FORMAL SYNTHESIS OF A UNIFICATION ALGORITHM 
BY THE DEDUCTIVE-TABLEAU METHOD*

DANIELE NARDI

We present the formal derivation of a unification algorithm using the 
deductive-tableau method for program synthesis. The methodology is briefly 
described, with emphasis on the deduction rules used in the derivation. 
Starting from an input-output specification expressed in first-order logic, a 
unification algorithm is synthesized by proving the validity of the specifica-
tion. The termination of the synthesized program is also proved.

1. INTRODUCTION

In this paper we present a formal synthesis of a unification algorithm using the 
deductive-tableau method. Unification was first introduced in theorem proving by 
Robinson [10], and the problem is well known in the literature (see for example 
Siekmann [11]). The deductive-tableau method is a methodology for program 
synthesis developed by Manna and Waldinger [5]. It allows one to synthesize a 
program by proving the validity of a formula of the first-order predicate calculus 
expressing the input-output behavior of the desired program. The deductive-tableau 
method has been used to derive nontrivial programs such as a binary search 
program [7] and sorting programs [13]. Several motivations inspired our synthesis of 
a unification algorithm:

unification has already been used as a testbed for other automatic systems for 
program synthesis and program verification;

the derivation is rather complicated and provides a framework for the discussion 
of the features of the method; in particular it shows that the introduction of

*The work presented in this paper has been partly done while the author was visiting the Artificial 
Intelligence Center of SRI International, with a grant of the Italian CNR, and partly at the University of 
Rome, under a contract with Enidata.

Address correspondence to D. Nardi, Dipartimento di Informatica e Sistemistica, Università di Roma 
“La Sapienza,” Via Buonarroti 12, I-00185 Roma, Italy.

Received January 1987; accepted February 1988.
new, powerful deduction rules [8] becomes essential in order to keep the size of
the proof manageable;

the theory of unification is well established, and this work will serve as the basis
for further work in the automatic derivation of other unification algorithms.

The synthesis presented here follows the one developed by Manna and Waldinger
[6]. Their derivation was not expressed in a formal system: a first attempt to provide
a formal derivation has been presented by Eriksson [1], within the framework of the
logic-programming calculus. A comparison with Eriksson’s derivation will show
some of the advantages of the use of the deductive tableaus.

In the next section we sketch the essential features of the deductive tableaus;
then we present the specification and the derivation of a unification algorithm. The
presentation is structured according to the branches of the derivation that corre-
spond to the syntactic categories of the definition of expressions. The proof of the
termination of the algorithm, which is part of the synthesis, is included as well.
Finally, we discuss the related work and make some remarks on the use of deductive
tableaus.

2. DEDUCTIVE TABLEAUS

In this section we give a short and informal presentation of the deductive-tableau
method; in particular we focus on the aspects that are relevant to our derivation of
the unification algorithm. The reader is referred to the papers by Manna and
Waldinger [5, 8] for a complete account of the method.

The deductive-tableau method is a formal apparatus for program synthesis. The
approach is based on theorem proving: starting from a specification, a program is
obtained by proving the validity of the specification. The program thus derived is
guaranteed to be correct with respect to the initial specification.

A specification has the form

\[ f(a) \leftarrow \text{find } z \text{ such that } R[a, z] \text{ where } P[a]. \]

\( R[a, z] \) specifies the output condition, and \( P[a] \) specifies the input condition. \( f(a) \)
is the program obtained at the end of the derivation; it is expressed as a set of
mutually recursive expressions written in a side-effect free applicative language.

The input-output behavior of the target program is described by the following
sentence in the language of first-order logic:¹

\[ (\forall a)[\text{if } P[a] \text{ then } (\exists z)[R[a, z]]]. \]

The proof of the validity of the input-output condition leads to the synthesis of the
program. The tableau associated with a specification is represented by

<table>
<thead>
<tr>
<th>Assertions</th>
<th>Goals</th>
<th>( f(a) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( an P(a) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( on )</td>
<td>( R[a, z] )</td>
<td>( z )</td>
</tr>
</tbody>
</table>

¹Following the notation adopted in the framework of the deductive tableaus, we use the if-then form
to denote implication.
An and Gn are the labels that we use to refer to a particular row of the tableau. The truth of the tableau under a given interpretation is expressed by the sentence:

If all the instances of each assertion arc true, then some instance of at least one of the goals is true.

The distinction between assertions and goals reflects the usual distinction between hypothesis and thesis; any goal can be moved into the assertion column by simply negating it (and vice versa), without affecting the meaning of the tableau.

The output (usually associated with the goals) is said to be **suitable** if it satisfies the input-output condition whenever the corresponding instance of the goal is true.

A deduction consists of adding new rows to the tableau without changing its meaning; that is to say, if the old tableau is true under a certain interpretation, then the new tableau is true under the same interpretation, and if the output of the old tableau is suitable, then the output of the new tableau is suitable as well. The deduction terminates when either a true goal or a false assertion is derived: the associated output contains the desired program.

2.1. The Deduction Rules

The deduction is based on nonclausal resolution, which has been proved complete [4], combined with well-founded induction, which is necessary to obtain the recursive calls in the programs. Other rules [8,12] can be used in order to improve the deduction process: they generally allow for more compact and readable proofs. In particular, in the derivation of the unification algorithm, we apply the equality-replacement rule and the resolution-with-equality-matching rule.

We assume that the unification algorithm used to perform the deduction is associative-commutative, so that we do not have to consider these properties explicitly.

2.1.1. Splitting Rules. According to the meaning of the tableau, a disjunctive goal can be split on several rows, each with the same output of the original goal (\(\lor\)-rule). A dual \(\land\)-rule exists for assertions.

2.1.2. Transformation Rules. Transformation rules allow for the replacement of equivalent or equal subexpressions within assertions, goals, and outputs. In particular, the simplifications of propositional-logic formulas are expressed as transformation rules (e.g. \(P \land false \equiv false\)).

2.1.3. The Resolution Rule. The resolution rule allows to derive a new line of a tableau if it is possible to unify two subsentences of two rows of the tableau. In the ground case the rule is expressed by the following tableau:

<table>
<thead>
<tr>
<th>Assertions</th>
<th>Goals</th>
<th>(f(a))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F[P])</td>
<td></td>
<td>(s)</td>
</tr>
<tr>
<td>(G[P])</td>
<td></td>
<td>(t)</td>
</tr>
<tr>
<td>(F[true]\land G[false])</td>
<td></td>
<td>if (P) then (s) else (t)</td>
</tr>
</tbody>
</table>
If a subsentence $P$ occurs both in the goal $F$ and in the goal $G$, then by applying the resolution rule a new goal is derived as the conjunction of the initial goals, where the subsentence $P$ has been substituted by $true$ in one formula and $false$ in the other one. This rule represents a generalization of the standard resolution step on ground terms, which allows one to collapse two clauses containing $P$ and $\neg P$ respectively. The output of the new goal contains a conditional term, whose test-part is the subsentence $P$, then-part corresponds to the case of a $true$ $P$, and else-part to the case of a $false$ $P$.

The application of the resolution rule usually leads to sentences that can be simplified using the transformation rules. This step will generally be omitted, by presenting the already simplified sentence. The use of the polarity strategy [5] guarantees that the resolution step is applied successfully, namely that it is possible to simplify the result of the resolution.

The resolution rule works in the above form when it is applied to goals. According to the duality between assertions and goals, analogous rules are defined for the resolution between two assertions and between a goal and an assertion.

The resolution rule is generalized to the nonground case by applying it when two subsentences unify: the result is then obtained as in the ground case, but the most general unifier is applied to the sentences as well as to the output. We use boxes in the tableaus to surround the expressions that are unified at each step. In this way, the unifiers of the steps of the proof should be easily understandable from the context, and therefore they are omitted most of the time.

2.1.4. The Induction Rule. The induction rule consists of the introduction of an induction hypothesis and makes it possible to obtain a recursive call in the final program. If the induction rule is applied to the initial tableau, it allows for the introduction of the following assertion:

\[
\text{if } x >_{\omega} a \text{ then if } P[x] \text{ then } R[x, f(x)]
\]

This assertion, on the assumption that a well-founded relation $<_{\omega}$ exists, states that the program $f$ applied to the input $x$ will satisfy the input-output condition, provided that $x <_{\omega} a$. If the resolution rule can be applied between the induction hypothesis and a derived goal by unifying a subexpression containing $f(x)$, the resulting goal has a recursive call in the output. In such a goal, a conjunct involving the $<_{\omega}$ relation occurs, which represents the termination condition for the program and can be specified later in the proof by choosing an appropriate well-founded relation. A more general version of this rule allows for the synthesis of auxiliary procedures, but we skip it, since it is not used in our derivation.

2.1.5. The Equality-Replacement Rule. The equality-replacement rule is an extension of resolution that takes into account equality; in the ground case it is expressed by the following tableau:

<table>
<thead>
<tr>
<th>Assertions</th>
<th>Goals</th>
<th>$f(a)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F[t_1 = t_2]$</td>
<td>$s$</td>
<td></td>
</tr>
<tr>
<td>$G\langle t_1 \rangle$</td>
<td>$t$</td>
<td></td>
</tr>
<tr>
<td>$F[false] \wedge G\langle t_2 \rangle$</td>
<td>if $t_1 = t_2$ then $t$ else $s$</td>
<td></td>
</tr>
</tbody>
</table>
The rule applies when a term \( t_1 \) occurs in a goal as a member of an equality \( t_1 = t_2 \) and in another goal of the tableau. The result is the conjunction of the two goals: in the first one the equality \( t_1 = t_2 \) is substituted by \( \text{false} \), and in the other one some of the occurrences of the term \( t_1 \) are replaced by \( t_2 \). Also in the case of equality replacement, analogous rules can be formulated for the other combinations of goals and assertions, it can be generalized to the case of unifying terms, and a polarity strategy regulates its application. The rule, known as paramodulation [14], has been generalized to arbitrary binary relations [8].

2.1.6. The Equality-Matching Rule. The equality-matching rule allows us to proceed with the derivation in cases when unification fails. We present its formulation in the ground case for a pair of goals; all the extensions are definable as for the other rules:

<table>
<thead>
<tr>
<th>Assertions</th>
<th>Goals</th>
<th>( f(a) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F[P(t_1)] )</td>
<td></td>
<td>( s )</td>
</tr>
<tr>
<td>( G[P(t_2)] )</td>
<td></td>
<td>( t )</td>
</tr>
<tr>
<td>( t_1 = t_2 \land F[\text{true}] \land G[\text{false}] )</td>
<td>if ( P(t_1) ) then ( s ) else ( t )</td>
<td></td>
</tr>
</tbody>
</table>

The equality-matching rule applies when \( P(t_1) \) and \( P(t_2) \) are identical except for some occurrences of \( t_1 \) that are replaced by \( t_2 \) in \( P(t_2) \); then they can be unified, provided that \( t_1 = t_2 \). The rule is known as \( E \)-resolution [3], and it has been extended to the case of arbitrary binary relations [8].

3. BASIC THEORIES AND CONVENTIONS

Before introducing the basic theories, a few remarks about the notation used throughout the paper are in order.

We use lowercase letters to denote variables, preferably choosing the letter according to the name of the objects of the domain; e.g., \( e, e_1, \ldots \) denote variables ranging over expressions. We also have variables that range over generic objects to express general properties, such as those of the equality relation; these are denoted by \( w, w_1, \ldots \).

The symbols we use for constants are the same as the ones we use for variables, but overbarred. For example, \( \sigma \) is a variable that ranges over substitutions, while \( \bar{\sigma} \) is a constant denoting a substitution. The symbols denoting skolem functions are thus overbarred, and the arguments of the skolem function are explicitly mentioned. The only symbols denoting constants that are not overbarred are \( \text{nil} \) and \( \{ \} \), denoting the failure and the empty substitution, respectively; also, the input expressions are denoted by \( s_1 \) and \( s_2 \), and are not overbarred.

The theory of substitutions that we adopt is taken from Manna and Waldinger [6], except for the use of a simpler data structure for expressions, borrowed from Paulson [9].
3.1. Expressions

Expressions are inductively defined as follows:

- Constants are expressions.
- Variables are expressions.
- If \( e_1, e_2 \) are expressions, their combination \( e_1 \circ e_2 \) is an expression.

Function application is then expressed as a combination; for example \( f(a) \) is \( f \circ a \) and \( f(a, b) \) is \( (f \circ a) \circ b \). We use \( h \) and \( t \) as selectors for the head and the tail of a combination, with the obvious meaning:

\[
\begin{align*}
  h(e_1 \circ e_2) &= e_1, \\
  t(e_1 \circ e_2) &= e_2.
\end{align*}
\]

The following predicates assert that an object belongs to a specific syntactic category:

- \( E(e) \) is true iff \( e \) is an expression;
- \( B(b) \) is true iff \( b \) is a combination;
- \( V(v) \) is true iff \( v \) is a variable;
- \( C(c) \) is true iff \( c \) is a constant.

Besides \( e, e_1, \ldots, r, r_1, \ldots \), also \( r, r_1, \ldots \) are used to denote variables ranging over expressions; in order to keep the size of the formulas reasonable, we sometimes implicitly assume that \( b, b_1, \ldots \) denote combinations, \( v, v_1, \ldots \) denote variable expressions, and \( c, c_1, \ldots \) denote constant expressions.

The properties of expressions used in the proof are given below, in the form of deductive-tableau assertions. They are numbered according to the order in which they appear in the text:

| A1 | if \( E(e) \) then \( C(e) \lor V(e) \lor B(e) \) |
| A2 | if \( B(e_1) \) then \( \neg C(e_1) \land \neg V(e_1) \) |
| A3 | if \( B(b) \land C(c) \) then \( b \neq c \) |

The selection on the combination is handled by the following assertions.

| A4 | if \( B(b) \) then \( b = h(b) \circ t(b) \) |
| A5 | if \( h(b_1) = h(b_2) \land t(b_1) = t(b_2) \) then \( b_1 = b_2 \) |

Finally, the proper occurrence relation is denoted by the symbol \( \ll \). Following Manna and Waldinger, occurrence holds when either two expressions are equal, or proper occurrence holds (see [6, p. 11]). We notice that proper occurrence is irreflexive and does not hold when two expressions are equal; this is expressed by assertion A9:

| A6 | \( h(b) \ll b \) |
| A7 | \( t(b) \ll b \) |
| A8 | \( v_1 \ll v_2 \) |
| A9 | if \( e_1 \ll e_2 \) then \( e_1 \neq e_2 \) |
3.2. Substitutions

The definition of substitutions directly follows from the one used by Manna and Waldinger [6], adapted in terms of the new data structure for expressions.

Substitutions are defined as sets of replacements \( \{ v \leftarrow e \} \), where \( v \) is a variable and \( e \) an expression; \( \{ \} \) denotes the empty substitution.

We use \( \theta, \theta_1, \ldots, \theta_n, \ldots \) and \( \sigma, \sigma_1, \ldots, \sigma_n, \ldots \) to denote substitutions, and \( S \) for the predicate which is true when its argument is a substitution.

We introduce some definitions and properties of the substitutions which are used in the proof. The substitution-application function is denoted by \( \prec \), and is defined by:

\[
\begin{align*}
\text{A10} & : \text{if } C(c) \text{ then } c = c \prec \sigma \\
\text{A11} & : \text{if } V(v) \text{ then } e = v \prec \{ v \leftarrow e \} \\
\text{A12} & : \text{if } V(v_1) \land v_1 \neq e_1 \land v_1 \neq e_1 \text{ then } e_1 = e_1 \prec \{ v_1 \leftarrow e_2 \} \\
\text{A13} & : \text{if } B(b) \text{ then } b \prec a = (h(b) \prec a) \circ (t(b) \prec a)
\end{align*}
\]

The substitution-composition function is denoted by \( \circ \); it is easy to prove the identity of the composition with respect to the empty substitution:

\[
\begin{align*}
\text{A14} & : e \prec (\sigma_1 \circ \sigma_2) = (e \prec \sigma_1) \circ \sigma_2 \\
\text{A15} & : \{ \} \circ \sigma = \sigma
\end{align*}
\]

A useful property of substitutions, called monotonicity (see [6, p. 141]) is also used; it states that the proper occurrence relation is maintained after the substitution application:

\[
\begin{align*}
\text{A16} & : \text{if } e_1 \ll e_2 \text{ then } e_1 \ll \sigma \ll e_2 \ll \sigma
\end{align*}
\]

This property is not proved here, its derivation requires an induction on the data structure of combinations; proper occurrence is either verified by the equality with the head or the tail of the combination, or recursively checked in the head and the tail of the combination.

3.3. Sets of Variables

In order to prove the termination of the algorithm, we introduce a data structure for sets of variables, which are used to reason about the domain and the range of substitutions. Variables ranging over sets are denoted by \( x, x_1, \ldots \).

The symbol \( \text{vs} \) is used to denote the function that returns the set of variable symbols occurring in an expression:

\[
\begin{align*}
\text{A17} & : \text{vs}(h(b)) \subseteq \text{vs}(b) \\
\text{A18} & : \text{vs}(b) = \text{vs}(h(b)) \cup \text{vs}(t(b)) \\
\text{A19} & : \text{if } v \in \text{vs}(h(b)) \lor v \in \text{vs}(t(b)) \text{ then } v \in \text{vs}(b)
\end{align*}
\]

The function symbols \( \text{dom} \) and \( \text{rg} \) denote the functions returning the domain and the range of a substitution, respectively. The domain of a substitution consists of the set of variables containing the left-hand sides of the substitution replacements, while
the range is given by the set of variables occurring in the expressions on the right-hand sides. The properties involving the domain and the range of substitutions are discussed within the termination proof.

3.4. Failure

We use the symbol nil, distinguished from every symbol belonging to other domains, to denote the failure of the unification algorithm; a few useful properties about it are listed below:

<table>
<thead>
<tr>
<th>Assertion</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A20</td>
<td>${} \neq \text{nil}$</td>
</tr>
<tr>
<td>A21</td>
<td>${v \leftarrow e} \neq \text{nil}$</td>
</tr>
<tr>
<td>A22</td>
<td>if $\sigma_1 \circ \sigma_2 = \text{nil}$ then $\sigma_1 = \text{nil} \lor \sigma_2 = \text{nil}$</td>
</tr>
</tbody>
</table>

Assertions A20 and A21 can be derived from the fact that nil is not a substitution.

4. THE INITIAL SPECIFICATION

Unification is the process of finding a common instance of two expressions; when such an instance exists, the two expressions are unifiable and there exists a substitution, called a unifier, which makes the two expressions equal. In symbols, $\theta$ is a unifier of $s_1$ and $s_2$ if $s_1 \triangleleft \theta = s_2 \triangleleft \theta$.

When two expressions are unifiable, more than one unifier can be found, and in order to choose among them the generality of substitutions is introduced. A substitution $\theta$ is more general than $\sigma$, denoted by $\theta \geq \sigma$, if there exists some substitution $\phi$ such that $\sigma = \theta \circ \phi$.

A unification algorithm is usually required to find a most general unifier, that is, a unifier which is more general than any other unifier. In symbols, $\theta$ is a most general unifier of $s_1$ and $s_2$ if $(\forall \sigma)[(s_1 \triangleleft \sigma = s_2 \triangleleft \sigma) \text{ then } \theta \geq \sigma].$

Another important property of substitutions is idempotence; a substitution $\theta$ is idempotent if $\theta = \theta \circ \theta$.

Idempotence allows us to select those unifiers whose domain and range are disjoint; this property is useful in proving the termination of the algorithm, and therefore it is included in the initial specification. The role of idempotence in the algebra of unifiers is discussed by Lassez et al. [2].

The specification of a unification algorithm can be phrased as follows:

find $\theta$ such that $\theta$ is a most general, idempotent unifier of $s_1$ and $s_2$ and $\theta \neq \text{nil}$
or $s_1$ and $s_2$ are not unifiable and $\theta = \text{nil}$

where $s_1$ and $s_2$ are expressions.
Following Manna and Waldinger [6], when there exists a unifier for $s_1$ and $s_2$ the most generality and idempotence conditions can be rephrased as

$$(\forall \sigma)[\text{if } s_1 \sigma = s_2 \sigma \text{ then } \theta = \sigma \sigma].$$

In the tableaus we use $mgid(s_1, s_2, \theta)$ as an abbreviation for this formula. The initial tableau is shown below:

<table>
<thead>
<tr>
<th>Assertions</th>
<th>Goals</th>
<th>$uf(s_1, s_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A23 $E(s_1) \land E(s_2)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G1 $(s_1 \sigma = s_2 \sigma) \land \quad mgid(s_1, s_2, \theta) \land \theta \neq nil \quad \lor (\forall \sigma)[s_1 \sigma = s_2 \sigma] \land \theta = nil$</td>
<td>$\theta$</td>
<td></td>
</tr>
</tbody>
</table>

Assertion A23 specifies that $s_1$ and $s_2$ are expressions; goal G1 specifies either the conditions under which $\theta$ is a most general idempotent unifier of $s_1$ and $s_2$, or else that no substitution can make $s_1$ and $s_2$ equal. At the end of the proof of the validity of this tableau, a program to compute the unifier of two expressions, denoted by $uf(s_1, s_2)$, is obtained as a result of the transformations of the output associated with the application of the inference rules. The output is initially given by the variable $\theta$.

Goal G1 can be split into the success and the failure conditions. This is done by skolemizing the conjunct $(\forall \sigma)[s_1 \sigma = s_2 \sigma] [\text{the skolem function } \sigma(\theta) \text{ is introduced}]$ and applying the $\lor$-split rule:

| G2 | $(s_1 \sigma = s_2 \sigma) \land \quad mgid(s_1, s_2, \theta) \land \theta \neq nil$ | $\theta$ |
| G3 | $s_1 \sigma = s_2 \sigma [\sigma(\theta) \neq nil] \land \theta = nil$ | $\theta$ |

We then apply the resolution rule between the assertion stating the reflexivity of the equality relation and goal G3; the unification of the two expressions in the boxes causes the replacement of $\theta$ with nil:

| A24 $w = w$ | | |
| G3 | $s_1 \sigma = s_2 \sigma [\sigma(\theta) \neq nil] \land \theta = nil$ | $\theta$ |
| G4 | $s_1 \sigma = s_2 \sigma [\sigma(\theta) \neq nil]$ nill | nill |

The proof is structured according to the case analysis that corresponds to the different syntactic categories of the expressions: constant, variable, and combination cases. The final algorithm is then derived from the case of clash due to different syntactic categories, and from the previously developed cases. In order to improve readability, the proof of the termination of the algorithm is presented in a separate section, even though it is actually part of the proof of the combination case.
5. THE CONSTANT CASE

In the constant case the idea behind the proof is either to state that the unifier is the empty substitution or to detect the clash caused by the attempt to unify two different constants.

The first step consists in applying the equality-replacement rule between the basic assertion for the constant case, stating that substitutions do not affect constants, and the initial goal:

<table>
<thead>
<tr>
<th>A10</th>
<th>if $C(c)$ then $c = \sigma \circ c$</th>
</tr>
</thead>
</table>

| G2   | $\left( s_1 < \theta = s_2 < \theta \right) \land mgid(s_1, s_2, \theta) \land \theta \neq nil$ | $\theta$ |
| G5   | $C(s_1) \land (s_1 = s_2 < \theta) \land mgid(s_1, s_2, \theta) \land \theta \neq nil$ | $\theta$ |

The boxes enclose the two terms unified by this step; the unifier is $\{c \leftarrow s_1, \sigma \leftarrow \theta\}$. The result of the inference step, expressed by goal G5, is obtained by substituting the occurrences of the term $s_1 < \theta$ with $s_1$, that is, the left-hand side of the equality of assertion A10, after the application of the unifier. Goal G5 contains also the antecedent of the implication in assertion A10, as obtained by applying the unifier; this conjunct specifies that $s_1$ must be a constant.

We then apply the same rule to the other side of the equality in the current goal.

<table>
<thead>
<tr>
<th>A10</th>
<th>if $C(c)$ then $c = \sigma \circ c$</th>
</tr>
</thead>
</table>

| G5   | $C(s_1) \land C(s_2) \land s_1 = s_2 \land mgid(s_1, s_2, \theta) \land \theta \neq nil$ | $\theta$ |
| G6   | $C(s_1) \land C(s_2) \land s_1 = s_2 \land mgid(s_1, s_2, \theta) \land \theta \neq nil$ | $\theta$ |

5.1 The mgid Condition

We now prove the most generality and idempotence of the unifier, represented by the conjunct $mgid(s_1, s_2, \theta)$. The mgid condition is expressed by the formula

$$(\forall \sigma)[if \ s_1 < \sigma = s_2 < \sigma then \ \theta \circ \sigma = \sigma]$$

After a skolemization we obtain goal G7:

| G7   | $C(s_1) \land C(s_2) \land s_1 = s_2 \land$

$(if \ s_1 \circ \sigma(\theta) = s_2 < \sigma(\theta) then \ \theta \circ \sigma(\theta) = \sigma(\theta)) \land \theta \neq nil$ | $\theta$ |

We now apply twice the equality replacement rule with the definition of substitution application for constants:

| A10 | if $C(c)$ then $c = \sigma \circ c$ |
5.2. The Constant Clash
The case of constant clash is developed by applying the equality rule twice between the definition of substitution application for constants, expressed by assertion A10, and the initial goal G4:

5.3. The Generation of the If-Then-Else
We are now ready for the last step of this part of the proof, namely, deriving the branch of the program for the constant case. This is achieved by resolving the
following goals:

<table>
<thead>
<tr>
<th>Goal</th>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>G10</td>
<td>( C(s_1) \land C(s_2) \land s_1 = s_2 )</td>
<td>{ }</td>
</tr>
<tr>
<td>G11</td>
<td>( C(s_1) \land C(s_2) \land s_1 \neq s_2 )</td>
<td>nil</td>
</tr>
<tr>
<td>G12</td>
<td>( C(s_1) \land C(s_2) \land (s_1 = s_2 \Rightarrow { }) \land (s_1 \neq s_2 \Rightarrow \text{nil}) )</td>
<td></td>
</tr>
</tbody>
</table>

The application of the resolution rule between goals G10 and G11 leads to the formation of an if-then-else in the output, because the output of the two goals is different. In fact, every application of the resolution rules generates an if-then-else in the output, but we keep it only when the two branches of the conditional are different.

6. THE VARIABLE CASE

In the variable case we show how a substitution is built: in other words, if one of the two expressions to be unified is a variable \( v \), either it does not occur (properly) in the other expression \( e \) and the result is the substitution \( \{ v \leftarrow e \} \), or it does occur in \( e \) and the two expressions \( v \) and \( e \) are not unifiable.

We recall that, when both expressions are the same variable, the proper occurs relation does not hold and the resulting substitution is \( \{ v \leftarrow v \} \), that is, \{ \}.

The basic assertion A11 for the variable case is the definition of substitution application for variable expressions; it states that if we apply the substitution \( \{ v \leftarrow e \} \) to the expression \( v \), we obtain \( e \). We start by applying the equality replacement rule between assertion A11 and the initial goal G2:

A11 \quad \text{if } V(v) \text{ then } e = v \leftarrow \{ v \leftarrow e \} \\

G2 \quad \left( s_1 < \theta \land s_2 < \theta \right) \land mgid(s_1, s_2, \theta) \land \theta \neq \text{nil} \\

G13 \quad V(s_1) \land e = s_2 \leftarrow \{ s_1 \leftarrow e \} \land mgid(s_1, s_2, \{ s_1 \leftarrow e \}) \land \{ s_1 \leftarrow e \} \neq \text{nil} \\

\{ s_1 \leftarrow e \} \\

The next step consists in eliminating the conjunct asserting that the substitution we are building is not nil. To do this we introduce assertion A21, stating that a substitution such as \( \{ v_1 \leftarrow e_1 \} \) is different from nil. This assertion can be easily derived from the fact that nil is not a substitution:

A21 \quad \{ v_1 \leftarrow e_1 \} \neq \text{nil} \\

G13 \quad V(s_1) \land e = s_2 \leftarrow \{ s_1 \leftarrow e \} \land mgid(s_1, s_2, \{ s_1 \leftarrow e \}) \land \{ s_1 \leftarrow e \} \neq \text{nil} \\

G14 \quad V(s_1) \land e = s_2 \leftarrow \{ s_1 \leftarrow e \} \land mgid(s_1, s_2, \{ s_1 \leftarrow e \}) \\

\{ s_1 \leftarrow e \}
The idea behind the following steps of the proof is to specify the expression e so that the application of the substitution \( \{ s_1 \leftarrow e \} \) makes the input expressions equal. At the same time the conditions under which a unifying substitution exists are determined. We consider two cases: \( s_1 \) does not occur in \( s_2 \), and \( s_1 = s_2 \). In both cases \( s_1 \) is unchanged by the application of the substitution.

In the first case we use the variable case of the definition of substitution application, expressed by assertion A12, stating that if a variable does not occur at all in an expression, then replacing it has no effect on that expression. The fact that \( e_1 \) does not occur in \( e_2 \) (when occurs is not proper occurs) is expressed by the \( e_1 \neq e_2 \).

In the second case we apply resolution with equality matching between assertion A11 and again goal G14:

The resolution with equality matching allows for the application of the equality-replacement rule even if the terms do not match perfectly, provided that the equality of their mismatching parts of them is proved. This is the reason for the introduction of the conjunct \( s_1 = s_2 \) in goal G16.

We now combine the results of the two branches of the proof by applying the resolution rule. The resulting goal contains the condition for unifiability in the variable case in terms of the proper occurs relation:

Notice that both goals have the same output, so no conditional is generated. We could have replaced \( \{ s_1 \leftarrow s_2 \} \) with the empty substitution \( \{ \} \) when \( s_1 = s_2 \). This would have introduced the conditional

if \( s_1 = s_2 \) then \( \{ \} \) else \( \{ s_1 \leftarrow s_2 \} \)

in the final program. This can be avoided, because if \( s_1 = s_2 \) then \( \{ s_1 \leftarrow s_2 \} \) and \( \{ \} \) are equal substitutions.
6.1. The mgid Condition

We now show that the substitution \( \{ s_1 \leftarrow s_2 \} \) satisfies both the most-general-unifier condition and the idempotence condition. Since this part of the proof does not affect the output, readers not interested in the details may skip to goal G26.

We first use the definition of the most general and idempotent condition,

\[(\forall \sigma) [if \ s_1 \triangleleft \sigma = s_2 \triangleleft \sigma \ then \ \{ s_1 \leftarrow s_2 \} \circ \sigma = \sigma],\]

and skolemize it:

<table>
<thead>
<tr>
<th>G18</th>
<th>( V(s_1) \land s_1 \not= s_2 \land ) if ( s_1 \triangleleft \bar{\sigma} = s_2 \triangleleft \bar{\sigma} ) then ( { s_1 \leftarrow s_2 } \circ \bar{\sigma} = \bar{\sigma} ) { ( s_1 \leftarrow s_2 ) }</th>
</tr>
</thead>
</table>

We now modify goal G18 by introducing the definition for the equality of unifiers,

\( \sigma_1 = \sigma_2 = (\forall v_1) [v_1 \triangleleft \sigma_1 = v_1 \triangleleft \sigma_2] \),

and applying skolemization again; the result is goal G19:

<table>
<thead>
<tr>
<th>G19</th>
<th>( V(s_1) \land s_1 \not= s_2 \land ) if ( s_1 \triangleleft \bar{\sigma} = s_2 \triangleleft \bar{\sigma} ) then ( \bar{v} \triangleleft ({ s_1 \leftarrow s_2 } \circ \bar{\sigma}) = \bar{v} \triangleleft \bar{\sigma} ) { ( s_1 \leftarrow s_2 ) }</th>
</tr>
</thead>
</table>

We now use the basic property of composition of substitutions and apply the equality rule between it and the left-hand side of goal G19:

<table>
<thead>
<tr>
<th>A14</th>
<th>if ( e \triangleleft (\sigma_1 \circ \sigma_2) ) then ( (e \triangleleft \sigma_1) \triangleleft \sigma_2 )</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>G19</th>
<th>( V(s_1) \land s_1 \not= s_2 \land ) if ( s_1 \triangleleft \bar{\sigma} = s_2 \triangleleft \bar{\sigma} ) then ( \bar{v} \triangleleft ({ s_1 \leftarrow s_2 } \circ \bar{\sigma}) = \bar{v} \triangleleft \bar{\sigma} ) { ( s_1 \leftarrow s_2 ) }</th>
</tr>
</thead>
<tbody>
<tr>
<td>G20</td>
<td>( V(s_1) \land s_1 \not= s_2 \land ) if ( s_1 \triangleleft \bar{\sigma} = s_2 \triangleleft \bar{\sigma} ) then ( \bar{v} \triangleleft ({ s_1 \leftarrow s_2 }) \triangleleft \bar{\sigma} = \bar{v} \triangleleft \bar{\sigma} ) { ( s_1 \leftarrow s_2 ) }</td>
</tr>
<tr>
<td>-----</td>
<td>------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

The idea behind the rest of the mgid condition proof is simple, because we have to verify that either \( s_1 = \bar{v} \) and the goal is true because of the definition of substitution application for variables, or \( s_1 \not= \bar{v} \) and the goal is trivially true. The details of the proof are shown below; the first case starts with an application of the equality matching rule:

<table>
<thead>
<tr>
<th>A11</th>
<th>if ( V(v) ) then ( v \triangleleft (v \leftarrow e) = e )</th>
</tr>
</thead>
<tbody>
<tr>
<td>G20</td>
<td>( V(s_1) \land s_1 \not= s_2 \land ) if ( s_1 \triangleleft \bar{\sigma} = s_2 \triangleleft \bar{\sigma} ) then ( \bar{v} \triangleleft ({ s_1 \leftarrow s_2 }) \triangleleft \bar{\sigma} = \bar{v} \triangleleft \bar{\sigma} ) { ( s_1 \leftarrow s_2 ) }</td>
</tr>
<tr>
<td>-----</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>G21</td>
<td>( V(s_1) \land s_1 \not= s_2 \land ) ( (if \ s_1 \triangleleft \bar{\sigma} = s_2 \triangleleft \bar{\sigma} \ then \ s_2 \triangleleft \bar{\sigma} = s_1 \triangleleft \bar{\sigma}) \land \bar{v} = s_1 ) { ( s_1 \leftarrow s_2 ) }</td>
</tr>
</tbody>
</table>
The implication is eliminated by the commutativity of equality:

| A25 | if $w_1 = w_2$ then $w_2 = w_1$ |
| G21 | $V(s_1) \land s_1 \neq s_2 \land$  
     | (if $s_1 < \bar{\sigma} = s_2 < \bar{\sigma}$ then $s_2 < \bar{\sigma} = s_1 < \bar{\sigma}$) \land \bar{\nu} = s_1 \{ s_1 \leftarrow s_2 \} |
| G22 | $V(s_1) \land s_1 \neq s_2 \land \bar{\nu} = s_1 \{ s_1 \leftarrow s_2 \}$ |

Here we develop the other case using goal G20 and assertion A12:

| A12 | if $V(v_1) \land v_1 \neq e_1 \land v_1 \neq e_1$ then $e_1 = e_1 \leftarrow \{ v_1 \leftarrow e_2 \}$ |
| G20 | $V(s_1) \land s_1 \neq s_2 \land$  
     | if $s_1 < \bar{\sigma} = s_2 < \bar{\sigma}$ then $\bar{\nu} < \{ s_1 \leftarrow s_2 \} < \bar{\sigma} = \bar{\sigma} \neq s_1 \{ s_1 \leftarrow s_2 \}$ |
| G23 | $V(s_1) \land s_1 \neq s_2 \land$  
     | (if $s_1 < \bar{\sigma} = s_2 < \bar{\sigma}$ then $\bar{\nu} < \bar{\sigma} = \bar{\nu} < \bar{\sigma}$) \land \bar{\nu} \neq s_1 \land \bar{\nu} \neq s_1 \{ s_1 \leftarrow s_2 \}$ |

A resolution step allows us to remove the conjunct about the proper occurrence relation:

| A8 | $v_1 \neq v_2$ |
| G23 | $V(s_1) \land s_1 \neq s_2 \land$  
     | (if $s_1 < \bar{\sigma} = s_2 < \bar{\sigma}$ then $\bar{\nu} < \bar{\sigma} = \bar{\nu} < \bar{\sigma}$) \land \bar{\nu} \neq s_1 \land \bar{\nu} \neq s_1 \{ s_1 \leftarrow s_2 \}$ |
| G24 | $V(s_1) \land s_1 \neq s_2 \land$  
     | (if $s_1 < \bar{\sigma} = s_2 < \bar{\sigma}$ then $\bar{\nu} < \bar{\sigma} = \bar{\nu} < \bar{\sigma}$) \land \bar{\nu} \neq s_1 \{ s_1 \leftarrow s_2 \}$ |

The implication of goal G24 is eliminated by a resolution with the reflexivity of equality and a transformation rule:

| A24 | $w = w$ |
| G24 | $V(s_1) \land s_1 \neq s_2 \land$  
     | (if $s_1 < \bar{\sigma} = s_2 < \bar{\sigma}$ then $\bar{\nu} < \bar{\sigma} = \bar{\nu} < \bar{\sigma}$) \land \bar{\nu} \neq s_1 \{ s_1 \leftarrow s_2 \}$ |
| G25 | $V(s_1) \land s_1 \neq s_2 \land \bar{\nu} \neq s_1 \{ s_1 \leftarrow s_2 \}$ |
Finally the results of the two branches of the proof are combined, by a resolution step:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G22</td>
<td>$V(s_1) \land s_1 \not= s_2 \land \bar{v} = s_1$</td>
<td>${ s_1 \leftarrow s_2 }$</td>
</tr>
<tr>
<td>G25</td>
<td>$V(s_1) \land s_1 \not= s_2 \land \bar{v} \neq s_1$</td>
<td>${ s_1 \leftarrow s_2 }$</td>
</tr>
<tr>
<td>G26</td>
<td>$V(s_1) \land s_1 \not= s_2$</td>
<td>${ s_1 \leftarrow s_2 }$</td>
</tr>
</tbody>
</table>

This concludes the proof of the most general and idempotent condition for the variable case. As already said, the output has not been modified by this part of the proof.

6.2. The Variable Clash

The variable clash is developed by transforming the condition on the nonexistence of the unifier into the proper-occurrence relation on the input expressions. This is done in two steps: In the first one we transform the inequality of the failure goal G4 into proper occurrence.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A9</td>
<td>if $e_1 \ll e_2$ then $e_1 \neq e_2$</td>
</tr>
<tr>
<td>G4</td>
<td>$s_1 &lt; \bar{\sigma}(\text{nil}) \not&lt; s_2 &lt; \bar{\sigma}(\text{nil})$</td>
</tr>
<tr>
<td>G27</td>
<td>$s_1 &lt; \bar{\sigma}(\text{nil}) \ll s_2 &lt; \bar{\sigma}(\text{nil})$</td>
</tr>
</tbody>
</table>

In the second step we use the monotonicity of the proper-occurrence relation to express the condition in terms of the input expressions:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A16</td>
<td>if $e_1 \ll e_2$ then $e_1 \ll \sigma \ll e_2 \ll \sigma$</td>
</tr>
<tr>
<td>G27</td>
<td>$s_1 \ll \bar{\sigma}(\text{nil}) \ll s_2 \ll \bar{\sigma}(\text{nil})$</td>
</tr>
<tr>
<td>G28</td>
<td>$s_1 \ll s_2$</td>
</tr>
</tbody>
</table>

6.3. The Generation of the If-Then-Else

The variable branch of the algorithm can now be easily derived by resolving the goals already developed; notice that this step causes the occur check to appear in the output:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G26</td>
<td>$V(s_1) \land s_1 \not= s_2$</td>
<td>${ s_1 \leftarrow s_2 }$</td>
</tr>
<tr>
<td>G28</td>
<td>$s_1 \ll s_2$</td>
<td>$\text{nil}$</td>
</tr>
<tr>
<td>G29</td>
<td>$V(s_1)$ if $s_1 \not= s_2$ then ${ s_1 \leftarrow s_2 }$ else $\text{nil}$</td>
<td></td>
</tr>
</tbody>
</table>
7. THE COMBINATION CASE

The combination case requires the introduction of the induction hypothesis, which generates the recursive calls in the program. The proof strategy is the following: to get the unifier of the combination as a composition of the head unifier and the tail unifier, first we try to resolve the tails of the goal against the induction hypothesis, and then we try to resolve the heads. This strategy was adopted by Manna and Waldinger [6]; furthermore it is suggested by the order of the recursive calls in the expected output. However, it is not possible to have a complete separation between the proofs of the conjuncts of the heads and the tails. In particular, the proof branch of the mgid condition for the tails requires the existence of a most general idempotent unifier of the heads.

The presentation of the proof of the combination case is arranged in three parts. The first one splits the problem of finding a most general idempotent unifier in terms of the data structure components, namely the heads and the tails. The induction hypothesis is then introduced, which allows for the derivation of the tail branch (part two), and, subsequently, of the head branch (part three). As already observed, the proof of the mgid condition for the tails contains the derivation of the head unifier as well. The proof of the termination, consisting of the proof of the ordering relation introduced by the resolutions with the induction hypothesis, is presented in the next section, even though it is actually part of the proof of the combination case.

7.1. Preliminary Steps

The proof steps described below use the basic assertions about expressions, and allow us to split the proof into separate branches corresponding to the data-structure components.

We start by applying the equality-replacement rule between the two sides of the equality of the initial goal and the assertion stating the decomposition property of combinations:

\begin{align*}
A4 & \text{ if } B(b) \text{ then } b = h(b) \circ t(b) \\
G2 & \left( \begin{array}{c} s_1 \theta = s_2 \theta \end{array} \right) \land \text{mgid}(s_1, s_2, \theta) \land \theta \neq \text{nil} \\
G30 & B(s_1) \land B(s_2) \land (h(s_1) \circ t(s_1)) \theta = (h(s_2) \circ t(s_2)) \theta \land \text{mgid}(s_1, s_2, \theta) \land \theta \neq \text{nil}
\end{align*}

We then use twice the definition of substitution for combinations:

\begin{align*}
A13 & \text{ if } B(b) \text{ then } b \circ \sigma = (h(b) \circ \sigma) \circ (t(b) \circ \sigma)
\end{align*}
By a resolution step between assertion A5 and the current goal we can now split the problem of unifying two combinations into the two distinct problems of unifying the heads and the tails of the combinations. To obtain goal G32, we implicitly use the definitions of the head and the tail of combinations, namely, \( h(e_1 \circ e_2) = e_1 \) and \( t(e_1 \circ e_2) = e_2 \):

\[
B(s_1) \land B(s_2) \land \left( (h(s_1) \circ t(s_1)) \land \theta \right) = \left( (h(s_2) \circ t(s_2)) \land \theta \right)
\]

We then resolve four times with the definition of substitution composition:

\[
B(s_1) \land B(s_2) \land (h(s_1) \land \theta) = (h(s_2) \land \theta)
\]

We then apply resolution with equality matching, thus making the assumption that \( \theta \) has no influence on the heads of the combinations:

\[
e \circ (\theta_h \circ \theta_t) = (e \circ \theta) \circ \theta_i
\]

We then resolve four times with the definition of substitution composition:

\[
B(s_1) \land B(s_2) \land (h(s_1) \land \theta) = (h(s_2) \land \theta)
\]

We then apply resolution with equality matching, thus making the assumption that \( \theta \) has no influence on the heads of the combinations:
We finally split the condition $\theta_h \land \theta_i \neq \text{nil}$, by a resolution step and a transformation rule:

$$\begin{align*}
\text{A22} & \quad \text{if } \sigma_1 \land \sigma_2 = \text{nil} \quad \text{then } \sigma_1 = \text{nil} \lor \sigma_2 = \text{nil} \\
\text{G34} & \quad B(s_1) \land B(s_2) \land h(s_1) < \theta_h = h(s_2) < \theta_h \land \\
& \quad (t(s_1) < \theta_h) < \theta_i = (t(s_2) < \theta_h) < \theta_i \land \\
& \quad \text{mgid}(s_1, s_2, \theta_h \land \theta_i) \land \theta_h \land \theta_i \neq \text{nil} \\
\text{G35} & \quad B(s_1) \land B(s_2) \land h(s_1) < \theta_h = h(s_2) < \theta_h \land \\
& \quad (t(s_1) < \theta_h) < \theta_i = (t(s_2) < \theta_h) < \theta_i \land \\
& \quad \text{mgid}(s_1, s_2, \theta_h \land \theta_i) \land \theta_h \land \theta_i \neq \text{nil}
\end{align*}$$

This concludes the split of the success goal $G_2$, in terms of the head and tail components of the combination.

### 7.1.1. The Combination Clash

We now develop the clash branch of the proof, by splitting the condition expressed by goal $G_4$ into the class of the heads and the clash of the tails. The steps of the proof are straightforward. The first one is a double application of the equality replacement rule:

$$\begin{align*}
\text{A13} & \quad \text{if } B(b) \quad \text{then } b \land \sigma = (h(b) \land \sigma) \land (t(b) \land \sigma) \\
\text{G4} & \quad s_1 \land \sigma(\text{nil}) \neq s_2 \land \sigma(\text{nil}) \\
\text{G36} & \quad B(s_1) \land B(s_2) \land (h(s_1) \land \sigma(\text{nil})) \land (t(s_1) \land \sigma(\text{nil})) \neq \\
& \quad (h(s_2) \land \sigma(\text{nil})) \land (t(s_2) \land \sigma(\text{nil})) \\
& \quad \text{nil}
\end{align*}$$

The next step is an application of the resolution rule:

$$\begin{align*}
\text{A5} & \quad \text{if } h(b_1) = h(b_2) \land t(b_1) = t(b_2) \quad \text{then } b_1 - b_2
\end{align*}$$
Goal G37 can be finally split using the $\lor$-split rule:

<table>
<thead>
<tr>
<th>G38</th>
<th>$B(s_1) \land B(s_2) \land h(s_1) &lt; \tilde{\sigma}(\text{nil}) \neq h(s_2) &lt; \tilde{\sigma}(\text{nil})$</th>
<th>nil</th>
</tr>
</thead>
<tbody>
<tr>
<td>G39</td>
<td>$B(s_1) \land B(s_2) \land t(s_1) &lt; \tilde{\sigma}(\text{nil}) \neq t(s_2) &lt; \tilde{\sigma}(\text{nil})$</td>
<td>nil</td>
</tr>
</tbody>
</table>

7.1.2. The Induction Hypothesis. We can now introduce the induction hypothesis, assuming the existence of a well-founded relation over pairs of expressions. This relation represents the termination condition, and will be specified and proved when we discuss the termination:

The induction hypothesis is used first to resolve with the tails of the combination, then to resolve with the heads.

7.2. Tails

Here we develop the proof for the tails of the combination. The first step is the resolution of goal G35 with the induction hypothesis:
In goal G40 we write $\theta^*_t$ instead of $uf(t(s_1) < \theta_h, t(s_2) < \theta_h)$ to keep the size of the formulas manageable. The unifier of this step of the proof is $\{r_1 \leftarrow t(s_1) < \theta_h, r_2 \leftarrow t(s_2) < \theta_h, \theta_t \leftarrow uf(t(s_1) < \theta_h, t(s_2) < \theta_h)\}$.

The resolution with the induction hypothesis brings into the goal the ordering condition for the tails and the negation of the failure condition. In fact, let the structure of the induction hypothesis be represented by the formula

if $A$ then $(B \land C \land D) \lor (E \land F)$;

when the conjuncts in the boxes corresponding to $B$ and $D$ are substituted by $false$, the formula becomes

if $A$ then $(false \land C \land false) \lor (E \land F)$,

which simplifies to

if $A$ then $(E \land F)$.

The assertion-goal resolution requires then that this formula be negated when added in the resulting goal, giving the two conjuncts $A \land \neg(E \land F)$, which, after the application of the unifier, correspond to the last two lines of goal G40.

7.2.1. The mgid Condition. We now prove the most general and idempotence conditions for the tails. Since this involves the use of the most general and idempotence condition for the heads, we need to apply the induction hypothesis to derive the unifier of the heads.

Expanding the definition of the most general and idempotent condition, namely

$(\forall \sigma)[if s_1 < \sigma = s_2 < \sigma then \theta \circ \sigma = \sigma]$,

and skolemizing it, we obtain goal G41. Since there will be no variable conflicts, we use $\bar{\sigma}$ without the explicit indication of the arguments:

$$
G41 \quad B(s_1) \land B(s_2) \land h(s_1) < \theta_h = h(s_2) < \theta_h \land
\begin{align*}
& \left( t(s_1) < \theta_h \right) \land \left( t(s_2) < \theta_h \right) \land \\
& \neg((t(s_1) < \theta_h) \land \sigma \neq (t(s_2) < \theta_h) \land \sigma \land \theta_t^* = \text{nil})
\end{align*}
$$

| G35 | $B(s_1) \land B(s_2) \land h(s_1) < \theta_h = h(s_2) < \theta_h \land$
| | $\left( t(s_1) < \theta_h \right) \land \left( t(s_2) < \theta_h \right) \land$
| | $mgid(s_1, s_2, \theta_h \circ \theta_t) \land \theta_h \neq \text{nil} \land \theta_t \neq \text{nil}$
| $\theta_h \circ \theta_t$ |

| G40 | $B(s_1) \land B(s_2) \land h(s_1) < \theta_h = h(s_2) < \theta_h \land$
| | $mgid(s_1, s_2, \theta_h \circ \theta^*_t) \land \theta_h \neq \text{nil} \land$
| | $\left( t(s_1) < \theta_h \right) \land \left( t(s_2) < \theta_h \right) \land$
| | $\neg((t(s_1) < \theta_h) \land \sigma \neq (t(s_2) < \theta_h) \land \sigma \land \theta^*_t = \text{nil})$ | $\theta_h \circ \theta^*_t$ |
We now transform the antecedent of the \textit{mgid} conjunct by splitting the condition of $\bar{\sigma}$ being a unifier of $s_1$ and $s_2$ in the corresponding equations for the heads and the tails of the combinations. This requires a double application of the equality-replacement rule with assertion A13, and a resolution with assertion A5:

<table>
<thead>
<tr>
<th>A13</th>
<th>if $B(b)$ then $b \triangleleft \sigma = (h(b) \triangleleft \sigma) \circ (t(b) \triangleleft \sigma)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$B(s_1) \wedge B(s_2) \wedge h(s_1) \triangleleft \theta_h = h(s_2) \triangleleft \theta_h$ $\wedge$</td>
</tr>
<tr>
<td></td>
<td>$\begin{cases} s_1 \triangleleft \bar{\sigma} = s_2 \triangleleft \bar{\sigma} \text{ then } \theta_h \circ \theta_t^* \circ \bar{\sigma} = \bar{\sigma} \end{cases}$ $\wedge$</td>
</tr>
<tr>
<td>G41</td>
<td>$\theta_h \neq \text{nil} \wedge \langle t(s_1) \triangleleft \theta_h, t(s_2) \triangleleft \theta_h \rangle \prec_u \langle s_1, s_2 \rangle \wedge$</td>
</tr>
<tr>
<td></td>
<td>$\lnot ((t(s_1) \triangleleft \theta_h) \triangleleft \sigma \neq (t(s_2) \triangleleft \theta_h) \triangleleft \sigma \wedge \theta_t^* = \text{nil})$</td>
</tr>
<tr>
<td></td>
<td>$\theta_h \circ \theta_t^*$</td>
</tr>
<tr>
<td>G42</td>
<td>$B(s_1) \wedge B(s_2) \wedge h(s_1) \triangleleft \theta_h = h(s_2) \triangleleft \theta_h$ $\wedge$</td>
</tr>
<tr>
<td></td>
<td>$\begin{cases} (h(s_1) \triangleleft \bar{\sigma}) \circ (t(s_1) \triangleleft \bar{\sigma}) = (h(s_2) \triangleleft \bar{\sigma}) \circ (t(s_2) \triangleleft \bar{\sigma}) \text{ then } \theta_h \circ \theta_t^* \circ \bar{\sigma} = \bar{\sigma} \wedge \theta_h \neq \text{nil} \wedge$</td>
</tr>
<tr>
<td></td>
<td>$\langle t(s_1) \triangleleft \theta_h, t(s_2) \triangleleft \theta_h \rangle \prec_u \langle s_1, s_2 \rangle \wedge$</td>
</tr>
<tr>
<td></td>
<td>$\lnot ((t(s_1) \triangleleft \theta_h) \triangleleft \sigma \neq (t(s_2) \triangleleft \theta_h) \triangleleft \sigma \wedge \theta_t^* = \text{nil})$</td>
</tr>
<tr>
<td></td>
<td>$\theta_h \circ \theta_t^*$</td>
</tr>
<tr>
<td>A5</td>
<td>if $h(b_1) = h(b_2)$ and $t(b_1) = t(b_2)$ then $b_1 = b_2$</td>
</tr>
</tbody>
</table>

The next step consists of an application of the equality replacement rule with the induction hypothesis:

| G42 | $\begin{cases} B(s_1) \wedge B(s_2) \wedge h(s_1) \triangleleft \theta_h = h(s_2) \triangleleft \theta_h \wedge \text{ then } \theta_h \circ \theta_t^* \circ \bar{\sigma} = \bar{\sigma} \wedge \theta_h \neq \text{nil} \wedge \wedge$ |
|     | $\langle t(s_1) \triangleleft \theta_h, t(s_2) \triangleleft \theta_h \rangle \prec_u \langle s_1, s_2 \rangle \wedge$ |
|     | $\lnot ((t(s_1) \triangleleft \theta_h) \triangleleft \sigma \neq (t(s_2) \triangleleft \theta_h) \triangleleft \sigma \wedge \theta_t^* = \text{nil})$ |
|     | $\theta_h \circ \theta_t^*$ |

| G43 | $\begin{cases} B(s_1) \wedge B(s_2) \wedge h(s_1) \triangleleft \theta_h = h(s_2) \triangleleft \theta_h \wedge \text{ then } \theta_h \circ \theta_t^* \circ \bar{\sigma} = \bar{\sigma} \wedge \theta_h \neq \text{nil} \wedge$ |
|     | $\langle t(s_1) \triangleleft \theta_h, t(s_2) \triangleleft \theta_h \rangle \prec_u \langle s_1, s_2 \rangle \wedge$ |
|     | $\lnot ((t(s_1) \triangleleft \theta_h) \triangleleft \sigma \neq (t(s_2) \triangleleft \theta_h) \triangleleft \sigma \wedge \theta_t^* = \text{nil})$ |
|     | $\theta_h \circ \theta_t^*$ |

A27 if $r_1 \triangleleft \sigma_1 = r_2 \triangleleft \sigma_1$ then $uf(r_1, r_2) \triangleleft \sigma_1 = \sigma_1$ $\wedge$ $uf(r_1, r_2) \neq \text{nil}$ $\lor \lnot (r_1 \triangleleft \sigma = r_2 \triangleleft \sigma) \wedge uf(r_1, r_2) = \text{nil}$
The unifier of this step of the proof is \( \{ r_1 \leftarrow t(s_1) \land \theta_h, \ r_2 \leftarrow t(s_2) \land \theta_h, \ \sigma_1 \leftarrow \bar{\sigma} \} \).

We now apply twice the equally replacement rule with the definition of substitution composition:

\[
A14 \quad e \triangleleft (\sigma_1 \land \sigma_2) = (e \triangleleft \sigma_1) \land \sigma_2
\]

The idea behind the proof is now that, if the heads of the combinations are unifiable, then there exists a more general idempotent \( \theta^* \) such that \( e^* \land \sigma = \bar{\sigma} \). Therefore we apply the resolution rule against the induction hypothesis again, getting a unifier for the heads. This first resolution with the inductive hypothesis...
could in fact be delayed, since it is the next step that allows us to prove the \textit{mgid} condition for the tails. The complete resolution with the inductive hypothesis for the heads has been anticipated for readability reasons; otherwise a number of variables would remain unbound, making the notation heavier:

\begin{align*}
\text{if } \langle r_1, r_2 \rangle \prec_U \langle s_1, s_2 \rangle \text{ then} & \\
& r_1 \prec u f (r_1, r_2) = r_2 \prec u f (r_1, r_2) \wedge \\
& \text{mgid} (r_1, r_2, u f (r_1, r_2)) \wedge \text{uf} (r_1, r_2) \neq \text{nil} \\
& \forall r_1 \prec \sigma \neq r_2 \prec \sigma \wedge \text{uf} (r_1, r_2) = \text{nil}
\end{align*}

<table>
<thead>
<tr>
<th>A27</th>
</tr>
</thead>
</table>
| \begin{align*}
B (s_1) \wedge B (s_2) \wedge & h (s_1) \prec \theta_h = h (s_2) \prec \theta_h \wedge \\
(\text{if } h (s_1) \prec \bar{\sigma} = h (s_2) \prec \bar{\sigma} \wedge t (s_1) \prec \bar{\sigma} = t (s_2) \prec \bar{\sigma} & \text{ then } \theta_h \circ \bar{\sigma} - \bar{\sigma} \wedge \theta_h \neq \text{nil}) \\
(\text{if } h (s_1) \prec \theta_h \circ \bar{\sigma} = t (s_2) \prec \theta_h \circ \bar{\sigma} & \wedge \\
\langle t (s_1) \prec \theta_h, t (s_2) \prec \theta_h \rangle \prec_U \langle s_1, s_2 \rangle & \wedge \\
-((t (s_1) \prec \theta_h) \prec \sigma \neq (t (s_2) \prec \theta_h) \prec \sigma \wedge \theta_t^* = \text{nil})
\end{align*}|

<table>
<thead>
<tr>
<th>G45</th>
</tr>
</thead>
</table>
| \begin{align*}
B (s_1) \wedge B (s_2) \wedge & h (s_1) \prec \theta_h = h (s_2) \prec \theta_h \wedge \\
(\text{if } h (s_1) \prec \bar{\sigma} = h (s_2) \prec \bar{\sigma} \wedge t (s_1) \prec \bar{\sigma} = t (s_2) \prec \bar{\sigma} & \text{ then } \theta_h \circ \bar{\sigma} - \bar{\sigma} \wedge \theta_h \neq \text{nil}) \\
(\text{if } h (s_1) \prec \theta_h \circ \bar{\sigma} = t (s_2) \prec \theta_h \circ \bar{\sigma} & \wedge \\
\langle t (s_1) \prec \theta_h^*, t (s_2) \prec \theta_h^* \rangle \prec_U \langle s_1, s_2 \rangle & \wedge \\
-((t (s_1) \prec \theta_h^*) \prec \sigma \neq (t (s_2) \prec \theta_h^*) \prec \sigma \wedge \theta_t^{**} = \text{nil}) & \wedge \\
\langle h (s_1), h (s_2) \rangle \prec_U \langle s_1, s_2 \rangle & \wedge \\
-((h (s_1) \prec \sigma \neq h (s_2) \prec \sigma \wedge \theta_h^{**} = \text{nil})
\end{align*}|

In goal G46 we use a new symbol \( \theta_h^* \) instead of the recursive call \( u f (h (s_1), h (s_2)) \). We also use \( \theta_t^{**} \), standing for \( \theta_t^* \) where \( \theta_h \) has been replaced by \( \theta_h^* \), that is,

\[
\theta_t^{**} = u f (t (s_1) \prec u f (h (s_1), h (s_2)), t (s_2) \prec u f (h (s_1), h (s_2))).
\]

Again two conjuncts specifying the ordering condition and the negation of the failure condition for the heads are introduced by the resolution with the induction hypothesis.\(^2\)

\[^2\text{In the conjunct negating the failure condition for the heads, the variable } \sigma \text{ should be renamed to avoid conflicts with the other free variable of the similar conjunct for the tails. We have avoided the renaming to simplify the notation: in fact both variables never get bound in the rest of the proof.}\]
We then apply the equality replacement rule with the induction hypothesis to prove the mgid condition:

\[
\text{if } \langle r_1, r_2 \rangle \prec_u \langle s_1, s_2 \rangle \text{ then }
\]
\[
r_1 \prec uf(r_1, r_2) = r_2 \prec uf(r_1, r_2) \wedge
\]
\[
A27 (\text{if } r_1 \prec \sigma_1 = r_2 \prec \sigma_1 \text{ then } \boxed{uf(r_1, r_2) \circ \sigma_1 = \sigma_1}) \wedge
\]

\[
uf(r_1, r_2) \neq \text{nil}
\]

\[
\vee \neg (r_1 \prec \sigma = r_2 \prec \sigma) \wedge uf(r_1, r_2) = \text{nil}
\]

\[
G46 B(s_1) \wedge B(s_2) \wedge
\]

\[
(\text{if } h(s_1) \prec \bar{\sigma} = h(s_2) \prec \bar{\sigma} \wedge t(s_1) \prec \bar{\sigma} = t(s_2) \prec \bar{\sigma}
\]

\[
\text{then } \boxed{\theta_h^* \circ \bar{\sigma} = \bar{\sigma}} \wedge t(s_1) \prec \left( \boxed{\theta_h^* \circ \bar{\sigma}} \right)
\]

\[
= t(s_2) \prec \left( \theta_h^* \circ \bar{\sigma} \right) \wedge
\]

\[
\langle t(s_1) \prec \theta_h^*, \ t(s_2) \prec \theta_h^* \rangle \prec_u \langle s_1, s_2 \rangle \wedge
\]

\[
\neg (\langle t(s_1) \prec \theta_h^* \rangle \prec \sigma \neq (t(s_2) \prec \theta_h^* \rangle \prec \sigma \wedge \theta^*_t = \text{nil})
\]

\[
\langle h(s_1), h(s_2) \rangle \prec_u \langle s_1, s_2 \rangle \wedge
\]

\[
(h(s_1) \prec \sigma = h(s_2) \prec \sigma \wedge \theta_h^* = \text{nil})
\]

\[
G47 B(s_1) \wedge B(s_2) \wedge
\]

\[
(\text{if } h(s_1) \prec \bar{\sigma} = h(s_2) \prec \bar{\sigma} \wedge t(s_1) \prec \bar{\sigma} = t(s_2) \prec \bar{\sigma}
\]

\[
\text{then } \bar{\sigma} = \bar{\sigma}) \wedge
\]

\[
h(s_1) \prec \bar{\sigma} = h(s_2) \prec \bar{\sigma} \wedge t(s_1) \prec \bar{\sigma} = t(s_2) \prec \bar{\sigma} \wedge
\]

\[
\langle t(s_1) \prec \theta_h^*, \ t(s_2) \prec \theta_h^* \rangle \prec_u \langle s_1, s_2 \rangle \wedge
\]

\[
\neg (\langle t(s_1) \prec \theta_h^* \rangle \prec \sigma \neq (t(s_2) \prec \theta_h^* \rangle \prec \sigma \wedge \theta^*_t = \text{nil}) \wedge
\]

\[
\langle h(s_1), h(s_2) \rangle \prec_u \langle s_1, s_2 \rangle \wedge
\]

\[
(h(s_1) \prec \sigma = h(s_2) \prec \sigma \wedge \theta_h^* = \text{nil})
\]

The implication of goal G47 can now be eliminated using the reflexivity of equality and a transformation rule:

\[
A24 \boxed{w = w}
\]
By a resolution with goal G43 it is possible to eliminate of the conditions of $\bar{\sigma}$ being a unifier of both the heads and the tails of the inputs.

In the last step we implicitly unify all the other conjuncts, so that $\theta_h$ is bound to $\theta_h^*$. 
7.2.2. *The Clash of the Tails.* The next step of the proof is to resolve the failure branch for the tails with the induction hypothesis. In order to do that, we transform it into a suitable form, expressing $\bar{\sigma}(\text{nil})$ as a composition of the unifier of the heads and $\bar{\sigma}(\text{nil})$ itself. This requires a step on the unifier of the heads, to resolve the failure branch for the heads of the combinations against the induction hypothesis:

| \text{A27} | \text{if} \langle r_1, r_2 \rangle \not\prec u \langle s_1, s_2 \rangle \text{ then} \\
| | r_1 \not\prec u f(r_1, r_2) = r_2 \not\prec u f(r_1, r_2) \land \\
| | \left( \begin{array}{l}
| | \begin{array}{l}
| | \text{if} \quad r_1 \not\prec \sigma_1 = r_2 \not\prec \sigma_1 \\
| | \text{then} \quad uf(r_1, r_2) \circ \sigma_1 = \sigma_1 \\
| | \end{array}
| | \right) \land \\
| | \bar{u} f(r_1, r_2) \neq \text{nil} \\
| | \lor \\
| | \left( \begin{array}{l}
| | \begin{array}{l}
| | \neg (r_1 \not\prec \sigma = r_2 \not\prec \sigma) \land uf(r_1, r_2) = \text{nil}
| | \end{array}
| | \right) \\
| | 

| \text{G38} | B(s_1) \land B(s_2) \land \neg (h(s_1) \not\prec \bar{\sigma}(\text{nil}) = h(s_2) \not\prec \bar{\sigma}(\text{nil})) \\
| | 

| \text{G50} | B(s_1) \land B(s_2) \land \neg (h(s_1), h(s_2)) \not\prec u \langle s_1, s_2 \rangle \land \\
| | \neg (h(s_1) \not\prec \theta_h^* \neq h(s_2) \not\prec \theta_h^* \land \theta_h^* \circ \bar{\sigma}(\text{nil}) = \bar{\sigma}(\text{nil}) \land \theta_h^* \neq \text{nil}) \\
| | 

Intuitively, the next step can be justified by the following reasoning: if the failure condition holds, then any of the conjuncts in the success goal can be false; in particular we can try to prove that the conjunct $\theta_h^* \circ \bar{\sigma}(\text{nil}) = \bar{\sigma}(\text{nil})$ is false, using it to transform the failure branch for the tails. This is done by an application of the equality-replacement rule:

| \text{G50} | B(s_1) \land B(s_2) \land \langle h(s_1), h(s_2) \rangle \not\prec u \langle s_1, s_2 \rangle \land \\
| | \neg (h(s_1) \not\prec \theta_h^* \neq \theta_h^* \land \theta_h^* \circ \bar{\sigma}(\text{nil}) = \bar{\sigma}(\text{nil}) \land \theta_h^* \neq \text{nil}) \\
| | 

| \text{G39} | B(s_1) \land B(s_2) \land t(s_1) \not\prec \bar{\sigma}(\text{nil}) \neq t(s_2) \not\prec \bar{\sigma}(\text{nil}) \\
| | 

| \text{G51} | B(s_1) \land B(s_2) \land \langle h(s_1), h(s_2) \rangle \not\prec u \langle s_1, s_2 \rangle \land \\
| | t(s_1) \not\prec (\theta_h^* \circ \bar{\sigma}(\text{nil})) \neq t(s_2) \not\prec (\theta_h^* \circ \bar{\sigma}(\text{nil})) \\
| | 

We then apply twice the equality-replacement rule with the definition of substitution composition:

| \text{A14} | e \not\prec (\sigma_1 \circ \sigma_2) = (e \not\prec \sigma_1) \not\prec \sigma_2 \\
| | 

| \text{G51} | B(s_1) \land B(s_2) \land \langle h(s_1), h(s_2) \rangle \not\prec u \langle s_1, s_2 \rangle \land \\
| | t(s_1) \not\prec (\theta_h^* \circ \bar{\sigma}(\text{nil})) \neq t(s_2) \not\prec (\theta_h^* \circ \bar{\sigma}(\text{nil})) \\
| | 

| \text{G52} | B(s_1) \land B(s_2) \land \langle h(s_1), h(s_2) \rangle \not\prec u \langle s_1, s_2 \rangle \land \\
| | (t(s_1) \not\prec \theta_h^*) \not\prec \bar{\sigma}(\text{nil}) \neq (t(s_2) \not\prec \theta_h^*) \not\prec \bar{\sigma}(\text{nil}) \\
| | 

| \text{G52} | B(s_1) \land B(s_2) \land \langle h(s_1), h(s_2) \rangle \not\prec u \langle s_1, s_2 \rangle \land \\
| | (t(s_1) \not\prec \theta_h^*) \not\prec \bar{\sigma}(\text{nil}) \neq (t(s_2) \not\prec \theta_h^*) \not\prec \bar{\sigma}(\text{nil}) \\
| | 

| \text{G52} | B(s_1) \land B(s_2) \land \langle h(s_1), h(s_2) \rangle \not\prec u \langle s_1, s_2 \rangle \land \\
| | (t(s_1) \not\prec \theta_h^*) \not\prec \bar{\sigma}(\text{nil}) \neq (t(s_2) \not\prec \theta_h^*) \not\prec \bar{\sigma}(\text{nil}) \\
| | 

We then apply twice the equality-replacement rule with the definition of substitution composition.
and finally resolve against the failure branch of the induction hypothesis; \( \theta^{*'} \) and \( \theta^{*}_{h} \) are the same as before.

If \( \langle r_{1}, r_{2} \rangle \prec_{u} \langle s_{1}, s_{2} \rangle \) then

\[
\begin{align*}
& r_{1} \prec uf(r_{1}, r_{2}) = r_{2} \prec uf(r_{1}, r_{2}) \land \\
& mgid(r_{1}, r_{2}, uf(r_{1}, r_{2})) \land uf(r_{1}, r_{2}) \neq nil \\
& \lor r_{1} \prec \sigma \neq r_{2} \prec \sigma \land uf(r_{1}, r_{2}) = nil
\end{align*}
\]

<table>
<thead>
<tr>
<th>Rule</th>
<th>Premise</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>G52</td>
<td>( B(s_{1}) \land B(s_{2}) \land \langle h(s_{1}), h(s_{2}) \rangle \prec_{u} \langle s_{1}, s_{2} \rangle \land \langle t(s_{1}) \prec \theta^{<em>}<em>{h}, t(s</em>{2}) \prec \theta^{</em>}<em>{h} \rangle \prec</em>{u} \langle s_{1}, s_{2} \rangle \land \langle (t(s_{1}) \prec \theta^{<em>}_{h}) \land \theta^{</em>'} = (t(s_{2}) \prec \theta^{<em>}<em>{h}) \land mgid(s</em>{1}, s_{2}, \theta^{</em>'}) \land \theta^{*'} \neq nil \rangle )</td>
<td>( nil )</td>
</tr>
<tr>
<td>G53</td>
<td>( B(s_{1}) \land B(s_{2}) \land \langle h(s_{1}), h(s_{2}) \rangle \prec_{u} \langle s_{1}, s_{2} \rangle \land \langle t(s_{1}) \prec \theta^{<em>}<em>{h}, t(s</em>{2}) \prec \theta^{</em>}<em>{h} \rangle \prec</em>{u} \langle s_{1}, s_{2} \rangle \land \langle (t(s_{1}) \prec \theta^{<em>}_{h}) \land \theta^{</em>'} = (t(s_{2}) \prec \theta^{<em>}<em>{h}) \land mgid(s</em>{1}, s_{2}, \theta^{</em>'}) \land \theta^{*'} \neq nil \rangle )</td>
<td>( nil )</td>
</tr>
</tbody>
</table>

7.2.3. The Generation of the If-Then-Else. The first branch of the combination case can now be derived by a resolution with goal G49:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Premise</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>G53</td>
<td>( B(s_{1}) \land B(s_{2}) \land \langle h(s_{1}), h(s_{2}) \rangle \prec_{u} \langle s_{1}, s_{2} \rangle \land \langle t(s_{1}) \prec \theta^{<em>}<em>{h}, t(s</em>{2}) \prec \theta^{</em>}<em>{h} \rangle \prec</em>{u} \langle s_{1}, s_{2} \rangle \land \langle (t(s_{1}) \prec \theta^{<em>}_{h}) \land \theta^{</em>'} = (t(s_{2}) \prec \theta^{<em>}<em>{h}) \land mgid(s</em>{1}, s_{2}, \theta^{</em>'}) \land \theta^{*'} \neq nil \rangle )</td>
<td>( nil )</td>
</tr>
<tr>
<td>G49</td>
<td>( B(s_{1}) \land B(s_{2}) \land h(s_{1}) \prec \theta^{<em>}<em>{h} = h(s</em>{2}) \prec \theta^{</em>}<em>{h} \neq nil \land \langle t(s</em>{1}) \prec \theta^{<em>}<em>{h}, t(s</em>{2}) \prec \theta^{</em>}<em>{h} \rangle \prec</em>{u} \langle s_{1}, s_{2} \rangle \land \langle (t(s_{1}) \prec \theta^{<em>}_{h}) \land \theta^{</em>'} = \sigma \neq (t(s_{2}) \prec \theta^{<em>}<em>{h}) \land mgid(s</em>{1}, s_{2}, \theta^{</em>'}) \land \theta^{*'} \neq nil \rangle )</td>
<td>( \theta^{<em>}_{h} \circ \theta^{</em>'} )</td>
</tr>
<tr>
<td></td>
<td>( \langle h(s_{1}), h(s_{2}) \rangle \prec_{u} \langle s_{1}, s_{2} \rangle \land \langle h(s_{1}) \prec \sigma \neq h(s_{2}) \prec \sigma \land mgid(s_{1}, s_{2}, \theta^{<em>'}) \land \theta^{</em>'} \neq nil \rangle )</td>
<td>( \theta^{<em>}_{h} \circ \theta^{</em>'} )</td>
</tr>
<tr>
<td>G54</td>
<td>( B(s_{1}) \land B(s_{2}) \land h(s_{1}) \prec \theta^{<em>}<em>{h} = h(s</em>{2}) \prec \theta^{</em>}<em>{h} \neq nil \land \langle t(s</em>{1}) \prec \theta^{<em>}<em>{h}, t(s</em>{2}) \prec \theta^{</em>}<em>{h} \rangle \prec</em>{u} \langle s_{1}, s_{2} \rangle \land \langle h(s_{1}), h(s_{2}) \rangle \prec_{u} \langle s_{1}, s_{2} \rangle \land \langle h(s_{1}) \prec \sigma \neq h(s_{2}) \prec \sigma \land mgid(s_{1}, s_{2}, \theta^{<em>'}) \land \theta^{</em>'} \neq nil \rangle )</td>
<td>( \text{if } \theta^{<em>'} = nil \text{ then } nil ) | ( \text{else } \theta^{</em>}_{h} \circ \theta^{*'} )</td>
</tr>
</tbody>
</table>
7.2.4. The Termination Conjunct. We now assume that the relation \( \prec_u \) holds for the tails, namely that we can prove the conjunct \( \langle t(s_1) \prec \theta^*_h, t(s_2) \prec \theta^*_h \rangle \prec_u \langle s_1, s_2 \rangle \). This part of the proof will be given in the following section. Here we continue the proof using goal G69. The only condition required in this part of the proof is that \( \theta^*_h \) must be a most general idempotent unifier of the heads. Therefore, goal G69 contains the conjunct expressing the mgid condition for the heads:

\[
\begin{align*}
G69 & \quad B(s_1) \land B(s_2) \land \\
& \quad h(s_1) \prec \theta^*_h = h(s_2) \prec \theta^*_h \land \\
& \quad \text{mgid}(h(s_1), h(s_2), \theta^*_h) \land \theta^*_h \neq \text{nil} \land \\
& \quad \langle h(s_1), h(s_2) \rangle \prec_u \langle s_1, s_2 \rangle \land \\
& \quad \neg(h(s_1) \prec \sigma \neq h(s_2) \prec \sigma \land \theta^*_h = \text{nil})
\end{align*}
\]

| \( A27 \) | \( \langle r_1, r_2 \rangle \prec_u \langle s_1, s_2 \rangle \) then \\
|---|---|
| \( r_1 \prec uf(r_1, r_2) = r_2 \prec uf(r_1, r_2) \land \\
| \text{mgid}(r_1, r_2, uf(r_1, r_2)) \land \text{uf}(r_1, r_2) \neq \text{nil} \land \text{uf}(r_1, r_2) = \text{nil} \land r_1 \prec \sigma \neq r_2 \prec \sigma |

7.3. Heads

We have proved all the conjuncts concerning the tails of the input, and generated the recursive call for the tails in the output. Our plan is now to prove goal G69 and generate the appropriate calls for the heads. Actually, part of this plan has already been accomplished because we have already made some assumption about the unifier of the heads. Therefore the conjuncts in goal G69 can now be easily proved by a resolution with the induction hypothesis:

\[
\begin{align*}
G69 & \quad B(s_1) \land B(s_2) \land \\
& \quad h(s_1) \prec \theta^*_h = h(s_2) \prec \theta^*_h \land \\
& \quad \text{mgid}(h(s_1), h(s_2), \theta^*_h) \land \theta^*_h \neq \text{nil} \land \\
& \quad \langle h(s_1), h(s_2) \rangle \prec_u \langle s_1, s_2 \rangle \land \\
& \quad \neg(h(s_1) \prec \sigma \neq h(s_2) \prec \sigma \land \theta^*_h = \text{nil})
\end{align*}
\]

\[
\begin{align*}
G70 & \quad B(s_1) \land B(s_2) \land \\
& \quad \langle h(s_1), h(s_2) \rangle \prec_u \langle s_1, s_2 \rangle \land \\
& \quad \neg(h(s_1) \prec \sigma \neq h(s_2) \prec \sigma \land \theta^*_h = \text{nil})
\end{align*}
\]

if \( \theta^*_t \neq \text{nil} \) then \( \text{nil} \) else \( \theta^*_h \land \theta^*_t \neq \text{nil} \)
7.3.1. The Generation of the If-Then-Else. We now generate the conditional branch for the head unifier:

G50

\[
\begin{align*}
B(s_1) \land B(s_2) \land \\
\langle h(s_1), h(s_2) \rangle <_u \langle s_1, s_2 \rangle \land \\
\lnot (h(s_1) < \theta^*_h \land h(s_2) < \theta^*_h) \\
\theta^*_h \circ \bar{\sigma}(\text{nil}) = \bar{\sigma}(\text{nil}) \land \theta^*_h \neq \text{nil}
\end{align*}
\]

\[\text{nil}\]

G70

\[
\begin{align*}
B(s_1) \land B(s_2) \land \\
\langle h(s_1), h(s_2) \rangle <_u \langle s_1, s_2 \rangle \land \\
\lnot (h(s_1) < \sigma \neq h(s_2) < \sigma) \\
\theta^*_h = \text{nil}
\end{align*}
\]

\[\text{if } \theta^*_t = \text{nil}\]

\[\text{then } \text{nil}\]

\[\text{else } \theta^*_h \circ \theta^*_t\]

G71

\[
\begin{align*}
B(s_1) \land B(s_2) \land \\
\langle h(s_1), h(s_2) \rangle <_u \langle s_1, s_2 \rangle \\
\lnot (h(s_1) < \sigma \neq h(s_2) < \sigma) \\
\theta^*_h \neq \text{nil}
\end{align*}
\]

\[\text{if } \theta^*_h = \text{nil}\]

\[\text{then } \text{nil}\]

\[\text{else if } \theta^*_t = \text{nil}\]

\[\text{then } \text{nil}\]

\[\text{else } \theta^*_h \circ \theta^*_t\]

The unabbreviated form of the output is the following:

\[
\begin{align*}
\text{if } uf(h(s_1), h(s_2)) = \text{nil} \\
\text{then } \text{nil} \\
\text{else if } uf(t(s_1) < uf(h(s_1), h(s_2)), t(s_2) < uf(h(s_1), h(s_2))) = \text{nil} \\
\text{then } \text{nil} \\
\text{else } uf(h(s_1), h(s_2)) \circ uf(t(s_1) < uf(h(s_1), h(s_2)), t(s_2) < uf(h(s_1), h(s_2))).
\end{align*}
\]

7.3.2. The Termination Conjunct. We finally assume that the relation \(<_u\) holds for the heads and get to the final goal for the combination case, G75 (the proof of the conjunct \(\langle h(s_1), h(s_2) \rangle <_u \langle s_1, s_2 \rangle\) is given in the next section):

G75

\[
\begin{align*}
\text{if } \theta^*_h = \text{nil} \\
\text{then } \text{nil} \\
\text{else if } \theta^*_t = \text{nil} \\
\text{then } \text{nil} \\
\text{else } \theta^*_h \circ \theta^*_t.
\end{align*}
\]

This concludes the proof for the combination case, except for the proof of the termination, namely the \(<_u\) conjuncts of goal G54 and G71, which is presented in the next section.

8. THE PROOF OF THE TERMINATION

The proof of the termination is similar to the one presented by Manna and Waldinger [6]. In fact, we use the same ordering condition, even though we have chosen a simpler data structure for expressions, which results in a more compact
proof. The presentation is divided into two subsections, the first one relative to the tail branch and the second one to the head branch. In order to keep the size of the tableaus manageable, we omit the output and the conjuncts that are not modified by this part of the proof.

Before discussing the proof steps, we present a few basic assertion about the domain and range properties of substitutions that are used in the proof. These assertions could be derived from the theory of substitutions using the deductive tableaus; their derivation, although nontrivial, is outside the scope of this paper.

**Assertion A28**

\[\text{if } e \triangleleft \sigma \neq e \text{ then } \bar{v} \in \text{us}(e) \land \bar{v} \in \text{dom}(\sigma)\]

Assertion A28 is obtained by the invariance corollary (see [6, p. 16]) and can be read as follows: if a substitution affects an expression, then there must be a variable belonging to both the domain of the substitution and the expression.

**Assertion A29**

\[\text{us}(e \triangleleft \sigma) \subseteq \text{us}(e) \cup \text{rg}(\sigma)\]

Assertion A29 expresses the variable-introduction proposition (see [6, p. 19]): after the application of a substitution, the variables of the resulting expression are a subset of those of the initial expression and those introduced by the substitution, i.e. the range of the substitution.

**Assertion A30**

\[\text{if } \text{id}(\sigma) \land v_1 \in \text{dom}(\sigma) \text{ then } v_1 \notin \text{us}(e_1 \triangleleft \sigma)\]

Assertion A30 is obtained from the basic property of idempotent substitutions (see [6, p. 24]), stating that the range and the domain of an idempotent substitution are disjoint. This property plays a crucial role in the proof of the termination, and for this reason it has been included in the initial specification. We use the symbol \(\text{id}\) to denote the predicate which is true when its argument is an idempotent substitution. This is of course verified by each most general idempotent unifier.

**Assertion A31**

\[\text{if } U(e_1, e_2, \theta) \text{ then } \text{rg}(\theta) \subseteq \text{us}(e_1) \cup \text{us}(e_2)\]

We introduce an abbreviation to express that \(\theta\) is the most general idempotent unifier of the two input expressions \(s_1\) and \(s_2\):

\[U(s_1, s_2, \theta) \equiv (s_1 \triangleleft \theta = s_2 \triangleleft \theta) \land \text{mgid}(s_1, s_2, \theta) \land \theta \neq \text{nil}\]

Assertion A31 expresses the range proposition for unifiers (see [6, p. 26]), stating that the only variables in the range of a most general idempotent unifier are those occurring in the expressions being unified.

The relation \(\triangleleft_u\) is defined by the formula

\[
\langle r_1, r_2 \rangle \triangleleft_u \langle s_1, s_2 \rangle \equiv \text{us}(r_1) \cup \text{us}(r_2) \subseteq \text{us}(s_1) \cup \text{us}(s_2) \land \\
(\text{us}(r_1) \cup \text{us}(r_2) \neq \text{us}(s_1) \cup \text{us}(s_2) \lor r_1 \triangleleft s_1)
\]

### 8.1. The Tail-Unifier Condition

Here we prove the tail-unifier condition, namely the conjunct \(\langle t(s_1) \triangleleft \theta_h^*, t(s_2) \triangleleft \theta_h^* \rangle \triangleleft_u \langle s_1, s_2 \rangle\) of goal G54. Using the formula for the \(\triangleleft_u\) relation, and omitting
the other conjuncts and the output, we obtain the following tableau:

| G55 | $us(t(s_1) < \theta^*_h) \cup us(t(s_2) < \theta^*_h) \subseteq us(s_1) \cup us(s_2)$ $\wedge$
|     | $(us(t(s_1) < \theta^*_h) \cup us(t(s_2) < \theta^*_h) \neq us(s_1) \cup us(s_2))$ $\vee$
|     | $t(s_1) < \theta^*_h \preceq s_1$ |

The proof is organized as follows: initially we analyze the first conjunct of goal G55, proving that the relation $\subseteq$ holds; then we prove the tableau in the case $r(s_1) = e^* = t(s_1)$, and finally in the case $t(s_1) < \theta^*_h \neq t(s_1)$.

To prove the $\subseteq$ relation we use the fact that the variables of a combination are given by the union of the variables of the head and the variables of the tail. The next step consists of two applications of the equality replacement rule:

| A18 | $us(b) = us(h(b)) \cup us(t(b))$

| G56 | $us(t(s_1) < \theta^*_h) \cup us(t(s_2) < \theta^*_h) \subseteq$
|     | $us(h(s_1)) \cup us(h(s_2)) \cup us(t(s_1)) \cup us(t(s_2))$ $\wedge$
|     | $(us(t(s_1) < \theta^*_h) \cup us(t(s_2) < \theta^*_h) \neq us(s_1) \cup us(s_2))$ $\vee$
|     | $t(s_1) < \theta^*_h \preceq s_1$ |

The idea behind the next step of the proof is that if set containment holds for a subset of the containing set, it must hold for the containing set as well. This kind of reasoning is allowed within the deductive tableau by means of inference rules specific to the considered relation. In this case, we apply the $\subseteq$-replacement rule between the current goal and the range property of substitutions. Relation-replacement rules are analogous to the equality replacement; they are all instances of a general rule defined for binary relations by Manna and Waldinger [8]:

| A31 | if $U(e_1, e_2, \theta)$ then $rg(\theta) \subseteq us(e_1) \cup us(e_2)$

| G56 | $us(t(s_1) < \theta^*_h) \cup us(t(s_2) < \theta^*_h) \subseteq$
|     | $us(h(s_1)) \cup us(h(s_2)) \cup us(t(s_1)) \cup us(t(s_2))$ $\wedge$
|     | $(us(t(s_1) < \theta^*_h) \cup us(t(s_2) < \theta^*_h) \neq us(s_1) \cup us(s_2))$ $\vee$
|     | $t(s_1) < \theta^*_h \preceq s_1$ |
We apply the \( \subseteq \)-replacement again, using the variable-introduction proposition:

\[
A29 \quad \text{\textbf{us\text{(e \sigma)}}} \quad \subseteq \text{\textbf{us\text{(e) \cup \text{rg\text{(e)}})}}
\]

The \( \subseteq \)-conjunct is finally proved by resolving goal G58 against the assertion stating the reflexivity of that relation. Notice that in this case we assume a commutative unification:

\[
A32 \quad x \subseteq x
\]

\[
G57 \quad U(h(s_1), h(s_2), \theta^*_n) \wedge \\
\text{\textbf{us\text{(t\text{(s_1) \wedge \theta^*_n})}}} \cup \text{\textbf{us\text{(t\text{(s_2) \wedge \theta^*_n})}}} \subseteq \text{\textbf{rg\text{(\theta^*_n)}}} \cup \text{\textbf{us\text{(t\text{(s_1))}}} \cup \text{\textbf{us\text{(t\text{(s_2))}}}}} \wedge \\
(\text{\textbf{us\text{(t\text{(s_1) \wedge \theta^*_n})}}} \cup \text{\textbf{us\text{(t\text{(s_2) \wedge \theta^*_n})}}} \neq \text{\textbf{us\text{(s_1) \cup us\text{(s_2)}}}}} \vee \\
t(s_1) \wedge \theta^*_n \subseteq s_1)
\]

\[
G58 \quad U(h(s_1), h(s_2), \theta^*_n) \wedge \\
\text{\textbf{us\text{(t\text{(s_1) \cup \text{rg\text{(\theta^*_n)}}}}} \cup \text{\textbf{us\text{(t\text{(s_2) \cup \text{rg\text{(\theta^*_n)}})}}}}} \subseteq \text{\textbf{rg\text{(\theta^*_n)}}} \cup \text{\textbf{us\text{(t\text{(s_1))}}} \cup \text{\textbf{us\text{(t\text{(s_2))}}}}} \wedge \\
(\text{\textbf{us\text{(t\text{(s_1) \wedge \theta^*_n})}}} \cup \text{\textbf{us\text{(t\text{(s_2) \wedge \theta^*_n})}}} \neq \text{\textbf{us\text{(s_1) \cup us\text{(s_2)}}}}} \vee \\
t(s_1) \wedge \theta^*_n \subseteq s_1)
\]

\[
G59 \quad U(h(s_1), h(s_2), \theta^*_n) \wedge \\
(\text{\textbf{us\text{(t\text{(s_1) \wedge \theta^*_n})}}} \cup \text{\textbf{us\text{(t\text{(s_2) \wedge \theta^*_n})}}} \neq \text{\textbf{us\text{(s_1) \cup us\text{(s_2)}}}}} \vee \\
t(s_1) \wedge \theta^*_n \subseteq s_1)
\]
We can now split the goal using the if-split rule and develop the two branches of the proof corresponding to \( t(s_1) \triangleq \theta^*_h = t(s_1) \) and \( t(s_1) \triangleq \theta^*_h \neq t(s_1) \), respectively:

| \( \text{G60} \) | \( U(h(s_1), h(s_2), \theta^*_h) \land (t(s_1) \triangleq \theta^*_h) \triangleq s_1 \) |
| \( \text{G61} \) | \( U(h(s_1), h(s_2), \theta^*_h) \land (vs(t(s_1) \triangleq \theta^*_h) \cup vs(t(s_2) \triangleq \theta^*_h) \neq vs(s_1) \cup vs(s_2) \) |

Goal G60 is easily verified if the head unifier does not modify the tail of the expression. The rule applied is resolution with equality matching:

| \( \text{A7} \) | \( t(b) \triangleq b \) |
| \( \text{G60} \) | \( U(h(s_1), h(s_2), \theta^*_h) \land (t(s_1) \triangleq \theta^*_h) \triangleq s_1 \) |
| \( \text{G62} \) | \( U(h(s_1), h(s_2), \theta^*_h) \land t(s_1) \triangleq \theta^*_h = t(s_1) \) |

We now transform goal G61 by a resolution step with the assertions concerning set union and membership:

| \( \text{A33} \) | if \( u \in x_1 \land u \not\in x_2 \) then \( x_1 \neq x_2 \) |
| \( \text{G61} \) | \( U(h(s_1), h(s_2), \theta^*_h) \land vs(t(s_1) \triangleq \theta^*_h) \cup vs(t(s_2) \triangleq \theta^*_h) \neq vs(s_1) \cup vs(s_2) \) |
| \( \text{G63} \) | \( U(h(s_1), h(s_2), \theta^*_h) \land v \not\in vs(t(s_1) \triangleq \theta^*_h) \cup vs(t(s_2) \triangleq \theta^*_h) \land v \in vs(s_1) \cup vs(s_2) \) |

Below the resolution rule is applied twice:

| \( \text{A34} \) | \( v_1 \in x_1 \lor v_1 \in x_2 \Rightarrow v_1 \in x_1 \cup x_2 \) |
| \( \text{G63} \) | \( U(h(s_1), h(s_2), \theta^*_h) \land v \not\in vs(t(s_1) \triangleq \theta^*_h) \cup vs(t(s_2) \triangleq \theta^*_h) \land v \in vs(s_1) \cup vs(s_2) \) |
| \( \text{G64} \) | \( U(h(s_1), h(s_2), \theta^*_h) \land v \not\in vs(t(s_1) \triangleq \theta^*_h) \land v \not\in vs(t(s_2) \triangleq \theta^*_h) \land (v \in vs(s_1) \lor v \in vs(s_2)) \) |

At this point, the idempotence condition is essential: in fact we can resolve the current goal against assertion A30, stating that if \( \sigma \) is an idempotent substitution and a variable \( v_1 \) is in its domain, \( v_1 \) does not occur in the result of the application
The idempotence condition has been incorporated in the \textit{mgid} conjunct of $U(h(s_1), h(s_2), \theta_h^*)$.

Because $s_1$ is a combination, we obtain, by a resolution step, that $v$ occurs either in the head or in the tail:

\begin{align*}
A19 \quad & \text{if } v \in vs(h(b)) \lor v \in vs(t(b)) \text{ then } v \in vs(b) \\
G65 \quad & U(h(s_1), h(s_2), \theta_h^*) \land v \in dom(\theta_h^*) \land (v \in vs(s_1) \lor v \in vs(s_2)) \\
G66 \quad & U(h(s_1), h(s_2), \theta_h^*) \land v \in dom(\theta_h^*) \land (v \in vs(h(s_1)) \lor v \in vs(t(s_1)) \lor v \in vs(s_2)) \\
G67 \quad & U(h(s_1), h(s_2), \theta_h^*) \land t(s_1) < \theta_h^* \neq t(s_1) \\
\end{align*}

With this new goal we can transfer the condition on sets in terms of equality of expressions. The rule applied is resolution:

\begin{align*}
A28 \quad & \text{if } e < \sigma = e \text{ then } \overline{v} \in vs(e) \land \overline{v} \in dom(\sigma) \\
G66 \quad & U(h(s_1), h(s_2), \theta_h^*) \land v \in dom(\theta_h^*) \land (v \in vs(h(s_1)) \lor v \in vs(t(s_1)) \lor v \in vs(s_2)) \\
G67 \quad & U(h(s_1), h(s_2), \theta_h^*) \land t(s_1) < \theta_h^* \neq t(s_1) \\
\end{align*}

The two goals G62 and G67 are trivially resolved:

\begin{align*}
G62 \quad & U(h(s_1), h(s_2), \theta_h^*) \land t(s_1) < \theta_h^* = t(s_1) \\
G67 \quad & U(h(s_1), h(s_2), \theta_h^*) \land t(s_1) < \theta_h^* \neq t(s_1) \\
G68 \quad & U(h(s_1), h(s_2), \theta_h^*) \\
\end{align*}
The conjunct in goal G68 must now be added to the other conjuncts omitted so far, thus yielding goal G69, as introduced in Section 7.2.4:

\[
\begin{array}{|c|c|}
\hline
\text{G69} & B(s_1) \land B(s_2) \land \\
& h(s_1) \triangleq \theta_h^* = h(s_2) \triangleq \theta_h^* \land \\
& mgid(h(s_1), h(s_2), \theta_h^*) \land \theta_h^* \neq \text{nil} \land \\
& \langle h(s_1), h(s_2) \rangle \prec_u \langle s_1, s_2 \rangle \land \\
& -(h(s_1) \sigma \neq h(s_2) \sigma \land \theta_h^* = \text{nil}) \\
\hline
\end{array}
\]

if $\theta_h^* = \text{nil}$
then \text{nil}
else $\theta_h^* \land \theta_h^*'$

8.2. The Head-Unifier Condition

Here we prove the head-unifier condition, namely the conjunct $\langle h(s_1), h(s_2) \rangle \prec_u \langle s_1, s_2 \rangle$ of goal G71. Omitting the irrelevant conjuncts and the output, and using the formula for the $\prec_u$ relation, we obtain the following tableau:

\[
\begin{array}{|c|c|}
\hline
\text{G72} & vs(h(s_1)) \cup vs(h(s_2)) \subseteq vs(s_1) \cup vs(s_2) \land \\
& \left( vs(h(s_1)) \cup vs(h(s_2)) \neq vs(s_1) \cup vs(s_2) \lor h(s_1) \preceq s_1 \right) \\
\hline
\end{array}
\]

We first apply the resolution rule against the fact that the head of a combination always occurs in the combination:

\[
\begin{array}{|c|c|}
\hline
\text{A6} & h(b) \prec b \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{G72} & vs(h(s_1)) \cup vs(h(s_2)) \subseteq vs(s_1) \cup vs(s_2) \land \\
& \left( vs(h(s_1)) \cup vs(h(s_2)) \neq vs(s_1) \cup vs(s_2) \lor h(s_1) \preceq s_1 \right) \\
\hline
\text{G73} & vs(h(s_1)) \cup vs(h(s_2)) \subseteq vs(s_1) \cup vs(s_2) \\
\hline
\end{array}
\]

Then we split the problem for $s_1$ and $s_2$ using the $\preceq$-replacement rule:

\[
\begin{array}{|c|c|}
\hline
\text{A32} & x \preceq x \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{G73} & vs(h(s_1)) \cup vs(h(s_2)) \subseteq vs(s_1) \cup vs(s_2) \\
\hline
\text{G74} & vs(h(s_1)) \subseteq vs(s_1) \land vs(h(s_2)) \subseteq vs(s_2) \\
\hline
\end{array}
\]

This can finally be resolved using the fact that the variables of a combination include the variables of its head, giving the goal true:

\[
\begin{array}{|c|c|}
\hline
\text{A17} & vs(h(b)) \subseteq vs(b) \\
\hline
\end{array}
\]
The final goal obtained by the proof of the combination case is then goal G75, as introduced in Section 7.3.2.

\[
\text{G75} \quad \begin{array}{c}
B(s_1) \land B(s_2)
\end{array}
\]

if \( \theta^* = \text{nil} \)
then \( \text{nil} \)
else if \( \theta'' = \text{nil} \)
then \( \text{nil} \)
else \( \theta^* \circ \theta'' \)

This concludes the proof of the termination of the algorithm.

9. SYNTHESIS OF THE ALGORITHM

In order to build the algorithm, we have to take into account the clash due to different syntactic categories and then combine all the results derived by the case analysis into a single program.

9.1. Distinct Syntactic Categories

The clash for expressions of distinct syntactic categories is limited to the case in which we have a combination and a constant. The derivation is straightforward. The first step is an application of the resolution rule:

\[
\text{A3} \quad \text{if } B(b) \land C(c) \text{ then } b \neq c
\]

\[
\text{G4} \quad \begin{array}{c}
s_1 \triangleq \tilde{\sigma}(\text{nil}) \neq s_2 \triangleq \tilde{\sigma}(\text{nil})
\end{array}
\]

\[
\text{G76} \quad \begin{array}{c}
B(s_1 \triangleq \tilde{\sigma}(\text{nil})) \land C(s_2 \triangleq \tilde{\sigma}(\text{nil}))
\end{array}
\]

nil

The next two steps are applications of the equality replacement rule; in the first one we implicitly assume that \( B(e_1 \circ e_2) \) holds.

\[
\text{A13} \quad \text{if } B(b) \text{ then } b \triangleq \sigma = (h(b) \triangleq \sigma) \circ (\tau(b) \triangleq \sigma)
\]

\[
\text{G76} \quad \begin{array}{c}
B\left(s_1 \triangleq \tilde{\sigma}(\text{nil})\right) \land C(s_2 \triangleq \tilde{\sigma}(\text{nil}))
\end{array}
\]

nil

\[
\text{G77} \quad B(s_1 \land C(s_2 \triangleq \tilde{\sigma}(\text{nil})))
\]

nil
9.2. The Case Selection

The case selection consists of a few steps that allow to derive the final program: we use the initial assertion stating that the inputs of the unification program are two expressions and the goals obtained for the different cases. The results achieved so far are summarized by the following tableau:

<table>
<thead>
<tr>
<th>Case</th>
<th>Goal</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>G78</td>
<td>$B(s_1) \land C(s_2)$</td>
<td>$nil$</td>
</tr>
<tr>
<td>G79</td>
<td>$C(s_1) \land B(s_2)$</td>
<td>$nil$</td>
</tr>
<tr>
<td>G12</td>
<td>$C(s_1) \land C(s_2)$</td>
<td>if $s_1 = s_2$ then ${}$ else $nil$</td>
</tr>
<tr>
<td>G29</td>
<td>$V(s_1)$</td>
<td>if $s_1 \neq s_2$ then ${ s_1 \leftarrow s_2 }$ else $nil$</td>
</tr>
<tr>
<td>G80</td>
<td>$V(s_2)$</td>
<td>if $s_2 \neq s_1$ then ${ s_2 \leftarrow s_1 }$ else $nil$</td>
</tr>
<tr>
<td>G75</td>
<td>$B(s_1) \land B(s_2)$</td>
<td>if $\theta^* = nil$ then $nil$ else if $\theta'<em>* = nil$ then $nil$ else $\theta^*</em>* \circ \theta'_*$</td>
</tr>
</tbody>
</table>

In this tableau we included the goals G79 and G80, which can be obtained by derivations analogous to those leading to goal G78 and G29, respectively.

We initially use the decomposition property of expressions and resolve twice the goal relative to the combination case against this assertion:

<table>
<thead>
<tr>
<th>Case</th>
<th>Goal</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>if $E(e)$ then $C(e) \lor V(e) \lor B(e)$</td>
<td></td>
</tr>
<tr>
<td>G75</td>
<td>$\Box(s_1) \land B(s_2)$</td>
<td>if $\theta^<em><em>h = nil$ then $nil$ else if $\theta'</em></em> = nil$ then $nil$ else $\theta^<em>_</em> \circ \theta'_*$</td>
</tr>
<tr>
<td>G81</td>
<td>$E(s_1) \land E(s_2) \land \neg V(s_1) \land \neg V(s_2) \land \neg C(s_1) \land \neg C(s_2)$</td>
<td>if $\theta^<em><em>h = nil$ then $nil$ else if $\theta'</em></em> = nil$ then $nil$ else $\theta^<em>_</em> \circ \theta'_*$</td>
</tr>
</tbody>
</table>
We then resolve the current goal against the goal relative to clash due to distinct syntactic categories, obtaining a conditional in the output:

<table>
<thead>
<tr>
<th></th>
<th>( B(s_1) \land C(s_2) )</th>
<th>nil</th>
</tr>
</thead>
<tbody>
<tr>
<td>G78</td>
<td>( E(s_1) \land E(s_2) \land \neg V(s_1) \land \neg V(s_2) \land \neg C(s_1) \land \neg C(s_2) )</td>
<td>if ( \theta_h^* = \text{nil} ) then nil else if ( \theta_i^{<strong>} = \text{nil} ) then nil else ( \theta_h^* \land \theta_i^{</strong>} )</td>
</tr>
<tr>
<td>G81</td>
<td>( E(s_1) \land E(s_2) \land \neg V(s_1) \land \neg V(s_2) \land \neg C(s_1) \land B(s_1) )</td>
<td>if ( C(s_2) ) then nil else if ( \theta_i^{<strong>} = \text{nil} ) then nil else ( \theta_h^* \land \theta_i^{</strong>} )</td>
</tr>
<tr>
<td>G82</td>
<td>( E(s_1) \land E(s_2) \land \neg V(s_1) \land \neg C(s_1) \land B(s_1) )</td>
<td>if ( C(s_2) ) then nil else if ( \theta_i^{<strong>} = \text{nil} ) then nil else ( \theta_h^* \land \theta_i^{</strong>} )</td>
</tr>
</tbody>
</table>

The next step is a resolution with the assertion stating that a combination can be neither a constant nor a variable:

| A2  | if \( B(e_1) \) then \( \neg C(e_1) \land \neg V(e_1) \) | nil |
| G82 | \( E(s_1) \land E(s_2) \land \neg V(s_1) \land \neg C(s_1) \land B(s_1) \) | if \( C(s_2) \) then nil else if \( \theta_h^* = \text{nil} \) then nil else if \( \theta_i^{**} = \text{nil} \) then nil else \( \theta_h^* \land \theta_i^{**} \) |
| G83 | \( E(s_1) \land E(s_2) \land \neg V(s_2) \land B(s_1) \) | if \( C(s_2) \) then nil else if \( \theta_h^* = \text{nil} \) then nil else if \( \theta_i^{**} = \text{nil} \) then nil else \( \theta_h^* \land \theta_i^{**} \) |

We now develop another branch of the proof starting from the constant case:

| A1  | if \( E(e) \) then \( C(e) \lor V(e) \lor B(e) \) | nil |
| G12 | \( C(s_1) \land C(s_2) \) | if \( s_1 = s_2 \) then \{ \} else nil |
| G84 | \( C(s_1) \land E(s_2) \land \neg V(s_2) \land \neg B(s_2) \) | if \( s_1 = s_2 \) then \{ \} else nil |
The next resolution step generates a conditional in the output:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G79</td>
<td>( C(s_1) \land B(s_2) )</td>
<td>nil</td>
</tr>
<tr>
<td>G84</td>
<td>( C(s_1) \land E(s_2) \land \neg V(s_2) \land \neg B(s_2) )</td>
<td>if ( s_1 = s_2 ) then { } else nil</td>
</tr>
<tr>
<td>G85</td>
<td>( C(s_1) \land E(s_2) \land \neg V(s_2) )</td>
<td>if ( B(s_2) ) then nil else if ( s_1 = s_2 ) then { } else nil</td>
</tr>
<tr>
<td>A1</td>
<td>if ( E(e) ) then ( C(e) \lor V(e) \lor B(e) )</td>
<td></td>
</tr>
<tr>
<td>G85</td>
<td>( C(s_1) \land E(s_2) \land \neg V(s_2) )</td>
<td>if ( B(s_2) ) then nil else if ( s_1 = s_2 ) then { } else nil</td>
</tr>
<tr>
<td>G86</td>
<td>( E(s_1) \land E(s_2) \land \neg V(s_1) \land \neg V(s_2) \land \neg B(s_1) )</td>
<td>if ( B(s_2) ) then nil else if ( s_1 = s_2 ) then { } else nil</td>
</tr>
</tbody>
</table>

Here we recombine the two branches of the proof, thus generating another conditional in the output:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>G83</td>
<td>( E(s_1) \land E(s_2) \land \neg V(s_2) \land B(s_1) )</td>
<td>if ( C(s_2) ) then nil else if ( \theta^<em>_h = \text{nil} ) then nil else if ( \theta^</em> = \text{nil} ) then nil else ( \theta^<em>_h \land \theta^</em> )</td>
</tr>
<tr>
<td>G86</td>
<td>( E(s_1) \land E(s_2) \land \neg V(s_1) \land \neg V(s_2) \land \neg B(s_1) )</td>
<td>if ( B(s_2) ) then nil else if ( s_1 = s_2 ) then { } else nil</td>
</tr>
<tr>
<td>G87</td>
<td>( E(s_1) \land E(s_2) \land \neg V(s_1) \land \neg V(s_2) )</td>
<td>if ( B(s_2) ) then nil else if ( C(s_2) ) then nil else if ( \theta^<em>_h = \text{nil} ) then nil else if ( \theta^</em>_ = \text{nil} ) then nil else ( \theta^<em>_h \land \theta^</em> ) else if ( B(s_2) ) then nil else if ( s_1 = s_2 ) then { } else nil</td>
</tr>
</tbody>
</table>
We now can resolve with assertion A23, representing the hypothesis on the input of the program, and eliminate the conjuncts asserting that \( s_1 \) and \( s_2 \) are expressions:

\[
A23 \quad E(s_1) \land E(s_2)
\]

| \(G87\) | \(E(s_1) \land E(s_2) \land \neg V(s_1) \land \neg V(s_2)\) | if \(B(s_1)\)
|---|---|---|
| | | then if \(C(s_2)\) then \(\text{nil}\)
| | | else if \(\theta_h^* = \text{nil}\) then \(\text{nil}\)
| | | else if \(\theta_{i'}^* = \text{nil}\) then \(\text{nil}\)
| | | else \(\theta_h^* \circ \theta_{i'}^*\)
| | | else if \(B(s_2)\) then \(\text{nil}\)
| | | else if \(s_1 = s_2\) then \{\} else \(\text{nil}\) |

| \(G88\) | \(\neg V(s_1) \land \neg V(s_2)\) | if \(B(s_1)\)
|---|---|---|
| | | then if \(C(s_2)\) then \(\text{nil}\)
| | | else if \(\theta_h^* = \text{nil}\) then \(\text{nil}\)
| | | else if \(\theta_{i'}^* = \text{nil}\) then \(\text{nil}\)
| | | else \(\theta_h^* \circ \theta_{i'}^*\)
| | | else if \(B(s_2)\) then \(\text{nil}\)
| | | else if \(s_1 = s_2\) then \{\} else \(\text{nil}\)

The last two steps of the proof consist of two resolutions of this goal against the goals relative to the variable case: the first one for the case of \( s_1 \) variable with goal G29, and the second one for the case of \( s_2 \) variable with goal G80. The final goal true is then obtained with the following output:

if \(V(s_1)\)
then if \(s_1 \not\equiv s_2\) then \{s_1 \leftarrow s_2\} else \(\text{nil}\)
else if \(V(s_2)\)
then if \(s_2 \not\equiv s_1\) then \{s_2 \leftarrow s_1\} else \(\text{nil}\)
else if \(B(s_1)\)
then if \(C(s_2)\) then \(\text{nil}\)
else if \(uf(h(s_1), h(s_2)) = \text{nil}\) then \(\text{nil}\)
else if \(uf(t(s_1) \land uf(h(s_1), h(s_2)),
\quad t(s_2) \land uf(h(s_1), h(s_2))) = \text{nil}\)
then \(\text{nil}\)
else \(uf(h(s_1), h(s_2))
\quad \circ uf(t(s_1) \land uf(h(s_1), h(s_2)),
\quad t(s_2) \land uf(h(s_1), h(s_2)))
else if \(B(s_2)\) then \(\text{nil}\) else if \(s_1 = s_2\) then \{\} else \(\text{nil}\)

This concludes the proof.
10. RELATED WORK AND CONCLUSIONS

We have presented a formal synthesis of a unification algorithm by the deductive-tableau method. The size of both the derivation and the output program shows the effectiveness of the deductive tableaus for real-life applications.

We started from the work of Manna and Waldinger [6]: they provide the background for the theory of substitutions, and present an informal derivation which has been a guideline for the formal proof discussed here.

A proof of the correctness of the unification algorithm derived in Manna and Waldinger's paper [6] has been obtained by Paulson using the interactive theorem prover LCF [9]. His proof, from which we have borrowed the data structure for expressions, must be regarded as a verification of a given algorithm rather than a synthesis.

A derivation of a unification program has been obtained by Eriksson [1], within the framework of the logic programming calculus developed by Hansson and Tärrlund. His basic theory is slightly different from ours because he uses the standard definition of expressions. This entails a larger number of cases in the derived program, expressed by a set of extended Horn clauses and obtained by case analysis on the structure of the expressions. Furthermore, instead of having a symbol to explicitly denote failure (i.e. nil), this is reported as a failure of the Horn-clause interpreter.

Eriksson’s derivation uses natural deduction rules, and his proof, which does not establish the termination of the algorithm, requires about 2500 steps. The deductive-tableau method, providing more powerful and high-level deduction rules, has allowed us to construct a much more compact proof (less than 200 steps).

On the other hand, our synthesis posed several problems. This is in part due to the fact that, when using a resolution based system, it is not always obvious how to discover the next useful resolution step; in particular, in the proof of the combination case we could not always follow the informal reasoning of Manna and Waldinger [6]. Sometimes the proof strategy has benefited from the intuition of the expected output, and the proof steps have been suggested by the transformations on the output.

Our derivation has been obtained by hand, but it could have been constructed on an interactive system based on the deductive-tableau method. We believe that the best way to use the deductive tableaus for program synthesis is with the help of an interactive system which allows one to browse among the applicable rules at each step of the proof. In this way the user may avoid the difficulties in the application of some resolution steps, and follow different lines of reasoning, possibly leading to the discovery of nonintuitive programs, such as the binary-search program derived by Manna and Waldinger [7].

The proof presented here seems beyond the capabilities of current automatic theorem provers. We plan to continue our work, in particular by identifying planning strategies to decompose the proof into smaller parts, which could possibly be carried out automatically.

I wish to thank Richard Waldinger, who introduced me to deductive tableaus and patiently reviewed the proof at different stages of its development. I am grateful to the referees for their comments and their careful reading of the manuscript.
REFERENCES


