will be devoted to the proof of the next lemma; an informal discussion of its main ideas preceeds the proof. Our proof draws heavily on the ideas of [BG91a].

Lemma 3.6 $\Sigma \subseteq \mathcal{S}$.

We begin by describing the proof in outline. We aim to show that if $f \in \Sigma$ and $X_1, ..., X_n$ are terms in $\mathcal{S}$, then $X \equiv fX_1...X_n$ is an atomic-type CR^{+}-strongly normalizing term. Let $w(X)$ denote the weak normal form of the $\mathcal{CL}$ term X.

For an arbitrary infinite CR^{+}-reduction sequence

$$\rho : X \equiv Z_0 \longrightarrow Z_1 \longrightarrow Z_2 \longrightarrow \dots ,$$

each Z_i can be shown to be a Σ-term, so that ρ is really a sequence of $w\widehat{R}$-reductions and applications of Select. In fact, infinitely many of the reductions in ρ must take place "near the roots" of the terms Z_i, and therefore ρ induces an infinite sequence

$$\rho_w : w(Z_{i_0}) \longrightarrow w(Z_{i_1}) \longrightarrow \dots$$

of $\widehat{R}$-reductions and applications of Select. Infinitely many of the reductions in ρ_w are also "near the roots," and therefore "projectable" onto the Σ-parts of the terms $w(Z_{i_j})$, in a sense to be made precise by Definition 3.7. An infinite sequence ρ_w^{π} of R-reductions and applications of Select out of the Σ-parts of the terms $w(Z_{i_j})$ is in turn induced by ρ_w. But algebraic terms are easily seen to be terminating under Select and R-reduction, and we conclude that no such sequence ρ can exist, i.e., that f is indeed in $\mathcal{S}$.

To make this argument precise, we will require the following notion.

Definition 3.7 Associate to each CR-equivalence class of terms an infinite set of variables. For every term X define a new term X^{π}, called the *projection* of X, as follows:

- if X is not a Σ-term, then $X^{\pi} \equiv z$ for a fresh variable z associated with the CR-equivalence class of X, and
- otherwise, if $X \equiv fX_1...X_n$, then $X^{\pi} \equiv fX_1^{\pi}...X_n^{\pi}$.

More precisely, Definition 3.7 defines a *class* of projections of each term, unique up to variable renaming. But since any sequence $X \equiv Z_0 \longrightarrow Z_1 \longrightarrow Z_2 \longrightarrow \dots$ of CR^{+}-reductions contains only finitely many variables, we may assume without loss of generality that in constructing projections for the terms Z_i, all CR-equivalent subterms of all of the terms Z_i are replaced by the same fresh variable. In particular, CR-equivalent subterms of a single term Z_i are replaced by the same fresh variable.

Initially, we might try to pass directly from ρ to a sequence ρ^{π} of reductions on the Σ-parts of the terms Z_i without considering the effects of weak reduction on the Σ-parts of terms. In this situation we would not intermediately consider the weak normal forms of the terms Z_i in projecting reduction sequences, and would modify the definition of $\widehat{R}$ by omitting in Definition 2.11 the requirement that X be wR-irreducible when $R(X, X')$ does not hold. But the next example shows that, in passing from a term which is not in weak normal form to its Σ-part, projection would not necessarily preserve $\widehat{R}$-redexes for such a modified $\widehat{R}$ if R is not left-linear.

Example 3.8 Let R consist of the rule $gxx \longrightarrow b$. Then $gx(Kxy)$ admits a modified $\widehat{R}$-reduction, while $[gx(Kxy)]^{\pi} \equiv gxz$ does not.

The fact that $R \xrightarrow{\widehat{R}} Y$ implies $X^{\pi} \xrightarrow{\widehat{R}} Y^{\pi}$ if X is in weak normal form (indeed $X^{\pi} \overset{R}{\twoheadrightarrow}{}^{+} Y^{\pi}$ — see Lemma 3.11) motivates the introduction of the intermediate sequence ρ_w, and underscores the necessity of the second proviso in Definition 2.7(2) and the corresponding condition in Definition 2.11.

The problem in Example 3.8 can be summarized by saying that weak reductions can alter the Σ-boundaries of terms. Unfortunately R-reductions — and therefore replacing subterms of terms by their CR-normal forms — can also alter Σ-boundaries. Recalling that every $\widehat{R}$-redex is "almost σS" for some rule $S \longrightarrow T$ in R, in the sense that for every x_1 and x_2 in S^0 which correspond to different occurrences of the variable x in S, $\widehat{\sigma}x_1 \equiv \widehat{\sigma}x_2$, we see that requiring only that $\sigma\Delta$ be CR-valid Definition 2.8 is too permissive to insure that $X^\pi \xrightarrow{\widehat{R}} Y^\pi$ whenever $X \xrightarrow{\widehat{R}} Y$. But insisting instead that $\widehat{\sigma}\Delta$ be trivial is enough to guarantee that Σ-boundaries are preserved under projection, as desired.

Ultimately, the motivation for investigating the sequence ρ^π_w is that we do not need to know exactly what lies below the Σ-boundaries of the terms Z_i in ρ to understand the essential nature of an infinite CR^+-reduction sequence; we need only know that the reductions in ρ reflect validity preserving R^LVT-steps and that their Σ-boundaries are preserved by projection. Although we may not be able to enforce any constraints on the *entire* substitution parts of the terms Z_i in ρ, it will turn out that we *can* at least insist that their non-Σ-parts behave nicely. Indeed, we can arrange that if $S \longrightarrow T$ is a rule in R, then for each subterm σS^0 of one of the terms Z_i in which x_1 and x_2 correspond to different occurrences of the same variable in S, the CR-normal forms of the non-Σ-parts of σx_1 and σx_2 are identical, and the Σ-parts of σx_1 and σx_2 are identical. The requirement that $\widehat{\sigma}x_1 \equiv \widehat{\sigma}x_2$ is exactly this constraint.

We begin the formal proof by collecting some easy facts about X^π.

Lemma 3.9 *1. Let X be a Σ-term. If $X \xrightarrow{sel} Y$, then $X^\pi \xrightarrow{sel} Y^\pi$.*

2. $\widehat{X^\pi} \equiv X^\pi$.

3. If S is algebraic, $\sigma^\pi S \equiv (\sigma S)^\pi$, where σ^π is such that for every variable x, $\sigma^\pi x \equiv (\sigma x)^\pi$.

4. If X and Y are atomic type weak normal forms such that $\widehat{X} \equiv \widehat{Y}$, then $X^\pi \equiv Y^\pi$.

Proof. Straightforward. □

The following definition will be useful in analyzing the effects of CR^+-reductions on terms.

Definition 3.10 For every term X, $Bot(X) = \{Y | \exists d \in B_\Sigma(X) \text{ such that } Y \equiv X/d\}$.

It is easy to see that if $Y \in Bot(X)$ then $X \xrightarrow{sel}\!\!\!\!\rightarrow Y$, and if S is algebraic then $Bot(\sigma S) = \bigcup\{Bot(\sigma x) | x \in Vars(S)\}$.

Say that a CR^+-reduction out of X is a *bottom reduction* if the redex subterm of X is in $Bot(X)$, and a *top reduction* otherwise. The following lemma shows that top $\widehat{R}$-reductions yield non-identical terms in the passage from ρ_w to ρ^π_w.

Lemma 3.11 *If $X \xrightarrow{\widehat{R}} Y$ and X is in weak normal form, then $X^\pi \xrightarrow{R}\!\!\!\!\rightarrow Y^\pi$, and if the first reduction is a top reduction, then $X^\pi \not\equiv Y^\pi$.*

Proof. The proof is by induction on X. If X is not a Σ-term, neither is Y, so that $X^\pi \equiv Y^\pi$. Otherwise, $X \equiv fX_1...X_n$ for $f \in \Sigma$.

If $\widehat{R}(X,Y)$ does not hold, then $Y \equiv fX_1...X'_k...X_n$, so that by the induction hypothesis, $X_k^\pi \xrightarrow{R}\!\!\!\!\rightarrow X_k'^\pi$, and therefore $X^\pi \xrightarrow{R}\!\!\!\!\rightarrow Y^\pi$. If $X \xrightarrow{\widehat{R}} Y$ is a top redex, then so is $X_k \xrightarrow{R} X'_k$. Thus, $X_k^\pi \not\equiv X_k'^\pi$, and so $X^\pi \not\equiv Y^\pi$.

If $\widehat{R}(X,Y)$ using a rule $S \longrightarrow T$ in R and matching substitution σ, then $X^\pi \xrightarrow{R^0} Y^\pi$ also using $S \longrightarrow T$ but with matching substitution σ^π. To see that in fact, $X^\pi \xrightarrow{R} Y^\pi$, observe that for all x_1 and x_2 in S^0 replacing the same variable in S, $\sigma^\pi(x_1) \equiv (\sigma x_1)^\pi$ and $\sigma^\pi(x_2) \equiv (\sigma x_2)^\pi$, and σx_1 and σx_2 are atomic-type weak normal forms such that $\widehat{\sigma x_1} \equiv \widehat{\sigma x_2}$. Then $\widehat{\sigma^\pi x_1} \equiv \sigma^\pi x_1 \equiv (\sigma x_1)^\pi \equiv (\sigma x_2)^\pi \equiv \sigma^\pi x_2 \equiv \widehat{\sigma^\pi x_2}$, where $(\sigma x_1)^\pi \equiv (\sigma x_2)^\pi$ by Lemma 3.9. □

The next corollary guarantees that termination of bottom terms is closed under $w\widehat{R}$-reductions and applications of Select.

Corollary 3.12 *If $X \xrightarrow{CR^+} X'$ using no applications of Add Argument, then for all $Y' \in Bot(X')$ there exists $Y \in Bot(X)$ such that $Y \xrightarrow{CR^+}\!\!\!\!\rightarrow Y'$. Moreover, if the first reduction is a bottom reduction then there exist terms $Y \in Bot(X)$ and $Y' \in Bot(X')$ such that $Y \xrightarrow{CR^+}\!\!\!\!\rightarrow^+ Y'$.*

Proof. The assertion is proved by induction on X. If X is not a Σ-term, then $Bot(X) = \{X\}$, so we may take $Y \equiv X$. Then if $Y' \in Bot(X')$, we have $X' \xrightarrow{sel}\!\!\!\!\rightarrow Y'$, so that $X \xrightarrow{CR^+} X' \xrightarrow{sel}\!\!\!\!\rightarrow Y$, i.e., $X \xrightarrow{CR^+}\!\!\!\!\rightarrow^+ Y'$. Otherwise, X is of the form $fX_1...X_n$.

If X' is obtained from X by a head $\widehat{R}$-reduction or an application of Select, then if $Y' \in Bot(X')$, we have $Y' \in Bot(X)$, and so we may take $Y \equiv Y'$.

If $X \equiv fX_1...X_n \xrightarrow{CR^+} fX_1...X'_k...X_n \equiv X'$, then either $Y \in Bot(X_i)$ for some $i \neq k$, or else $Y \in Bot(X'_k)$. In the first case, $Y' \in Bot(X_i) \subseteq Bot(X)$, so we may take $Y \equiv Y'$. If $Y' \in Bot(X'_k)$, then by the induction hypothesis, there exists $Y \in Bot(X_k)$ such that $Y \xrightarrow{CR^+}\!\!\!\!\rightarrow Y'$, and this Y is also in $Bot(X)$. Moreover, if the original reduction is a bottom reduction, then $X_k \xrightarrow{CR^+}\!\!\!\!\rightarrow X'_k$ is as well, and so by the induction hypothesis, there exist terms $Y \in Bot(K_k)$ and $Y' \in Bot(X'_k)$ such that $Y \xrightarrow{CR^+}\!\!\!\!\rightarrow^+ Y'$. But then $Y \in Bot(X)$ and $Y' \in Bot(X')$ as desired. □

Corollary 3.13 *If $Bot(X) \subseteq \mathcal{T}$ and $X \xrightarrow{CR^+}\!\!\!\!\rightarrow X'$ not using any applications of Add Argument, then $Bot(X') \subseteq \mathcal{T}$.*

Proof. It suffices to see that if $X \xrightarrow{CR^+} X'$ not using any applications of Add Argument, then $Bot(X') \subseteq \mathcal{T}$; the result is then obtained by induction on the length of the reduction sequence $X \xrightarrow{CR^+}\!\!\!\!\rightarrow X'$. Let $Y \in Bot(X')$. Then there exists $Y \in Bot(X)$ such that $Y \xrightarrow{CR^+}\!\!\!\!\rightarrow Y'$, and since $Y \in \mathcal{T}$, we must have $Y' \in \mathcal{T}$, too. □

For the proof of termination, we will want to see that if $X \xrightarrow{CR^+} Y$, then $w(X) \xrightarrow{CR^+}\!\!\!\!\rightarrow w(Y)$, and if the first reduction is a top reduction, then $w(X) \xrightarrow{CR^+}\!\!\!\!\rightarrow^+ w(Y)$ with at least one top reduction done. That is, we must see that the various top CR^+-reductions are preserved in passing from ρ to ρ_w.

It is easy to see that $X \xrightarrow{w} Y$ implies $w(X) \overset{w}{\twoheadrightarrow} w(Y)$, and the original reduction cannot be a top reduction. We also have that $X \xrightarrow{sel} Y$ implies $w(X) \xrightarrow{sel} w(Y)$, and both reductions are top reductions. But we further require that if $X \xrightarrow{\widehat{R}} Y$, then $w(X) \overset{\widehat{R}}{\twoheadrightarrow} w(Y)$ and when the first reduction is a top reduction, there must be at least one top $\widehat{R}$-reduction in $w(X) \overset{\widehat{R}}{\twoheadrightarrow} w(Y)$ (see Corollary 3.18); we will make use of the following "parallelized" version of $\widehat{R}$-reduction in proving this.

Definition 3.14 Define $\mapsto$ to be the least relation such that

1. $\equiv \subseteq \mapsto$
2. $\widehat{R} \subseteq \mapsto$
3. if $A_i \mapsto B_i$ for $i = 1, ..., n$, then $A_1 ... A_n \mapsto B_1 ... B_n$.

An $\mapsto$-reduction is *proper* if it is either an $\widehat{R}$-reduction, or if it is an application of the third clause of Definition 3.14, where the head symbol of A_1 is a constant from Σ and for some i, $A_i \mapsto B_i$ by a proper $\mapsto$-reduction.

Lemma 3.15 *1. If $X \mapsto Y$ but not $\widehat{R}(X,Y)$, then the head symbols of X and Y are identical.*

2. $\xrightarrow{\widehat{R}} \subseteq \mapsto \subseteq \overset{\widehat{R}}{\twoheadrightarrow}$. Moreover, if $X \mapsto Y$ is a proper reduction, then $X \overset{\widehat{R}}{\twoheadrightarrow}^{+} Y$ and at least one top reduction is done in this reduction sequence.

3. If $X \xrightarrow{\widehat{R}} Y$ by a top reduction, then $X \mapsto Y$ via a proper reduction.

Proof. The first two statements are proved by induction over the shortest derivation of $X \mapsto Y$; the third is proved by routine induction on X. □

Lemma 3.16 *If $\widehat{R}(X,Y)$, then $\widehat{R}(w(X), w(Y))$.*

Proof. If $R(X,Y)$ does not hold, the $w(X) \equiv X$ and $w(Y) \equiv Y$, and there is nothing to prove. If $R(X,Y)$, then $X \equiv \theta S$ and $Y \equiv \theta T$ for some rule $S \longrightarrow T$ in R and substitution θ. Then $w(X) \equiv \theta' S$ and $w(Y) \equiv \theta' T$ for the substitution θ' such that $\theta'(x) \equiv w(\theta x)$ for all variables x, and so indeed $R(w(X), w(Y))$.

The "parallelization" of $\widehat{R}$ in Definition 3.14 exactly supports $\widehat{R}$-reduction out of weak normal forms.

Proposition 3.17 *If $X \mapsto Y$, then $w(X) \mapsto w(Y)$, and if the first reduction is proper, then so is the second.*

Proof. By induction over the measure $\tau(X) = \langle \tau_1(X), \tau_2(X) \rangle$, where $\tau_1(X)$ is the number of wR-reductions out of X and $\tau_2(X)$ is the size of X.

If X is in weak normal form, then since $X \mapsto Y$, we have $X \overset{R^0}{\twoheadrightarrow} Y$, and since algebraic rewriting does not introduce new weak redexes, Y must be in weak normal form. Thus $w(X) \equiv X \mapsto Y \equiv w(Y)$.

If X is not a head weak redex, then $X \equiv hX_1...X_n$ and either h is a non-redex atom or else X is not of atomic type. We consider cases for the reduction $X \mapsto Y$:

- If $X \equiv Y$ there is nothing to prove.
- If $\widehat{R}(X,Y)$, then by Lemma 3.16, $\widehat{R}(w(X), w(Y))$.
- Otherwise, since X and Y must have identical head symbols by Lemma 3.15, $Y \equiv hY_1...Y_n$ and $X_i \mapsto Y_i$ for $i = 1, ..., n$. If $X \mapsto Y$ is proper, then $h \in \Sigma$ and there exists a k such that $X_k \mapsto Y_k$ is proper. Then by the induction hypothesis, $w(X_i) \mapsto w(Y_i)$ for $i = 1, ..., n$, so that $w(X) \mapsto w(Y)$, and, moreover, if $X \mapsto Y$ is proper, then so is $X_k \mapsto Y_k$ for some k, and $w(X_k) \mapsto w(Y_k)$ is as well. But then since $h \in \Sigma$, $w(X) \mapsto w(Y)$ is also proper.

If X is a head weak redex, then $\widehat{R}(X,Y)$ does not hold. Assume without loss of generality that $X \not\equiv Y$. Then the head of Y is identical to that of X, and the argument terms in Y are all obtained from its argument terms in X by $\mapsto$-reduction. Let X_0 and Y_0 be the terms obtained from X and Y, respectively, by doing the head weak reductions. Then $X_0 \mapsto Y_0$, and by the induction hypothesis, we have $w(X) \equiv w(X_0) \mapsto w(Y_0) \equiv w(Y)$. □

Top $\widehat{R}$-reductions are thus preserved in the passage from ρ to ρ_w:

Corollary 3.18 *If $X \xrightarrow{\widehat{R}} Y$, then $w(X) \overset{\widehat{R}}{\twoheadrightarrow} w(Y)$, and if the first is a top reduction, then $w(X) \overset{\widehat{R}}{\twoheadrightarrow}^{+} w(Y)$ with at least one top reduction done in this sequence.*

Proof. Straightforward application of Lemmas 3.15 and 3.17. □

The final fact we will need is easily stated and proved.

Lemma 3.19 *If X is algebraic, then every sequence of R-reductions and applications of Select originating in X terminates.*

Proof. By induction over the measure $\nu(X) = \langle \nu_1(X), \nu_2(X) \rangle$, where $\nu_1(X)$ is the shortest length of an R-reduction out of X and $\nu_2(X)$ is the size of X. □

We are now in a position to prove Lemma 3.6.

Proof of Lemma 3.6: Let $X_i \in \mathcal{S}$ for $i = 1, ..., n$. We show that $X \equiv fX_1...X_n \in \mathcal{T}$. Note that by Lemma 3.4, $X_i \in \mathcal{T}$ for each i.

Since for any $Y \in Bot(X)$ there exists an $i \in \{1, ..., n\}$ such that $X \xrightarrow{sel} X_i \overset{sel}{\twoheadrightarrow} Y$, and since $X_i \in \mathcal{S}$ and $\mathcal{S}$ is closed under $\overset{CR^+}{\twoheadrightarrow}$, by Lemmas 3.3 and 3.4 we have that $Y \in \mathcal{T}$. So for every term X' such that $X \overset{CR^+}{\twoheadrightarrow} X'$ without using an application of Add Argument, $Bot(X') \subseteq \mathcal{T}$ by Corollary 3.13.

Now, let

$$\rho : X \equiv Z_0 \longrightarrow Z_1 \longrightarrow Z_2 \longrightarrow ...$$

be an infinite CR^+-reduction sequence out of X. If some Z_i is not a Σ-term, then the first such Z_i must be obtained from Z_{i-1} by either an application of Select or $\widehat{R}$. In both cases, Z_i must be in $Bot(Z_{i-1})$, and therefore $Z_i \in \mathcal{T}$. Thus, every term in the infinite sequence ρ must in fact be a Σ-term, and consequently, no applications of Add Argument are done in ρ.

If only finitely many top reductions are done in ρ, let Z_i be the result of the last one. Then some term $Y \in Bot(Z_i)$ must admit an infinite sequence of $w\widehat{R}$-reductions and applications of Select. But this is impossible, since $Bot(Z_i) \subseteq \mathcal{T}$ for each i by Corollary 3.13.

So there must be infinitely many top reductions in ρ. Consider the sequence of reductions induced by ρ, given by

$$\rho_w : w(Z_{i_0}) \longrightarrow w(Z_{i_1}) \longrightarrow w(Z_{i_2}) \longrightarrow \ldots.$$

This sequence of $\widehat{R}$-reductions and applications of Select exists by Corollary 3.18 and the fact that Select only applies to terms whose head symbols are non-redex atoms. Since each top $\widehat{R}$-reduction in ρ yields at least one top $\widehat{R}$-reduction in ρ_w and each application of Select in ρ yields one in ρ_w, the sequence ρ_w contains infinitely many top $\widehat{R}$-reductions and applications of Select.

Consider, finally, the sequence

$$\rho_w^\pi : (w(Z_{i_0}))^\pi \longrightarrow (w(Z_{i_1}))^\pi \longrightarrow \ldots$$

of algebraic terms. By Lemmas 3.11 and 3.9(1) this sequence consists of infinitely many R-reductions and applications of Select. But by Lemma 3.19, every such sequence of reductions is terminating. Thus no infinite sequence ρ can exist, so that $f \in \Sigma$, completing the proof. □

Taken together, Lemmas 3.3 through 3.6 prove Theorem 2.10(3).

Theorem 3.20 *Every $\mathcal{CL}$-term is in $\mathcal{T}$.*

4 The Unification Problem

We show that we may generate a complete set of higher-order E-unification transformations from R^LVT in precisely the same way that narrowing transformations are derived from a convergent term rewriting system.

4.1 Transformations for higher-order E-unification

Recall that a typical narrowing step involves syntactic unification. Since terms may have incompletely specified types in our setting, the syntactic unification here must be rich enough to incorporate type-unifications needing to be performed. In particular, in applying the transformations of the next definition, the computation of the unifier μ implicitly involves some type unification; accordingly, we will take as our transformations for syntactic unification Eliminate, Decompose, and Type-Unify as defined in [Dou93] together with a transformation which deletes trivial pairs of a system; the proof given there is easily adapted to show soundness and syntactic unification completeness of this set of transformations.

Definition 4.1 Let R be a convergent term rewriting system, and R^0 the unconditional part of a conditionalization of R. The set R^LUT is obtained by adding the following transformations to those for syntactic unification:

- *Weak Narrow:*
$$\Gamma;\, \langle X, Y\rangle \Longrightarrow [\mu],\ \mu\Gamma,\ \langle \mu X^*, \mu Y\rangle\,,$$
where there exists a non-variable occurrence u of X and a weak reduction rule $S \longrightarrow T$ with fresh variables such that μ is a most general unifier of X/u and S, and $X^* \equiv X[u \leftarrow T]$.

- *Add Argument:*
$$\Gamma;\, \langle X, Y\rangle \Longrightarrow [\mu],\ \mu\Gamma,\ \langle (\mu X)d, (\mu Y)d\rangle\,,$$
where $\mu \equiv (\pi \to \pi')$ is a most general type-unifier of the set consisting of the type of X, the type of Y, and (just in case these are each atomic types) the type $(s \to t)$, for fresh type-variables s and t, and where d is built from the first fresh parameter in *Args*, given type π.

- *Split:*
$$\Gamma;\, \langle xX_1 \cdots X_n, hZ_1 \cdots Z_m Y_1 \cdots Y_n\rangle \Longrightarrow$$
$$[\mu],\ \mu\Gamma,\ \langle z_1, \mu Z_1\rangle, \ldots, \langle z_m, \mu Z_m\rangle, \langle \mu X_1, \mu Y_1\rangle, \ldots, \langle \mu X_n, \mu Y_n\rangle\,,$$
where $m, n \geq 0$, $x \in \mathit{Vars}$, h is a pure atom, each z_i is a fresh indeterminate given the same type as Z_i, $1 \leq i \leq m$, and μ is a most general unifier of x and $hz_1 \cdots z_m$.

- *R^L-Narrow*
$$\Gamma;\, \langle X, Y\rangle \Longrightarrow [\mu], \mu\Gamma, \langle \mu X^*, \mu Y\rangle, \mu\Delta,$$
when there exists a non-variable occurrence u of X and a rule $S^0 \longrightarrow T$ in R^0 with fresh variables and associated witnessing system of conditions $\mu\Delta$ such that μ is a most general unifier of X/u and S^0, and $X^* \equiv X[u \leftarrow T]$.

We adopt the convention that no R^LUT-step is to be done out of a solved or trivial pair. This respects the intuition that the solved part of a system is merely a record of an answer substitution being constructed.

To see the need for unification with $(s \to t)$ in the definition of *Add Argument*, suppose $X \equiv Z_1^{\alpha \to u} a^\alpha$ and $Y \equiv Z_2^{\beta \to v} b^\beta$ for some types α and β and type-variables u and v. In order that we can apply *Add Argument* to $\langle X, Y\rangle$, we require that both X and Y be of functional type. The way we enforce this constraint is by unifying the type variables u and v with one another as well as with a "generic functional type" $(s \to t)$.

An implementation of R^LUT would presumably not treat *Add Argument* as a separate transformation, but would rather incorporate it into more generous versions of *Weak Narrow* and *R^L-Narrow* which supply arguments as needed. It is easier to analyze the transformations separately, however, and we want to emphasize the fact that the R^LUT transformations are immediately derived from the R^LVT transformations.

We will need to be careful about the set of variables occurring in a system. It is easily checked that if $\Gamma \Longrightarrow \Gamma'$, then $\mathit{Supp}(\Gamma) \subseteq \mathit{Supp}(\Gamma')$. In addition, solved variables remain solved after application of an R^LVT transformation, i.e., if $\Gamma \Longrightarrow \Gamma'$, then $\{x | x \text{ is solved in } \Gamma\} \subseteq \{x | x \text{ is solved in } \Gamma'\}$. The verification relies on the conventions that the transformations are not performed on trivial pairs and that distinct terms do not have the same type-erasures (the latter insures that distinct variables are not identified after application of a type-substitution).

To respect the intuition that constants from *Args* are not part of our unification problems and are introduced only as dummy arguments, we must confine our attention to pure problems and substitutions. Of course, any problem can be considered a pure one by suitably defining *Args*.

Our method of computing CR-unifiers is sound:

Lemma 4.2 (Soundness) *If θ is a pure CR-unifier of Γ' and $\Gamma \Longrightarrow \Gamma'$ via an R^L*UT-*step, then $\theta\Gamma$ is CR-valid.*

Proof. Use the notation of Definition 4.1. Our hypothesis entails that $\theta[\mu]$ is CR-valid, so $\mu \leq_{CR} \theta$, and hence $\theta\mu =_{CR} \theta$. It follows that $\theta\mu\Gamma =_{CR} \theta\Gamma$, and so we need only show that θ CR-unifies the "redex-pair" of the transformation.

When the transformation is *Weak Narrow* or R^L-*Narrow*, the argument is exactly as for first-order narrowing.

When the transformation is *Add Argument* we want to see that $\theta X =_{CR} \theta Y$. But $\theta((\mu X)d) =_{CR} \theta((\mu Y)d)$, that is, $(\theta X)(\theta d) =_{CR} (\theta Y)(\theta d)$ and we may invoke the extensionality rule since θ is pure and so θd is guaranteed to be new to $\langle \theta X, \theta Y \rangle$.

In the case of *Split*, the fact that $\theta X_i =_{CR} \theta\mu X_i =_{CR} \theta\mu Y_i =_{CR} \theta Y_i$ for $1 \leq i \leq n$ implies that we need only argue that $\theta \langle x, hZ_1 \cdots Z_m \rangle$ is CR-valid. We compute: $\mu x \equiv \mu h z_1 \cdots z_m$ by definition of μ, so $\theta x =_{CR} \theta\mu x \equiv \theta\mu(hz_1 \cdots z_m) =_{CR} \theta(hz_1 \cdots z_m)$, but our hypothesis implies that for each i, $\theta z_i =_{CR} \theta Z_i$. □

We now address completeness. The Lifting Lemma below is the main tool for proving that for any system Γ, the set of transformations R^LUT can enumerate a complete set of CR-unifiers for Γ when R is convergent. For its statement and proof we require the following notion. Denote by $D\theta$ the domain of a substitution θ.

Definition 4.3 A pure idempotent substitution θ is a *normalized CR-unifier* of a system Γ if

- $D\theta_0$ and the type-erasures of the terms in $D\theta_1$ are contained in $Supp(\Gamma)$,
- $\theta\Gamma$ is CR-valid, and
- for each unsolved variable x of Γ, θx is in CR-normal form.

Write NCRU(Γ) for the set of normalized CR-unifiers of Γ.

In outline, the proof of the Lifting Lemma is a standard construction, but there are subtleties in the use of normalized substitutions. We need to know that CR-normal forms are wR-irreducible, unique in their CR-equivalence classes, and closed under subterm extraction (in fact, *any* class of terms with these properties would suffice for our purposes). All but the last assertion can be derived from classical facts about normal forms on $\mathcal{LC}$ and $\mathcal{CL}$. Closure under subterm formation, however, seems to require a complete reconstruction of the classical theory of *strong reduction* ([CF58], [Hin67], [Ler67]) in the presence of R-reduction to establish a generalization of Curry's normal form theorem and its converse, namely that the classes of CR-normal forms and terms which are R-strongly irreducible are the same. The class of CR-normal forms as defined above is shown in [Joh92] to possess the properties we require.

In the proof of the Lifting Lemma we will require the facts that i) if X is in CR-normal form and Y and Z are subterms of X such that $Y =_{CR} Z$, then $Y \equiv Z$, and ii) if X is in CR-normal form, then X is not the redex-term for any validity preserving R^L-REDUCE step.

Lemma 4.4 (Lifting Lemma) *Let $\theta \in NCRU(\Gamma)$ and let $\langle X, Y \rangle$ be an unsolved pair in Γ. If*

$$\theta\Gamma \longrightarrow \Pi$$

is a validity preserving $R^L\vee\mathrm{T}$-step out of $\theta\langle X, Y \rangle$, then there exists a Π' and θ' with

$$\Gamma \Longrightarrow \Pi'$$

such that

1. $\theta' \equiv \theta[Supp(\Gamma)]$,
2. $\theta'\Pi' \cong \Pi$, *and*
3. $\theta' \in NCRU(\Pi')$.

Proof. Write Γ as $\Gamma'; \langle X, Y \rangle$; since $\langle X, Y \rangle$ is not solved, θ is CR-normal on the variables of X and Y.

In case Π is obtained by WEAK REDUCE, we have

$$\theta\Gamma \equiv \theta\Gamma'; \theta\langle X, Y \rangle \longrightarrow \theta\Gamma', \langle (\theta X)', \theta Y \rangle \equiv \Pi$$

where $(\theta X)' \equiv (\theta X)[u \leftarrow \delta T]$ for a combinatory weak reduction rule $S \to T$ with fresh variables and matching substitution δ such that $D\delta \subseteq Vars(S)$. In addition, u is a non-variable occurrence of X since θ is pointwise weakly normal on the variables of X, and so $(\theta X)/u \equiv \theta(X/u)$. Letting σ be a most general unifier of X/u and S, and putting $X' \equiv X[u \leftarrow T]$, the following is a *Weak Narrow* step:

$$\Gamma \equiv \Gamma'; \langle X, Y \rangle \Longrightarrow [\sigma], \sigma\Gamma', \langle \sigma X', \sigma Y \rangle \equiv \Pi'.$$

Take θ' to be $\theta \cup \delta$. It is easy to see that $\theta' \equiv \theta[Supp(\Gamma)]$ since the variables of S are fresh.

To check that $\theta'\Pi' \cong \Pi$, observe that since θ' unifies X/u and S, $\sigma \leq \theta'$ so that $\theta'[\sigma]$ is trivial and $\theta'\sigma \equiv \theta'$. The verification that $\theta'\sigma X' \equiv (\theta X)'$ is routine.

To verify that $\theta' \in NCRU(\Pi')$, first note that $\theta'\Pi'$ is CR-valid since $\theta'\Pi' \cong \Pi$ and WEAK REDUCE steps preserve CR-validity, so that Π is CR-valid. Since θ is pure and $D\delta \subseteq Vars(S)$, θ' is pure. Now let z be an unsolved variable of Π'; we show that $\theta' z$ is in CR-normal form. Such a z is either a variable from Γ or is introduced by σ. If z is from Γ then z was unsolved there since the WEAK REDUCE step was not done out of a solved pair, and so $\theta' z \equiv \theta z$ is CR-normal. So suppose z is introduced by σ. Then z is a variable in $(\sigma X)/u$, i.e. for some x in X/u, z is in σx. This implies that $\theta' z$ is a subterm of $\theta'\sigma x$. But this latter term is simply θx, which is CR-normal, and subterms of CR-normal terms are CR-normal.

In case Π is obtained by validity preserving R^L-REDUCE, we have

$$\theta\Gamma \equiv \theta\Gamma'; \langle \theta X, \theta Y \rangle \longrightarrow \theta\Gamma', \langle (\theta X)', \theta Y \rangle, \delta\Delta \equiv \Pi$$

where $(\theta X)' \equiv (\theta X)[u \leftarrow \delta T]$ for a rule $S^0 \longrightarrow T$ in R^0 with fresh variables and witnessing system $\delta\Delta$ where $D\delta \subseteq Vars(S^0)$.

Since θ is pointwise CR-normal on the variables of X, the remark immediately preceding this lemma insures that u is a non-variable occurrence of X, and, in fact, $\theta(X/u) \equiv \delta S^0$. Letting σ be a most general unifier of X/u and S^0 and putting $X' \equiv X[u \leftarrow T]$, the following is an R^L-*Narrow* step:

$$\Gamma \equiv \Gamma'; \langle X, Y\rangle \implies [\sigma], \sigma\Gamma', \langle \sigma X', \sigma Y\rangle, \sigma\Delta \equiv \Pi'.$$

Take θ' to be $\theta \cup \delta$. It is easy to see that $\theta' \equiv \theta[Supp(\Gamma)]$ since the variables of S^0 are fresh.

To check that $\theta'\Pi' \cong \Pi$, observe that since θ' unifies X/u and S^0, $\sigma \leq \theta'$ so that $\theta'[\sigma]$ is trivial and $\theta'\sigma \equiv \theta'$. Also $\theta'\sigma\Delta \equiv \theta'\Delta \equiv \delta\Delta$. That $\theta'\sigma X' \equiv (\theta X)'$ is a routine calculation.

To verify that $\theta' \in NCRU(\Pi')$, first note that $\theta'\Pi'$ is CR-valid since $\theta'\Pi' \cong \Pi$ and we are assuming that the R^L-REDUCE step in question is validity preserving. Since θ is pure and $D\delta \subseteq Vars(S^0)$, θ' is pure. The proof that θ' is CR-normal on the unsolved variables of Γ exactly as in the case the R^LVT-step is WEAK REDUCE.

In case Π is obtained by ADD ARGUMENT, we have (letting the type of θX be $(\alpha \to \beta)$):

$$\theta\Gamma \equiv \theta\Gamma'; \langle \theta X, \theta Y\rangle \longrightarrow \theta\Gamma', \langle (\theta X)d, (\theta Y)d\rangle \equiv \Pi;$$

let the type of X be τ_1 and the type of Y be τ_2, and in case these are each atomic types, let $(s \to t)$ be the type introduced as in the definition of *Add Argument*. An application of *Add Argument* yields Π':

$$\Gamma \equiv \Gamma'; \langle X, Y\rangle \implies [\sigma], \sigma\Gamma', \langle (\sigma X)e, (\sigma Y)e\rangle \equiv \Pi';$$

Choose θ' to be $\theta \cup \delta$, where $\delta_0 \equiv \{s := \alpha, t := \beta\}$ and δ_1 is the identity. Clearly $\theta' \equiv \theta[Supp(\Gamma)]$.

To verify that $\theta'\Pi' \cong \Pi$, first observe that θ'_0 unifies τ_1, τ_2, and $(s \to t)$ since it maps each of these to $(\alpha \to \beta)$. So $\sigma \leq \theta'_0$, and therefore $\theta'[\sigma]$ is trivial. Furthermore, $\theta'\sigma \equiv \theta'$ so that $\theta'\sigma$ and θ agree on Γ', X, and Y. Finally, since $\theta'(\sigma X) \equiv \theta X$ has type $(\alpha \to \beta)$, $\theta' e$ must indeed have type α. Moreover, the fact that θ is pure implies that Γ and $\theta\Gamma$ involve the same *Args* parameters, and so d and e have the same type-erasure.

To see that $\theta' \in NCRU(\Pi')$, first note that $\theta'\Pi'$ is CR-valid as before; then observe that θ' is appropriately CR-normal since the R^LVT-step was not done out of a solved pair, and so new unsolved term-variables appear in Π'. It is clear that θ' is pure.

In case Π is obtained by DECOMPOSE, we have two cases: If θX and θY have the same constant at the head, then X and Y also have these constants at the head, and we may obtain Π' by applying the syntactic unification transformation Decompose to $\langle X, Y\rangle$ and take θ' to be θ. Otherwise, we can describe $\langle X, Y\rangle$ and $\theta\langle X, Y\rangle$ as follows:

$$\langle X, Y\rangle \equiv \langle xX_1...X_k, hZ_1...Z_mY_1...Y_k\rangle$$

where $m, k \geq 0$, $x \in Vars$, h is pure, and

$$\theta\langle X, Y\rangle \equiv \langle aA_1...A_nB_1...B_mL_1...L_k, aA_1...A_nC_1...C_mQ_1...Q_k\rangle,$$

for some $n \geq 0$ with

$$\begin{gathered} aA_1..A_nB_1...B_m \equiv \theta x, \\ L_i \equiv \theta X_i, \\ aA_1...A_n \equiv \theta h, \\ C_i \equiv \theta Z_i, \text{ and} \\ Q_i \equiv \theta Y_i. \end{gathered}$$

The repetition of the A_i is justified by the remark immediately preceding this lemma. The assertion that h cannot be a constant from *Args* follows from the fact that θ is a pure substitution.

We obtain Π' by applying *Split*:

$$\Gamma \equiv \Gamma';\ \langle xX_1...X_k, hZ_1...Z_mY_1...Y_k\rangle \Longrightarrow$$
$$[\sigma], \sigma\Gamma',\ \langle z_1, \sigma Z_1\rangle, ..., \langle z_m, \sigma Z_m\rangle, \langle \sigma X_1, \sigma Y_1\rangle, ..., \langle \sigma X_k, \sigma Y_k\rangle \equiv \Pi'.$$

where σ is a most general unifier of x and $hz_1...z_m$. Take θ' to be $\theta \cup \delta$, where δ_0 is the identity and $\delta_1 \equiv \{z_1 := B_1, ..., z_m := B_m\}$. As before, $\theta' \equiv \theta[Supp(\Gamma)]$.

To check that $\theta'\Pi' \cong \Pi$, we first see that θ' unifies x with $hz_1...z_m$, since applying θ' to each yields $aA_1...A_nB_1...B_m$. So $\theta'\sigma \equiv \theta'$ and the pairs of $\theta'\Pi'$ match the pairs of Π except that the trivial system $\theta[\sigma]$ does not appear in Π and when $n > 0$, Π' will not include pairs corresponding to the pairs $\langle A_i, A_i\rangle$ in Π.

As usual, $\theta'\Pi'$ is CR-valid, and the fact that each B_i is pure and CR-normal guarantees that θ' is pure and CR-normal. Thus $\theta' \in NCRU(\Pi')$. □

4.2 Refinements

Before proving that the R^LUT transformations can enumerate a complete set of CR-unifiers for convergent R and arbitrary systems of $\mathcal{CL}$-terms, we indicate a way to decrease the non-determinism inherent in our CR-unification method.

Definition 4.5 A system is *semi-simple* if it is irreducible with respect to WEAK REDUCE, ADD ARGUMENT, and R-REDUCE.

We will see that it suffices to apply R^LUT-steps only to semi-simple systems. Reducing the system obtained after every R^LUT-step to a semi-simple system clearly yields a CR-unification procedure with a smaller search space than would be otherwise obtained; reducing a system to a semi-simple one recalls the SIMPL phase of Huet's classical higher-order unification algorithm. In the present setting, we may view such a reduction as the normalization phase of a normalized narrowing algorithm ([Ret87]).

With this in mind, we isolate the following corollary to Theorem 2.5:

Lemma 4.6 *Any sequence of* WEAK REDUCE, ADD ARGUMENT, *and* R-REDUCE *steps applied to a system will terminate in a semi-simple system with the same CR-unifiers.*

Transformation-based unification methods attempt to reduce a system to be unified to a solved form which essentially represents a unifier. For the purposes of unification modulo a theory, however, it is not sufficient to have transformed a system into a unifiable one since some semantic unifiers may be more general than a most general syntactic unifier. For example, $\langle Kax, Kay\rangle$ has the identity substitution as a semantic unifier but not as a syntactic unifier.

On the other hand, if Γ is a *solved* system, then for any R, the substitution associated with Γ is more general than any CR-unifier. The following lemma will allow us to ignore the solved part of a system when searching for CR-unifiers; indeed, we will not apply any transformation out of a solved pair. This restriction is consistent with the intuition that the solved part of a system is merely a record of an answer substitution being constructed.

Lemma 4.7 *Let Γ be syntactically unifiable. If θ is a CR-unifier of Γ and a syntactic unifier of the unsolved part of Γ, then $mgu(\Gamma) \leq_{CR} \theta$.*

Proof. Let the solved and unsolved parts of Γ be $[\sigma]$ and Π respectively. We first claim that if γ is $mgu(\Pi)$, then $mgu(\Gamma)$ is $\gamma\sigma$. Certainly $\gamma\sigma[\sigma]$ is trivial and the fact that $\gamma\sigma\Pi$ is trivial follows from the fact that σ is the identity on Π, so that $\gamma\sigma$ is a unifier. To see that it is most general, let δ be any unifier of $[\sigma], \Pi$. It suffices to show that $\delta\gamma\sigma \equiv \delta$. But since γ is idempotent and $\gamma \leq \delta$, we have $\delta\gamma \equiv \delta$. Similarly $\delta\sigma \equiv \delta$ since δ unifies $[\sigma]$.

Since θ unifies Π, $\gamma \leq \theta$ and so $\gamma\sigma \leq \theta\sigma$. But since θ E-unifies $[\sigma]$ and σ is idempotent, $\theta\sigma =_{CR} \theta$. □

4.3 The Algorithm and Its Completeness

Definition 4.8 The non-deterministic algorithm $\mathcal{RU}$ is the following process:
Repeatedly:

1. Reduce the system to a semi-simple system via RVT-steps and then apply some R^LUT-step to a non-trivial unsolved pair.
2. If at any point the system is syntactically unifiable by a pure substitution then return a most general unifier of the system without transforming the system (but do not necessarily halt).

Observe that the reduction of a system to a semi-simple system uses ordinary RVT-steps, which are guaranteed to preserve unifiers. Indeed, the R^L-REDUCE transformations never appear in the description of the algorithm $\mathcal{RU}$, emphasizing the fact that they exist only as a theoretical underpinning for the R^LUT transformations, and so would never appear in an implementation of $\mathcal{RU}$.

Algorithm $\mathcal{RU}$ generates a stream of CR-unifiers for a given input system, although if we are to avoid losing unifiers we cannot necessarily halt computation after arriving at a syntactically unifiable system. The next example illustrates this phenomenon.

Example 4.9 Consider the semi-simple system $\Gamma \equiv \langle uax, uay \rangle$. If every run of the algorithm $\mathcal{RU}$ were eagerly to return a most general unifier of a syntactically unifiable system, then no $\mathcal{RU}$-computation out of Γ would return a unifier of Γ more general than the substitution $\{u \mapsto K\}$, which is a CR-unifier of Γ for any R. Algorithm $\mathcal{RU}$ would therefore not be complete.

It follows from Lemmas 4.2 and 4.6 that if Algorithm $\mathcal{RU}$ is run on an initial system Γ and returns a substitution θ, then θ is a CR-unifier of Γ. The main result of this paper is a converse.

Example 4.10 Let R be consist of the equation $f(gz) \longrightarrow a$, and consider the semi-simple CR-unifiable pair $\langle f((ugx)y), a \rangle$. Computing with Algorithm $\mathcal{RU}$ gives

$$\langle f((ugx)y), a \rangle \Longrightarrow \langle u, K \rangle, \langle v, gx \rangle, \langle w, y \rangle, \langle f(gy), a \rangle \Longrightarrow \langle u, K \rangle, \langle a, a \rangle, \langle v, gx \rangle, \langle w, y \rangle,$$

where in the first step ugx is unified with the left-hand side of a rule $Kvw \to v$, and in the second the rewriting of $f(gy)$ is performed. The substitution θ, where θ_0 is the identity and $\theta_1 = \{u \mapsto K\}$, is thus a CR-unifier of the original pair.

Example 4.11 Let R consist of the rule $fzz \longrightarrow a$ and let Γ be the system

$$\langle f(ux)(u(SKx)), a\rangle\,,$$

where u has type $(r \to r) \to 0$ and x has type $r \to r$. One application of R^L-*Narrow* yields

$$\langle y, ux\rangle\,, \langle z, u(SKx)\rangle\; \langle ux, u(SKx)\rangle\,,$$

(here and below we suppress writing trivial pairs). The syntactic unification transformation Decompose gives

$$\langle y, ux\rangle\,, \langle z, u(SKx)\rangle\,, \langle x, (SKx)\rangle\,.$$

After an application of *Add Argument* we have

$$\langle y, ux\rangle\,, \langle z, u(SKx)\rangle\,, \langle xp, (SKx)p\rangle\,.$$

Two WEAK REDUCE steps further give

$$\langle y, ux\rangle\,, \langle z, u(SKx)\rangle\,, \langle xp, p\rangle\,,$$

and one *Weak Narrow* using $Sz_1z_2z_3 \to z_1z_3(z_2z_3)$ and pair $\langle xp, p\rangle$ yields

$$\langle y, u(Sz_1z_2)\rangle\,, \langle z, u(SK(Sz_1z_2))\rangle\,, \langle x, Sz_1z_2\rangle\,, \langle z_1p(z_2p), p\rangle\,, \langle z_3, p\rangle\,.$$

Another *Weak Narrow*, this time employing $Kw_1w_2 \to w_1$ and pair $\langle z_1p(z_2p), p\rangle$ leaves us with

$$\langle y, u(SKz_2)\rangle\,, \langle z, u(SK(SKz_2))\rangle\,, \langle x, SKz_2\rangle\,, \langle z_1, K\rangle\,, \langle w_1, p\rangle\,, \langle w_2, z_2p\rangle\,, \langle z_3, p\rangle\,.$$

This system is solved, so we extract the CR-unifier θ where θ_0 is the identity and $\theta_1 = \{x \mapsto SKz_2\}$.

Theorem 4.12 (Completeness) *Let R be convergent, and let θ be a pure CR-unifier of Γ. Then there is a computation of Algorithm $\mathcal{RU}$ on Γ producing a pure CR-unifier σ of Γ with $\sigma \leq_{CR} \theta[Supp(\Gamma)]$.*

Proof. Since every pure CR-unifier of Γ is pointwise CR-equal to a normalized CR-unifier of Γ, we may prove the theorem under the additional hypothesis that $\theta \in NCRU(\Gamma)$.

Let the *degree* of a system be the maximum length of an R^LVT-sequence out of it. The proof is by induction on the degree of $\theta\Gamma$.

If θ is a unifier of the unsolved part of Γ, then Γ is unifiable and Algorithm $\mathcal{RU}$ can return a most general unifier σ. By Lemma 4.7, $\sigma \leq_{CR} \theta$. This situation obtains if the degree of $\theta\Gamma$ is 0.

Otherwise, we define a system Π' and a substitution θ' as follows:

1. If Γ is not semi-simple, apply an R^LVT-step to obtain Π' and let θ' be θ.
2. Otherwise, choose an unsolved pair $\langle X, Y\rangle$ from Γ so that $\theta X \not\equiv \theta Y$ and perform a validity preserving R^LVT-step out of $\langle \theta X, \theta Y\rangle$ yielding Π (such a step is possible by part (2) of Theorem 2.10). The Lifting Lemma applies, yielding Π' and θ'.

In each case, the action performed is an $\mathcal{RU}$-step, $\theta' \in NCRU(\Pi')$ and the degree of $\theta'\Pi'$ is less than the degree of $\theta\Gamma$ (using the facts that $\theta'\Pi' \cong \Pi$ and that no R^LVT-steps are done out of trivial pairs).

By induction, there is a computation of Algorithm $\mathcal{RU}$ on Π' producing a CR-unifier σ of Π' with $\sigma \leq_{CR} \theta'[Supp(\Pi')]$. By soundness, σ is a CR-unifier of Γ. Since $Supp(\Gamma) \subseteq Supp(\Pi')$, $\sigma \leq_{CR} \theta'[Supp(\Gamma)]$. But since $\theta' \equiv \theta[Supp(\Gamma)]$, $\sigma \leq_{CR} \theta[Supp(\Gamma)]$ as desired. □

It is important to observe that weak termination of validity preserving R^LVT-steps is all that is required to show CR-unification completeness of R^LUT.

5 Conclusion

We have defended the claim that the formulation of higher-order logic using combinators allows routine algebraic techniques to be applied to higher-order unification and leads to a smooth interaction between these methods and the standard tools of first-order unification.

The ordinary higher-order unification problem submits to a narrowing algorithm in the combinator setting. For higher-order E-unification in the situation that E itself allows narrowing, the combined problem is solved by combining these algorithms (in a naive way for the left-linear case, and by passing to conditional rewriting otherwise). The result is a complete enumeration procedure with a finitely-branching search space; we expect that this technique will yield an alternative approach to that in [Sny90] to higher-order E-unification under arbitrary E.

In light of the theoretical intractability of general higher-order unification and the practical utility of *pre-unification*, a crucial line of investigation is now an analysis and implementation of higher-order E-preunification in the combinator framework, and the development of tools to control the redundancy which is inherent (as shown by Huet) in any complete enumeration method for higher-order unification. It will also be important to focus on particular standard equational theories such as associativity and commutativity; our conjecture is that the known algorithms for such situations can be readily adapted to the algebraic higher-order context.

References

[Bar90] F. Barbanera. Adding algebraic rewriting to the Calculus of Constructions: strong normalization preserved. *Extended Abstracts, The Second International Workshop on Conditional and Typed Rewriting Systems*, Center for Pattern Recognition and Machine Intelligence, 1990.

[BG91a] V. Breazu-Tannen and J. Gallier. Polymorphic rewriting conserves algebraic strong normalization. To appear, *Theoretical Computer Science.*

[BG91b] V. Breazu-Tannen and J. Gallier. Polymorphic rewriting conserves algebraic confluence. To appear, *Information and Computation.*

[BK86] J. A. Bergstra and J. W. Klop. Conditional rewrite rules: confluence and termination. *Journal of Computer and System Sciences* 32, pp. 322–362, 1986.

[Bou90] A. Boudet. Unification in a combination of equational theories: an efficient algorithm. *Proceedings of the Tenth Conference on Automated Deduction*, Springer-Verlag LNAI 449, pp. 292–307, 1990.

[Bre88] V. Breazu-Tannen. Combining algebra and higher-order types. *Proceedings of the Third Annual IEEE Symposium on Logic in Computer Science*, IEEE Press, pp. 82–90, 1988.

[CF58] H. B. Curry, R. Feys. *Combinatory Logic, Vol. I*, North-Holland, 1958.

[Der87] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation* 3, pp. 69–116, 1987.

[DJ90] N. Dershowitz, J.-P. Jouannaud. Rewrite systems. *Handbook of Theoretical Computer Science B: Formal Methods and Semantics*, J. van Leeuwen, ed., North-Holland, pp. 243–320, 1990.

[Dou92] D. J. Dougherty. Adding algebra to the untyped lambda calculus. *Proceedings, Fourth International Conference on Rewriting Techniques and Applications*, Springer-Verlag LNCS 488, pp. 37–48, 1991. Also in *Information and Computation* 101-2, pp. 251–267, 1992.

[Dou93] D. J. Dougherty. Higher-order unification via combinators. *Theoretical Computer Science* B 114, pp. 273–298, 1993.

[Ell89] C. Elliott. Higher-order unification with dependent function types. *Proceedings of the Third International Conference on Rewriting Techniques and Applications*, Springer-Verlag LNCS 355, pp. 121–136, 1989.

[Ell90] C. Elliott. *Extensions and applications of higher-order unification.* Dissertation, Carnegie-Mellon University, 1990. Avaliable as Technical Report CMU-CS-90-134.

[EP89] C. Elliott and F. Pfenning. eLP: A common Lisp implementation of λ-Prolog in the Ergo support system. Available by ftp from *elp-request@cs.cmu.edu*, 1989.

[Fay79] M. Fay. First order unification in an equational theory. *Proceedings of the Fourth Workshop on Automated Deduction*, 1979.

[GS89a] J. H. Gallier and W. Snyder. Complete sets of transformations for general E-unification. *Theoretical Computer Science* 67, pp. 203–260, 1989.

[GS89b] J. H. Gallier and W. Snyder. Higher-order unification revisited: complete sets of transformations. *Journal of Symbolic Computation* 8, pp. 101–140, 1989.

[Hin67] R. Hindley. Axioms for strong reduction in combinatory logic. *Journal of Symbolic Computation* 32, pp. 224–236, 1967.

[HS86] J. R. Hindley and J. P. Seldin. *Introduction to Combinators and λ-Calculus*, Cambridge University Press, 1986.

[Hue75] G. Huet. A unification algorithm for typed λ-calculus. *Theoretical Computer Science* 1, pp. 27–57, 1975.

[Hus91] U. Hustadt. A complete transformation system for polymorphic higher-order unification. Technical Report MPI-I-91-228, Max-Planck-Institut für Informatik, Saarbrücken, 1991.

[JK91] J.-P. Jouannaud and C. Kirchner. Solving equations in abstract algebras: a rule-based study of unification. *Computational Logic: Essays in Honour of Alan Robinson*, ed. J. Lassez and G. Plotkin, MIT Press, Cambridge, pp. 257–321, 1991.

[JO91] J.-P. Jouannaud and M. Okada. A computation model for executable higher-order algebraic specification languages. *Proceedings of the Sixth Annual IEEE Symposium on Logic in Computer Science*, IEEE Press, pp. 350–361, 1991.

[Joh91] P. Johann. *Complete Sets of Transformations for Unification Problems.* Dissertation, Wesleyan University, 1991.

[Joh92] P. Johann. *Normal Forms in Combinatory Logic.* Technical Report, Wesleyan University, 1992. Submitted, *Notre Dame Journal of Formal Logic.*

[Klo80] J. W. Klop. *Combinatory Reduction Systems.* Mathematical Center Tracts 129, Amsterdam, 1980.

[Ler67] B. Lercher. Strong reduction and normal form in combinatory logic. *Journal of Symbolic Logic* 2, pp. 213–223, 1967.

[Mit90] J. Mitchell. Type systems for programming languages. *Handbook of Theoretical Computer Science, Volume B*, ed. J. van Leeuwen, MIT Press/Elsevier, pp.365–458, 1990.

[MM82] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems* 4, pp. 258–282, 1982.

[Nip90] T. Nipkow. Higher-order unification, polymorphism, and subsorts. *Extended Abstracts, The Second International Workshop on Conditional and Typed Rewriting Systems*, Center for Pattern Recognition and Machine Intelligence, 1990.

[NPS??] P. Narendran, F. Pfenning. and R. Statman. On the unification problem for cartesian closed categories.

[NQ91] T. Nipkow and Z. Qian. Modular Higher-order E-unification. *Proceedings, Fourth International Conference on Rewriting Techniques and Applications*, Springer-Verlag LNCS 488, pp. 200–214, 1991.

[Oka89] M. Okada. Strong normalizability for the combined system of the typed lambda calculus and an arbitrary convergent rewrite system. *Proceedings, ISSAC 89*, 1989.

[PE88] F. Pfenning and C. Elliott. Higher-order abstract syntax. In *Proceedings of SIGPLAN88 Symposium on Language Design and Implementation*, ACM Press, pp. 199–208, 1988.

[Pey87] S. L. Peyton-Jones. *The Implementation of Functional Programming Languages.* Prentice-Hall, 1987.

[Ret87] P. Réty. Improving basic narrowing techniques. *Proceedings of the Second International Conference on Rewriting Techniques and Applications*, 1987.

[Sch89] M. Schmidt-Schauss. Unification in a combination of arbitrary disjoint equational theories. *Journal of Symbolic Computation* 8, pp. 51–99, 1989.

[Sie89] J. Siekmann. Unification theory. *Journal of Symbolic Computation* 7, pp. 207–274, 1989.

[Sny90] W. Snyder. Higher-order E-unification. *Proceedings of the Tenth Conference on Automated Deduction*, Springer-Verlag LNAI 449, pp. 292–307, 1990.

[Vri87] R. C. deVrijer. *Surjective Pairing and Strong Normalization: Two Themes in Lambda Calculus.* Dissertation, Universiteit van Amsterdam, 1987.

[Vri90] R. C. deVrijer. *Unique Normal Forms for Combinatory Logic with Parallel Conditional, a Case Study in Conditional Rewriting.* Internal Report IR-240, Vrije Universiteit, Amsterdam, 1990.