On the Equivalence Problem for Attribute Systems

BRUNO COURCELLE AND PAUL FRANCHI-ZANNETTACCI

Université de Bordeaux I, U.E.R. de Mathématiques et Informatique,
351, Cours de la Libération 33405 Talence, France

The equivalence problem for strongly noncircular attribute systems reduces to the equivalence problem for primitive recursive schemes with parameters. We solve the equivalence problem for non-nested separated primitive recursive program schemes hence the equivalence problem for non-nested separated attribute systems.

INTRODUCTION

The authors have shown in Courcelle et al. (1982) that certain attribute systems called strongly noncircular can be translated into recursive program schemes taking derivation trees as arguments and called primitive recursive schemes with parameters. In particular, the equivalence problem for strongly noncircular attribute systems reduces to the equivalence problem for the latter class of program schemes.

This latter problem seems deeply related with the DPDA equivalence problem. We solve it in the special case of separated and non-nested primitive recursive schemes hence we obtain the decidability of the equivalence problem for separated and non-nested attribute systems (they are necessarily strongly noncircular).

Our proof generalizes the decidability of the equivalence problem for purely synthesized attribute systems solved in Courcelle et al. (1982). As a corollary we obtain the decidability of the equivalence problem for non-nested recursive schemes shown to be decidable in Courcelle (1978) with help of a chain of reductions to the equivalence problem for finite-turn DPDAs shown to be decidable by Valiant (1974).

The present proof does not use any reduction to finite-turn DPDAs. But it uses the formalism of decision systems introduced in Courcelle (1983).

1. PRELIMINARIES

We recall from Courcelle et al. (1982) all the necessary definitions. The reader familiar with this paper can skip Section 1.1–1.3.
1.1. Terms and Trees

Let $\mathcal{S}$ be a set of sorts. An $\mathcal{S}$-sorted signature (or simply an $\mathcal{S}$-signature) is a set $P$ (of function symbols) given with two mappings,

$$\alpha: P \to \mathcal{S}^*$$

and

$$\sigma: P \to \mathcal{S}$$

($\alpha(p)$ is called the arity of $p$ in $P$),

$$(\alpha(p)$$

is called the sort of $p$ in $P$).

The length of $\alpha(p)$ is called the rank of $p$ and is denoted by $\rho(p)$.

If $\alpha(p) = \varepsilon$ (we denote by $\varepsilon$ the empty word of any free monoid), then we say that $p$ is a constant or a variable (variables and constants will only be distinguished by their use for substitutions). Now let $P$ be an $\mathcal{S}$-signature. We define a heterogeneous $P$-magma as an object

$$M = \langle \langle M_s \rangle_{s \in \mathcal{S}}, P_M : p \in P \rangle,$$

where $M_s$ is a set, the carrier of sort $s$, and $P_M$ a total mapping

$$M_{s_1} \times \cdots \times M_{s_n} \to M_s, \quad \text{where} \quad \alpha(p) = s_1 \cdots s_n \quad \text{and} \quad \sigma(p) = s.$$

Let $X$ be an $\mathcal{S}$-sorted set of variables and $M(P, X)$ denote the free heterogeneous $P$-magma generated by $X$, consisting of well-formed terms written with $P$ and $X$.

Terms written with the set $P$ of function symbols (and possibly other variables) will be called $P$-terms. The words tree and term will be synonymous in this paper, and will refer to elements of some free magma.

We shall denote by $M(P, X)_s$ the carrier of sort $s$ of $M(P, X)$ and similarly for $M(P, X)$.

A term $t$ in $M(P, X)_s$ is thought of as denoting a value $t_M$ in any $P$-magma $M$. In fact $t_M = \text{eval}_M(t)$ where eval$_M$ denotes the unique homomorphism $M(P) \to M$. Similarly a term $t$ in $M(P, \{x_1, \ldots, x_k\})_s$ is thought of as denoting a function $M_{\sigma(x_1)} \times \cdots \times M_{\sigma(x_k)} \to M_s$ called a derived operator and denoted by $t_M$ or dero$_P(t)$ when we want to emphasize the existence of a unique homomorphism $\text{dero}_P_M: M(P, \{x_1, \ldots, x_k\})_s \to [M_{\sigma(x_1)} \times \cdots \times M_{\sigma(x_k)} \to M_s]$; we denote by $[D \to D']$ the set of total mappings from $D$ to $D'$ for sets $D$ and $D'$.

If $t$ is in $M(P, X)$, we denote by $\text{Var}_X(t)$ the set of variables of $X$ having at least one occurrence in $t$. We say that $t$ is $X$-linear if $|t|_x \leq 1$ for all $x$ in $X$.

We shall now use variables to define substitutions. Let $t$ be in $M(P, X)$ and $x_1, \ldots, x_k$ be distinct variables in $X$. Let $t_1, \ldots, t_k$ be in $M(P, X)$ and $\sigma(t_i) = \sigma(x_i)$ for $i = 1, \ldots, k$. We denote by $t[t_1/x_1, \ldots, t_k/x_k]$ the result of the simultaneous substitution of $t_i$ for each occurrence of $x_i$ in $t$ for all $i$ in $[k]$. We shall sometimes abbreviate $t[t_1/x_1, \ldots, t_k/x_k]$ into $t[t_1, \ldots, t_k]$.

Let $T$ be a subset of $M(P, X)$. We shall denote by $P(T)$ the set of terms the
form \( p(t_1, \ldots, t_n) \) for some \( p \) in \( P \) and some \( t_1, \ldots, t_n \) in \( T \) (such that \( p(t_1, \ldots, t_n) \in M(P, X) \), i.e., such that \( \sigma(t_1) \cdots \sigma(t_n) = a(p) \)). We shall denote by \( M(P, T) \) (with a slight ambiguity) the set of terms of the form

\[
s[t_1/v_1, \ldots, t_n/v_n]
\]

for some \( s \) in \( M(P, \{v_1, \ldots, v_n\}) \) linear in \( \{v_1, \ldots, v_n\} \) and some elements \( t_1, \ldots, t_n \) of \( T \) of proper sort. Equivalently "let \( u \) be in \( M(P, T) \)" will also be phrased "let \( u \) be of the form

\[
s[t_1, \ldots, t_n]
\]

for some linear \( P \)-term \( s \) for some \( t_1, \ldots, t_n \) in \( T \)." This latter formulation avoids an explicit mention of the (irrelevant) variables \( v_1, \ldots, v_n \). Note that \( T \subseteq M(P, T) \subseteq M(P, X) \).

1.2. Attribute Grammars

An attribute grammar is a triple \( \langle G, \Gamma, D \rangle \) consisting of:

1. a context-free grammar \( G \) with set \( \mathcal{N} \) of nonterminals and set \( P \) of production rules considered as an \( \mathcal{N} \)-sorted signature;

2. an attribute system \( \Gamma \) of type \( (P, F) \) (defined below) for some \( \mathcal{O} \)-sorted signature \( F \);

3. an interpretation \( D \), i.e., an \( F \)-magma.

An attribute system \( \Gamma \) of type \( (P, F) \) (where \( P \) and \( F \) are \( \mathcal{N} \) and \( \mathcal{O} \)-sorted signatures, respectively) consists of the following items:

(i) A finite set \( A \) of symbols called attributes, which is the disjoint union of \( A^{(s)} \), the set of synthesized attributes, and \( A^{(h)} \) the set of inherited attributes. Each attribute \( a \) has a sort \( \sigma(a) \) in \( \mathcal{O} \). For each \( a \) in \( A \), a subset \( \mathcal{N}_a \) of \( \mathcal{N} \) is given. For each \( S \) in \( \mathcal{N} \) we denote by \( A^{(s)}_S \) the set \( \{a \in A^{(s)} \mid S \in \mathcal{N}_a\} \) and similarly for \( A^{(h)}_S \) and \( A_S \).

(ii) For each \( p \) in \( P \), a set \( \Gamma_p \) of semantic rules in the form of a set of equations satisfying the following conditions:

(a) for all \( a \) in \( A^{(s)}_{\sigma(p)} \) there exist in \( \Gamma_p \) one and only one semantic rule defining \( a(\varepsilon) \), i.e., with left-hand side \( a(\varepsilon) \); it is of the form

\[
a(\varepsilon) = s[\ldots, z^{(i)}(\varepsilon), \ldots, b^{(j)}(l(j)), \ldots],
\]

where

for all \( i = 1, \ldots, i_0 \), \( z^{(i)} \) is an inherited attribute and \( z^{(i)} \in A^{(h)}_{\sigma(p)} \),

for all \( j = 1, \ldots, j_0 \), \( b^{(j)} \) is a synthesized attribute, \( l(j) \in [n] \)

and \( b^{(j)} \in A^{(s)}_{S_{\sigma(p)}} \) (where we assume that \( \alpha(p) = S_1, S_2 \ldots S_n \)).
s is an element of $M(F, \{v_1, \ldots, v_{i_0+j_0}\})$, $v_i$ is a variable of
sort $\sigma(z^{(i)})$ if $1 \leq i \leq i_0$ and of sort $\sigma(b^{(i-i_0)})$ if $i_0 + 1 \leq i \leq i_0 + j_0$.
And $s[t_1, \ldots, t_{i_0+j_0}]$ denotes $s[t_1/v_1, \ldots, t_{i_0+j_0}/v_{i_0+j_0}]$.

$(\beta)$ for all $k$ in $[n]$ and all $y$ in $A^{(h)}_S$ there exists one and only one
semantic rule in $I_p$ defining $y(k)$; it is of the form
$$y(k) = s[\ldots, z^{(i)}(e), \ldots, b^{(i)}(l(f)), \ldots],$$
where $s, z^{(i)}, b^{(i)}$ are exactly as in (1.2.1). An example is given in Section 1.5.

Convention. In the sequel we shall use letters $\{a, b, c, a', \ldots\}$ to denote
attributes of both kind and we shall reserve letters $\{y, z, z', \ldots\}$ to denote
inherited attributes.

1.3. Semantics of Attribute Grammars

Derivation trees of $G$ correspond to elements of $M(P)$. If the attribute
system $I'$ is noncircular Knuth (1968), Courcelle et al. (1982) every attribute
a has a value at every node $u$ of a tree $t$ in $M(P)$ provided $a$ belongs to $A_S$, 
where $S$ is the left part of the production rule labeling the node $u$. This value
belongs to $D_{\sigma(a)}$. (It depends also on values given to the inherited attributes
at the root of $t$.) We do not recall here how it can be computed Knuth
(1968), Courcelle et al. (1982).

In this paper we concentrate our attention on the functions
$$\varphi^{(D)}_{a,S} : M(P)_S \rightarrow [D_{\sigma(y_1)} \times \cdots \times D_{\sigma(y_k)} \rightarrow D_{\sigma(a)}]$$
such that $(y_1, \ldots, y_k)$ is a fixed enumeration of the set $A^{(h)}_S$ and
$\varphi^{(D)}_{a,S}(t)(d_1, \ldots, d_k)$ is the value of the synthesized attribute $a$ at the root of the
tree $t$, when the values $d_1, \ldots, d_k$ are given to the inherited attributes
$y_1, y_2, \ldots, y_k$ at the root of $t$.

We can also choose the free interpretation $M = M(F, A^{(h)})$ and letting
$$\varphi^{(F)}_{a,S} = \lambda t\varphi^{(M)}_{a,S}(t)(y_1, \ldots, y_k)$$
we get the factorization
$$\varphi^{(D)}_{a,S} = \text{derop}_D \circ \varphi^{(F)}_{a,S}$$
which is a fundamental result in the theory of program schemes (Engelfriet,
1980).

In this paper we shall investigate the equivalence problem for attribute
systems that can be precisely formulated as follows:
Let \( \Gamma \) and \( \Gamma' \) be two attribute systems of type \((P, F)\), let \( S \in \mathcal{N} \), \( a \in A_s^{(s)} \), and \( a' \in A_s^{(s)} \) such that \( \sigma(a) = \sigma(a') \). Let us assume that \( A_s^{(h)} \) is ordered in a fixed way as a sequence \( y_1, \ldots, y_k \). Let us also assume the same for \( A_s^{(h)} \) ordered as a sequence \( y'_1, \ldots, y'_k \) of same length, such that \( \sigma(y_i) = \sigma(y'_i) \) for all \( i \in [k] \).

We shall say that \((\Gamma, a)\) and \((\Gamma', a')\) are equivalent in \( D \) if \( \varphi_{a,S}^{(D)} \) and \( \varphi_{a',S}^{(D)} \) are the same functions \( M(P)_S \to [D_{\sigma(y_1)} \times \cdots \times D_{\sigma(y_k)}] \to D_{\sigma(a)} \). They are equivalent if they are equivalent in all \( D \). Without loss of generality, we can assume that \( y_i = y'_i \) for all \( i \in [k] \) (otherwise, rename the inherited attributes of \( \Gamma' \) appropriately). Then, we obtain, exactly as for program schemes:

**Proposition.** \((\Gamma, a)\) and \((\Gamma', a')\) are equivalent if and only if they are equivalent in the free interpretation, i.e., if and only if \( \varphi_{a,S}^{(F)} = \varphi_{a',S}^{(F)} \).

This shows that the equivalence problem for attribute systems reduces to deciding the equality of two tree-transductions. Unfortunately these transductions are very "high-level" ones and their equality problem is not at all obvious to solve (Courcelle and Franchi, 1982; Engelfriet, 1980; Datum and Guessarian, 1981). Hence we shall restrict our attention to special classes of attribute systems.

### 1.4. Non-Nested Attribute Systems

An attribute system \( \Gamma \) of type \((P, F)\) is non-nested if the inherited attributes do not depend on the synthesized ones, i.e., if for all \( p \in P \), all \( y \in A^{(h)} \) and all \( i \in [\rho(p)] \), every semantic rule of \( \Gamma_p \) defining \( y(i) \) is of the form

\[
y(i) = s[z^{(1)}(e), \ldots, z^{(k)}(e)]
\]

for some \( F \)-term and some \( z^{(1)}, \ldots, z^{(k)} \) in \( A^{(h)}_{\rho(p)} \). The other semantic rules, i.e., those defining the values of synthesized attributes are of the general form (1.2.1) of definition (1.2).

**Remark.** Non-nested attribute systems are very special ones. Informally speaking, the evaluation of attributes can be done in one depth-first traversal of the tree (starting at the root and returning to it).

For practical use, more complicated attribute grammars (although not necessarily of the most general type) are required.

It follows from Proposition 3.22 of Courcelle et al. (1982) and from Franchi (1982) that non-nested attribute systems can be translated into recursive program schemes of a certain type, called primitive recursive schemes with parameters. The general construction is rather difficult to describe but it is much simpler in the case of non-nested attribute systems; it produces primitive recursive schemes of a special form also called non-nested. Let us first show an example.
1.5. Example

\[ \mathcal{N} = \{S, T\}, \quad P = \{p, q, r, s\}, \]

\[ \sigma(p) = S, \quad a(p) = TS, \quad \sigma(q) = T, \quad a(q) = T, \]

\[ \sigma(r) = T, \quad a(r) = \varepsilon, \quad \sigma(s) = S, \quad a(s) = \varepsilon. \]

Let \( \Gamma \) consist of the following sets of rules:

\[ \Gamma_p \begin{cases} 
  a(\varepsilon) = f_1(b(1), a(2)), \\
  y(1) = f_2, \\
  x(2) = f_3(x(\varepsilon)); 
\end{cases} \]

\[ \Gamma_q \begin{cases} 
  b(\varepsilon) = g_1(y(\varepsilon), b(1)), \\
  y(1) = g_2(y(2)); 
\end{cases} \]

\[ \Gamma_r 
  b(\varepsilon) = h(y(\varepsilon)); \]

\[ \Gamma_s 
  a(\varepsilon) = k(x(\varepsilon)). \]

These rules also use \( F = \{f_1, f_2, f_3, g_1, g_2, h, k\} \) with \( \mathcal{O} \) reduced to a single sort. The attributes \( a, b \) are synthesized and \( x, y \) are inherited.

The value of the attribute \( a \) at the root of an \( S \)-rooted tree \( t \) can be written as \( \varphi_a(t, d) \), where \( d \) is the value of the attribute \( x \) at the root of \( t \). Similarly, the value of \( b \) at the root of a \( T \)-rooted tree \( t \) can be written \( \varphi_b(t, d) \), where \( d \) is the value of \( y \) at the root of \( t \). From this and the equations of \( \Gamma \), the functions \( \varphi_a \) satisfy the following set of equations, denoted by \( \Sigma(\Gamma) \):

\[ \varphi_a(p(t_1, t_2), x) = f_1(\varphi_b(t_1, f_2), \varphi_a(t_2, f_3(x))), \]

\[ \varphi_a(s, x) = k(x), \]

\[ \varphi_b(q(t_1), y) - g_1(y, \varphi_b(t_1, g_2(y))), \]

\[ \varphi_b(r, y) = h(y), \]

for all \( T \)-tree \( t_1 \) and all \( S \)-tree \( t_2 \).

It is easy to see that these equations define \( \varphi_a(t, x) \) and \( \varphi_b(t', y) \) in a unique way for all \( S \)-rooted tree \( t \) and all \( T \)-rooted tree \( t' \). They form a primitive recursive scheme with parameters.

1.6. A Class of Recursive Program Schemes

Primitive recursive schemes with parameters have been introduced and discussed in Courcelle et al. (1982). We do not recall the definition but we only define the subclass of the non-nested ones that we shall use in this paper. For simplicity we shall call them non-nested schemes in the sequel.
Let $P$ and $F$ be as in Section 1.2. A non-nested scheme of type $(P, F)$ is an object $\Sigma$ consisting of the following items:

1. a finite $(\mathcal{N} \cup \mathcal{O})$-signature $\Phi$ called the set of function variables; each $\phi$ in $\Phi$ has an arity $\alpha(\phi)$ in $\mathcal{N} \mathcal{O}^*$ (of the form $\beta(\phi) \alpha'(\phi)$ with $\beta(\phi)$ in $\mathcal{N}$ and $\alpha'(\phi)$ in $\mathcal{O}^*$) and a sort $\sigma(\phi)$ in $\mathcal{O}$;

2. an $\mathcal{O}^*$-sorted set of variables $Y$ called parameters;

3. for each $p$ in $P$ and each $\phi$ in $\Phi$ such that $\beta(\phi) = \sigma(p)$, a defining equation $\Sigma_{p,\phi}$ of the form

$$\phi(p(x_1, \ldots, x_n), y_1, \ldots, y_m) = \tau,$$

where the $x_i$'s are distinct elements of $X$, the $y_i$'s are distinct elements of $Y$ and the right-hand side of (1.6.1) is in

$$M(F, \Phi(X_n \cup M(F, Y_m)) \cup Y_m)_{\sigma(\phi)}$$

and of course is well-type with respect to the signatures (we let $X_n = \{x_1, \ldots, x_n\}$ and $Y_m = \{y_1, \ldots, y_m\}$).

Considering $\tau$ as a tree, this means that there is no more than one occurrence of a symbol of $\Phi$ on any branch of $\tau$. An interpretation is an $F$-magma $D$.

The set of equations $\Sigma(\tau)$ written in Section 1.5 is of this type, with $\Phi = \{\phi_a, \phi_b\}$.

It has been shown in Courcelle et al. (1982) that a system of equations like $\Sigma$ has a unique solution for every interpretation $D$, i.e., that there exists a unique family $(\phi)_{\sigma \in \Phi}$ of functions with $\bar{\phi} : M(P)_{\beta(\phi)} \times D_{a_1} \times \cdots \times D_{a_m} \rightarrow D_{\sigma(\phi)}$ (where $a_1 a_2 \cdots a_m = \alpha'(\phi)$) satisfying the equations of $\Sigma$ in an obvious sense. This family will be written $(\phi_D)_{\sigma \in \Phi}$.

Taking for $D$ the free interpretation $M = M(F, Y)$ and letting

$$\phi_F = \lambda t \in M(P)_{\beta(\phi)} \phi_M(t, y_1, \ldots, y_m)$$

yields the factorization

$$\phi_D = \text{derop}_D \circ \phi_F$$

or more precisely

$$\phi_D = \lambda t \in M(P)_{\beta(\phi)}, d_1 \in D_{a_1}, \ldots, d_m \in D_{a_m} \text{derop}_D(\phi_F(t))(d_1, \ldots, d_m).$$

So that for two function variables $\phi$ and $\phi'$ of same sort and arity, $\phi_D = \phi'_D$ for all $D$ if and only if $\phi_F = \phi'_F$.

Let us also recall from Corollary 4.22 of Courcelle et al. (1982) that $\phi_F(t)$ can be characterized as the normal form of $\phi(t, y_1, \ldots, y_m)$ with respect to the rewriting relation $\rightarrow_{\Sigma}$. 
Let us be precise about this point. If we consider $\Sigma$ as a set of pairs of terms, a rewriting relation $\to_\Sigma$ on $M(P \cup F \cup \Phi, X \cup Y)$ is associated with $\Sigma$ in a well-known way (see Huet, 1980). This relation is Noetherian, confluent (i.e., has the Church–Rosser property), so that every element $s$ of $M(P \cup F \cup \Phi, X \cup Y)$ has a unique normal form, $nf_\Sigma(s)$.

1.6.1. Remark. From the definition of $\Sigma$ we can see that, if $s' = nf_\Sigma(s)$ and $s'$ has a subterm of the form $\varphi(u_0, u_1, \ldots, u_n)$, then $u_0$ must be in $X$. (Otherwise, due to the restrictions concerning the sorts, $u_0$ must be of the form $p(u'_1, \ldots, u'_k)$ and some rule of $\Sigma$ can be applied to $s'$. Hence for $t$ in $M(P)_{\beta(w)}$ and $u$ in $M(P \cup F \cup \Phi, X \cup Y)$ such that $\varphi(t, y_1, \ldots, y_m) \to_\Sigma^* u$: $u = nf_\Sigma(\varphi(t, y_1, \ldots, y_m)) \Leftrightarrow u$ has no occurrence of function variables.

If $\Gamma$ is a non-nested attribute system, Construction (5.1) of Courcelle et al. (1982) applies and produces a non-nested scheme $\Sigma(\Gamma)$ with set of function variables $\Phi = \{\varphi_{a,S}/S \in \mathcal{N}, a \in \mathcal{A}_S(s)\}$ and such that for all interpretation $D$ $\varphi_{a,S}^{(D)} = (\varphi_{a,S})_D$, i.e., such that $\Sigma(\Gamma)$ defines the function $\varphi_{a,S}^{(D)}$ associated by Section 1.3 with $\varphi_{a,S}$ and $D$. Hence

1.6.2. Proposition. The equivalence problem for non-nested attribute systems reduces to the equivalence problem for non-nested schemes.

1.7. Attribute Systems and Non-Nested Schemes with Separated Sorts

Even for non-nested schemes we do not know any algorithm deciding the equivalence problem. Hence we introduce a further condition.

An attribute system $\Gamma$ of type $(P, F)$ is separated if the set $\mathcal{A}$ can be partitioned into $\mathcal{A} = \mathcal{A}^{(s)} \cup \mathcal{A}^{(h)}$ (with $\mathcal{A}^{(s)} \cap \mathcal{A}^{(h)} = \emptyset$) in such a way that:

\begin{align*}
\sigma(a) & \in \mathcal{A}^{(s)} \quad \text{for all } a \in \mathcal{A}^{(s)} \quad \text{(1.7.1)} \\
\sigma(y) & \in \mathcal{A}^{(h)} \quad \text{for all } y \in \mathcal{A}^{(h)} \quad \text{(1.7.2)}
\end{align*}

all symbols of $F$ occurring in a semantic equation defining a synthesized attribute have a sort in $\mathcal{A}^{(s)}$ (we shall denote by $F^{(s)}$ the set $\{f \in F/\sigma(f) \in \mathcal{A}^{(s)}\}$) \quad \text{(1.7.3)}

all symbols of $F$ occurring in a semantic equation defining an inherited attribute have a sort in $\mathcal{A}^{(h)}$ (we shall denote by $F^{(h)}$ the set $\{f \in F/\sigma(f) \in \mathcal{A}^{(h)}\}$). \quad \text{(1.7.4)}

Remark. The separatedness condition is a mild one. It is always possible to transform an attribute grammar $(G, \Gamma, D)$ such that $\Gamma$ is not separated into another one $(G, \Gamma', D')$ which is separated and is equivalent to the first one in some sense (not exactly the one of (1.3)).
It suffices to duplicate sorts that are common to synthesized and inherited attributes, and to duplicate function symbols of $F$ accordingly.

The interpretation $D$ must be transformed into an interpretation $D'$ where some domains $D_a$ have been duplicated into two, say $D'_{(a,s)}$ and $D'_{(a,h)}$ both equal to $D_a$.

We do not give the details.

This transformation does not necessarily preserve the equivalence so that we cannot conclude that the equivalence problem for attribute systems reduces to the equivalence problem for separated attribute systems.

For non-nested schemes, the corresponding concept is the following: A non-nested scheme of type $(P,F)$ is \textit{separated} if the set $\mathcal{A}$ can be partitioned into $\mathcal{A} = \mathcal{A}^{(s)} \cup \mathcal{A}^{(h)}$ in such a way that

\begin{align*}
\sigma(\varphi) &\in \mathcal{A}^{(s)} \text{ for all } \varphi \text{ in } \Phi \quad (1.7.5) \\
\sigma(y) &\in \mathcal{A}^{(h)} \text{ for all parameter } y \text{ in } \mathcal{Y} \quad (1.7.6) \\
\text{the right-hand side of any equation is a member of } &\mathcal{M}(F^{(s)}, \mathcal{Y} \cup \Phi(\mathcal{M}(F^{(h)}, \mathcal{Y}))), \quad (1.7.7)
\end{align*}

where $F^{(s)}$ and $F^{(h)}$ are associated with $\mathcal{A}^{(s)}$ and $\mathcal{A}^{(h)}$ as in (1.7.3) and (1.7.4). Note that the only symbols of $F^{(h)}$ that may occur in such an equation have an arity in $\mathcal{A}^{(h)*}$. Hence we shall assume that $\sigma(f) \in \mathcal{A}^{(h)*}$ for all $f$ in $F^{(h)}$.

By inspecting Construction (5.1) of Courcelle et al. (1982) we can prove that $\Sigma(\Gamma)$ is non-nested and separated when $\Gamma$ is. The rest of the paper will be devoted to the proof of the

**Main Theorem.** \textit{The equivalence problem for separated non-nested schemes is solvable.}

And we shall obtain:

**Corollary.** \textit{The equivalence problem for separated non-nested attribute systems is solvable.}

2. \textbf{First-Order Unification}

In this section, we recall some well-known results concerning first-order unification and fix our notations. Let $F$ be a finite $\mathcal{A}$-sorted signature. Let $V, W$ denote finite sets of $\mathcal{A}$-sorted variables. A $V$-substitution is a sort preserving map $\delta: V \rightarrow \mathcal{M}(F, W)$ (where $W$ is not necessarily disjoint from $V$) which is canonically extended into $\delta: \mathcal{M}(F, V) \rightarrow \mathcal{M}(F, W)$, by $\delta(t) =$
$t[\delta(v_1)/v_1, \delta(v_2)/v_2,...]$. All the sets of variables $Y, V, W, W',...$, that we shall introduce in the sequel will be assumed finite.

A $V$-renaming is a substitution $\delta: M(F, V) \rightarrow M(F, W)$ such that $\delta$ is a bijection $V \rightarrow W$. If $t = \delta(t')$ we say that $t$ is a $V$-renaming of $t'$ and we shall denote this by $t \equiv_{V} t'$.

Let $(t, t')$ be a pair of elements of $M(F, V)$. We say that a substitution $\delta: M(F, V) \rightarrow M(F, W)$ satisfies the equation $(t, t')$ if $\delta(t) = \delta(t')$. We say also $\delta$ is a unifier of the pair $(t, t')$. Let $\mathcal{E}$ be a subset of $M(F, V)^2$, i.e., a set of equations on $M(F, V)$. We shall denote by $\text{UNIF}(\mathcal{E})$ the set of all $V$-substitutions $\delta$ which satisfy all equations in $\mathcal{E}$. It is well known that for every finite set of equations $\mathcal{E}$ such that $\text{UNIF}(\mathcal{E}) \neq \emptyset$, there exists a most general unifier, i.e., a substitution $\delta: V \rightarrow M(F, W)$ (for some $W$) such that

$$\text{UNIF}(\mathcal{E}) = \{ \mu \circ \delta/\mu \text{ is a } W\text{-substitution} \}.$$  

If we assume that every $w$ in $W$ has at least one occurrence in some $\delta(v)$, then $\delta$ is unique up to a $W$-renaming, i.e., if $\delta': V \rightarrow M(F, W')$ is any such most general unifier, then $\delta' = \alpha \circ \delta$ for some bijection $W \rightarrow W'$. The most general unifier (mgu) can be computed by the algorithm of Martelli and Montanari (1982).

In fact we can define the most general unifier $\delta$ of $\mathcal{E}$ as a substitution $\delta: V \rightarrow M(F, V_0)$ such that

$$V_0 \subseteq V$$

$$(2.1.1a)$$

$$\delta(v) = v \quad \text{for all } v \in V_0.$$  

$$(2.1.1b)$$

And then

$$\text{UNIF}(\mathcal{E}) = \text{INVAR}(\delta),$$  

$$(2.1.2)$$

where $\text{INVAR}(\delta)$ denotes the set of all $V$-substitutions $\mu$ which are invariant by $\delta$, i.e., such that $\mu \circ \delta = \mu$. Note that the set $V_0$ is not uniquely defined by (2.1.1). Any substitution $\delta$ satisfying (2.1.1) and (2.1.2) is a mgu of $\mathcal{E}$ and will be called a base of $\text{UNIF}(\mathcal{E})$. The cardinality of $V_0$ will be called the dimension of $\delta$ and denoted by $\text{dim}(\delta)$. The dimension of $\delta$ is the number of independent parameters on which the general solution of $\mathcal{E}$ depends.

Let $\tau$ and $\tau'$ be two bases. (Note that $\text{INVAR}(\tau)$ and $\text{INVAR}(\tau') \neq \emptyset$.) Let us define $\tau \leq \tau'$ if $\text{INVAR}(\tau') \subseteq \text{INVAR}(\tau)$. Let $\tau \sim \tau'$ if $\text{INVAR}(\tau) = \text{INVAR}(\tau')$.

2.1. Lemma. (1) $\tau \leq \tau'$ if and only if $\tau' \circ \tau = \tau'$.

(2) $\tau \leq \tau'$ implies $\text{dim}(\tau') \leq \text{dim}(\tau)$. If $\tau \leq \tau'$ and $\text{dim}(\tau) = \text{dim}(\tau')$, then $\tau \sim \tau'$. 


Proof. (1) Clear from the definitions.

(2) Let $V_0$ and $V'_0$ be the subsets of $V$ associated with $\tau$ and $\tau'$, respectively, by (2.1.1). Then the mapping $\tau$ defines an injection $V'_0 \rightarrow V_0$. To see this, take $v$ in $V'_0$. Then $v = \tau'(v) = \tau'(v)$. It follows that $\tau(v) \in V$. In fact $\tau(v) \in V_0$ by (2.1.1). It is injective since $\tau' = \tau' \circ \tau$. Hence $\dim(\tau') \leq \dim(\tau)$.

If $\dim(\tau) = \dim(\tau')$, then $\tau$ is a bijection $V'_0 \rightarrow V_0$ and $\tau': V_0 \rightarrow V'_0$ is its inverse. We have to show that $\tau \circ \tau' = \tau$. Let $v \in V$. Then $\tau(v) \in M(F, V_0)$. Hence

$$\tau(v) = \tau \circ \tau' \circ \tau(v) \quad \text{(since } \tau \circ \tau' \mid V_0 = \text{id}_{V_0})$$

$$= \tau \circ \tau'(v) \quad \text{(since } \tau' \circ \tau = \tau').$$

Hence we have shown that $\tau \circ \tau' = \tau$. Finally $\tau' \leq \tau$ and $\tau \sim \tau'$. 

2.2. Proposition. For every infinite chain

$$\tau_0 \leq \tau_1 \leq \tau_2 \cdots \leq \tau_n \leq \cdots,$$

there exists an integer $n$ such that $\tau_m \sim \tau_n$ for all $m \geq n$.

Proof. The proof is an immediate consequence of Lemma 2.1 and the finiteness of $V$. 

2.3. Corollary. Let $\mathcal{E}$ be an infinite subset of $M(F, V)^2$. If $\text{UNIF}(\mathcal{E}) \neq \emptyset$, then it has a base, i.e., a most general unifier.

Proof. Let $\mathcal{E} = \bigcup_{n \geq 0} \mathcal{E}_n$, where $\mathcal{E}_n$ is an increasing sequence of finite subsets of $\mathcal{E}$. For all $n$, $\text{UNIF}(\mathcal{E}_n) \neq \emptyset$; hence $\mathcal{E}_n$ has a base $\tau_n$. We get an increasing sequence

$$\tau_0 \leq \tau_1 \leq \tau_2 \leq \cdots \leq \tau_n \leq \cdots.$$

If $n$ is the integer of Proposition 2.2, then $\text{UNIF}(\mathcal{E}_m) = \text{INVAR}(\tau_m = \tau_n)$ for all $m \geq n$ hence $\text{UNIF}(\mathcal{E}) = \bigcap_{m \geq 0} \text{UNIF}(\mathcal{E}_m) = \text{INVAR}(\tau_n)$. 

The following "compactness" result can be extracted from the above proof:

2.4. Proposition. A set $\mathcal{E}$ of equations on $M(F, V)$ has a solution if and only if every finite subset of $\mathcal{E}$ has a solution.

Let us conclude with a notation which will be used in examples. By

$$\{v_{i_1} \rightarrow t_1 ; v_{i_2} \rightarrow t_2 ; \ldots ; v_{i_k} \rightarrow t_k\},$$
where $v_1, \ldots, v_k$ are pairwise distinct, and $t_1, \ldots, t_k \in M(F, V)$, we denote the substitution $\delta: V \to M(F, V)$ such that $\delta(v_i) = t_j$ for all $j = 1, \ldots, k$ and $\delta(v) = v$ for all $v$ in $V - \{v_1, \ldots, v_k\}$.

### 3. The Decidability Result

#### 3.1. Introductory Remarks

We begin with a sequence of remarks, notations, conventions, and preliminary results. Let $\Sigma$ be a separated non-nested scheme as in Section 1.7. Let $\eta$ be the set of sorts of signature $P$.

Instead of $\mathcal{O}(s)$ and $\mathcal{O}(h)$ we shall use sets of sorts $\mathcal{O}$ and $\mathcal{B}$. Instead of $F(s)$ and $F(h)$ we shall use $F$ and $G$, respectively. Hence $F$ is an $(\mathcal{O} \cup \mathcal{B})$-signature, $G$ is a $\mathcal{B}$-signature, and the scheme $\Sigma$ is assumed to be of type $(P, F \cup G)$.

Let $\Phi$ be its set of function variables. Each $\varphi$ in $\Phi$ has a sort $\sigma(\varphi)$ in $\mathcal{O}$ and an arity $\alpha(\varphi) = \beta(\varphi) \alpha'(\varphi)$ in $\eta \beta^*$ (with $\beta(\varphi) \in \eta$).

Each function variable $\varphi$ defines a tree-transduction denoted by $\varphi_\mathcal{F}$ in Section 1.6. We shall use here the notation $\varphi^*$ instead of $\varphi_{F \cup G}$.

\[
\varphi^*: M(P)_{\beta(\varphi)} \to M(F \cup G, Y)_{\alpha(\varphi)}.
\]

Due to be restrictions concerning signatures, $\varphi^*$ maps $M(P)_{\beta(\varphi)}$ into $M(F, M(G, Y))_{\alpha(\varphi)}$. And as in Courcelle et al. (1982) we assume that $M(P)_{\alpha} \neq \emptyset$ for all $S$ in $\eta$. For every $\mathcal{B}$-sorted set of variables $W$, every term $t$ in $M(F \cup G \cup \Phi, \{x_1, \ldots, x_n\} \cup W)$ of sort $s$ in $\mathcal{O}$ defines a mapping

\[
t^*: M(P)_{\sigma(x_1)} \times \cdots \times M(P)_{\sigma(x_n)} \to M(F, M(G, W))_s
\]

in an obvious way: just interpret $\varphi$ as $\varphi^*$. It can also be defined by $t^*(u_1, u_2, \ldots, u_n) = \eta \delta^* (t[u_1/x_1, \ldots, u_n/x_n])$.

A pair of terms $(t, t')$ in $M(F \cup G \cup \Phi, \{x_1, \ldots, x_n\} \cup W)^2$ is called an equivalence.

Let $X$ be a fixed finite $\eta$-sorted set of variables (equal to $\{x_1, \ldots, x_n\}$) containing enough variables of each sort so that for every $p$ in $P$ we can find a term $p(y_1, \ldots, y_k)$ with $y_1, \ldots, y_p \in X$ and $y_i \neq y_j$ if $i \neq j$.

Let $W$ be a fixed $\mathcal{B}$-sorted set of variables. For $a$ in $\mathcal{O}$, let $M_a$ denote the
set of terms \( M(F \cup G \cup \Phi, X \cup W) \). By the constraints due to the arities, \( M_a = M(F, M(G, W) \cup \Phi(X \cup M(G, W))) \).

If \( t \in M_a \) and \( \delta \) is a substitution \( W \to M(G, W') \), then \( \delta(t)^* = \delta \circ (t^*) \). (3.1.4)

This follows from the given characterization of \( t^* \) in terms of \( n \mathbf{f} \) and the fact that for all \( u, u' \) in \( M(F \cup G \cup P \cup \Phi, W) \), if \( u \to \_ u' \), then \( \delta(u) \to \_ \delta(u') \).

Let \( (t, t') \) belong to \( M_a^2 \).

We say that \( \delta \) satisfies \( (t, t') \), and we denote this by \( \delta \vdash t \equiv t' \) if \( \delta(t)^*(w_1, \ldots, w_n) = \delta(t')^*(w_1, \ldots, w_n) \) for all \( (w_1, \ldots, w_n) \) in \( M(P)_{\sigma(x)} \times \cdots \times M(P)_{\sigma(x)} \), i.e., if the mappings \( \delta(t)^* \) and \( \delta(t')^* \) are the same.

We say that \( (t, t') \) is true if \( \delta \vdash (t, t') \) for all \( \delta: W \to M(G, W') \), all \( W' \). This is equivalent to \( \text{id}_w \vdash t \equiv t' \). We shall denote this by \( \vdash t \equiv t' \). Remark that \( \delta \vdash t \equiv t' \) implies \( \vdash \delta(t) \equiv \delta(t') \).

An equivalence is false if it is not true.

The set of false equivalences is recursively enumerable. It suffices to find some \( n \)-tuple of arguments \( (w_1, \ldots, w_n) \) such that \( t^*(w_1, \ldots, w_n) \neq t'^*(w_1, \ldots, w_n) \) to establish that \( (t, t') \) is false. By using the decision systems of Courcelle (1983) we shall show that the set of true equivalences is recursively enumerable. An algorithm will be given in Section 4.

We shall have to check equivalences \( (t, t') \) for \( t, t' \) in \( M_a \) and \( a \) in \( \mathcal{O} \).

Let \( V \) be a new \( \mathcal{P} \)-sorted set of variables. Any term \( t \) in \( M_a \) with a sort in \( \mathcal{O} \) can be written in an essentially unique way as \( \delta(u) \) for some \( V \)-linear term \( u \) in \( M_a' = M(F \cup \Phi, X \cup V) \), i.e., in \( M(F, M(G, V)) \) and some substitution \( \delta: V \to M(G, W) \).

Similarly, an equivalence \( (t, t') \) with \( t, t' \in M_a \) will be replaced by a triple \( \text{REP}(t, t') = (u, u', \delta) \) such that \( u, u' \in M_a' \), \( t = \delta(u) \), \( t' = \delta(u') \) and such that the pair \( (u, u') \) is \( V \)-linear. We mean by this that \( u \) and \( u' \) are \( V \)-linear and have no variable from \( V \) in common. In other words \( f(u, u') \) is \( V \)-linear.

We assume that for each sort \( b \) in \( \mathcal{B} \) the set of variables of sort \( b \) is \( \{v_1^b, v_2^b, \ldots, v_n^b \} \). We also require that in a pair \( u, u' \) the variables of \( V \) of any sort \( b \) are \( v_1^b, v_2^b, \ldots, v_k^b \) for some \( k \) and appear in this order from left to right. Hence there exists a unique triple \( (u, u', \delta) \) associated with \( (t, t') \).

For example, if \( \mathcal{B} \) is reduced to one sort and if \( t = \phi(x, g(y, z), a), t' = (x, a, h(z, y)) \), then \( \text{REP}(t, t') \) is the triple

\[
(\phi(x, v_1, v_2), \psi(x, v_3, v_4), \{v_1 \to g(y, z); v_2 \to a; v_3 \to a; v_4 \to h(z, y)\}).
\]

For \( u, u' \in M_a' \), we denote by \( \text{SAT}_V(u, u') \) the set of \( V \)-substitutions \( \delta \) which satisfy \( (u, u') \).
3.2. Lemma. There exists a subset $\mathcal{S}_{u,u'}$ of $M(F, V)_{a}^{2}$ such that $\text{SAT}_{\nu}(u, u') = \text{UNIF}(\mathcal{S}_{u,u'})$. Hence $\text{SAT}_{\nu}(u, u') \neq \emptyset$ if and only if there exist a base $\tau$ such that $\text{INVAR}(\tau) = \text{SAT}_{\nu}(u, u')$.

Proof. We have already noted that $[\delta(u)]^{*}(w_{1},...,w_{n}) = \delta(u^{*}(w_{1},...,w_{n}))$. Hence $\{\delta/\delta \models u \equiv u'\}$ is $\text{UNIF}(\mathcal{S}_{u,u'})$, where $\mathcal{S}_{u,u'} = \{(u^{*}(w_{1},...,w_{n}), u'^{*}(w_{1},...,w_{n}))/\text{for } X\text{-substitutions } W: X \rightarrow M(P)\}$. The second assertion is a consequence of Corollary 2.3.

Hence:

3.3. Proposition. Let $(t, t') \in M_{a}^{2}$ and $(u, u', \delta) = \text{REP}(t, t')$. Then $t \equiv t'$ if and only if there exists some substitution $\tau: V \rightarrow M(G, V)$ such that $\delta \in \text{INVAR}(\tau)$ and $\models u \equiv u'$.

Proof. For the "only if" part we take for $\tau$ a base of $\text{UNIF}(\mathcal{S}_{u,u'})$. If $\delta \circ \tau = \delta$ and $\models u \equiv u'$, then $\delta(\tau(u)) = \delta(u)$, $\delta(\tau(u')) = \delta(u')$, and $\models \tau(u) \equiv \tau(u')$.

Hence $\models \delta(\tau(u)) \equiv \delta(\tau(u'))$ and $\models \delta(u) \equiv \delta(u')$, i.e., $t \equiv t'$.

Since $\mathcal{S}_{u,u'}$ is infinite the determination of a base of $\text{UNIF}(\mathcal{S}_{u,u'})$ is not a trivial application of Martelli and Montanari (1982).

3.4. Decision Systems

We recall some definitions and results from Courcelle (1983). A decision system is an object $D = \langle \mathcal{A}, \mathcal{S}, \text{exp}, \text{split}, \vdash \rangle$ satisfying the following conditions:

$\mathcal{A}$ is a countable set of objects called assertions.

We assume that $\mathcal{A}$ is recursive, i.e., that we can decide whether a given object is in $\mathcal{A}$ or not. For every $n \geq 0$, an assertion $A$ in $\mathcal{A}$ is either $n$-true or $n$-false.

If an assertion is $(n+1)$-true, then it is $n$-true. (3.4.1)

An assertion is true if and only if it is $n$-true for all $n$. (3.4.2)

We let $\mathcal{A}_{A}$ denote $\mathcal{A} \cup \{A\}$, where $A$ is a new symbol standing for an "obviously" false assertion. $A$ is 0-false, whereas all $A$ in $\mathcal{A}$ are 0-true.

The symbol $\vdash$ denotes a subset of $P_{0}(\mathcal{A}) \times \mathcal{A}$ called a deduction relation.

We write $\mathcal{S} \vdash A$ instead of $(\mathcal{S}, A) \in \vdash$. 

We assume the following properties:

\[ A \in \mathcal{G} \text{ implies } \mathcal{G} \vdash A, \]  
\[ \mathcal{G}' \vdash A \text{ and } \mathcal{G} \vdash B \text{ for all } B \text{ in } \mathcal{G}' \text{ imply } \mathcal{G} \vdash A, \]  
\[ A \text{ is } n\text{-true if } \mathcal{G} \vdash A \text{ and } B \text{ is } n\text{-true for all } B \text{ in } \mathcal{G}. \]

Finally, we extend \( \vdash \) into a binary relation on \( \mathcal{P}(A) \) by letting:

\[ \mathcal{G} \vdash \mathcal{G}' \text{ if and only if for all } A \text{ in } \mathcal{G}', \text{ there exists a finite subset } \mathcal{G}'' \text{ of } \mathcal{G} \text{ such that } \mathcal{G}'' \vdash A. \]

\( \mathcal{G} \) is a subset of \( \mathcal{A} \), called the set of **elementary assertions**. We denote \( \mathcal{G} \cup \{ A \} \) by \( \mathcal{G}_A \). We denote by \( \text{exp} \) a partial recursive mapping \( \mathcal{A} \rightarrow \mathcal{P}_0(\mathcal{A}_A) \) the domain of which is a recursive subset of \( \mathcal{A} \) denoted by \( \mathcal{A}_{\text{exp}} \) and such that \( E \subseteq \mathcal{A}_{\text{exp}} \).

We say that an assertion \( A \) in \( \mathcal{A}_{\text{exp}} \) is **expanded** into a set \( \text{exp}(A) \) of assertions. We assume the following properties, for all \( A \) in \( \mathcal{A}_{\text{exp}} \)

\[ A \in T \text{ implies } \text{exp}(A) \subseteq \mathcal{E}, \]

\[ \text{if } \text{exp}(A) \subseteq n - \mathcal{E}, \text{ then } A \in (n + 1) - \mathcal{E}, \]

where \( \mathcal{E} \) (resp. \( n - \mathcal{E} \)) denotes the set of true (resp. \( n\)-true) assertions.

We define \( \text{exp}(\mathcal{G}) = \bigcup \{ \text{exp}(A)/A \in \mathcal{G} \} \) for any subset \( \mathcal{G} \) of \( \mathcal{A}_{\text{exp}} \). Let \( \mathcal{A}_{\text{split}} \) be the set of nonelementary assertions. We assume the existence of a multi-valued mapping \( \text{split}: \mathcal{A}_{\text{split}} \rightarrow \mathcal{P}_0(\mathcal{A}_A) \).

Its purpose is to **split** a “complex” assertion \( A \) into a finite set of “elementary” ones. We shall allow this to be done in different possible ways. Hence \( \text{split} \) is multivalued in the general case.

The axioms concerning \( \text{split} \) are the following ones (where \( A \) is in \( \mathcal{A}_{\text{split}} \) and \( \mathcal{G} \) is any value of \( \text{split}(A) \)):

\[ A \in \mathcal{G} \text{ implies } \mathcal{G} \subseteq \mathcal{G}, \]

\[ \text{if } A \notin \mathcal{G}, \text{ then } \mathcal{G} \vdash A. \]

Axiom (3.4.9) reduces the proof of a goal assertion \( A \) to several proofs of “simpler” ones.

We shall use part 2 of Theorem 2.14 of Courcelle (1983) saying the following:

3.5. **THEOREM.** If \( D = \langle \mathcal{A}, \mathcal{E}, \text{exp}, \text{split}, \vdash \rangle \) is a decision system such that \( \mathcal{G} \cap \mathcal{E} \) is finite and \( \vdash \) is recursively enumerable, then the truth of an assertion is semi-decidable.

The proof is based on the following definition and lemma: A set of assertions \( \mathcal{G} \) is **self-proving** if there exists a subset \( \mathcal{G}' \) of \( \mathcal{A}_{\text{exp}} \) such that
$C' \vdash C$ and $C \vdash \exp(C')$. Since $\vdash$ is a binary relation on $\mathcal{P}(\mathcal{A})$ and not on $\mathcal{P}(\mathcal{A}_\lambda)$, the above notations imply that $A \in \mathcal{C} \cup \mathcal{C}' \cup \exp(\mathcal{C}')$. Note that if $\mathcal{C} \subseteq \mathcal{C}_\exp$, $\mathcal{C}$ is self-proving if and only if $\mathcal{C} \vdash \exp(\mathcal{C})$ (which implies $A \in \mathcal{C} \cup \exp(\mathcal{C})$).

3.6. LEMMA. *A self-proving set of assertions is true.*

Then theorem (3.5) is the consequence of the following remarks:

(i) $\mathcal{C} \cap \mathcal{C}$ is self-proving,

(ii) $A$ is true if and only if $\mathcal{C} \vdash A$ for some finite self-proving subset $\mathcal{C}$ of $\mathcal{A}$,

(iii) for a finite subset $\mathcal{C}$ of $\mathcal{A}$, the property "$\mathcal{C}$ is self-proving" is semi-decidable.

3.7. A Decision System for Separated Non-Nested Schemes

We define a decision system $D=\langle \text{ASSERT}, \text{ELEM}, \text{EXP}, \text{SPLIT}, \vdash \rangle$ as follows: Its set of assertions ASSERT is the set of equivalences introduced above. We shall use the word equivalence instead of assertion in the sequel. The truth of an equivalence has already been defined. For $m \in \mathbb{N}$, an equivalence $(t, t')$ is $m$-true if

$$t^*(w_1, \ldots, w_n) = t'^*(w_1, \ldots, w_n)$$

for all $(w_1, \ldots, w_n)$ in $M(P)_{\sigma(x_1)} \times \cdots \times M(P)_{\sigma(x_n)}$ such that $\max \{|w_i|/1 \leq i \leq n\} < m$.

We now define the deduction relation $\mathcal{C} \vdash t \equiv t'$ (read $\mathcal{C}$ proves $(t, t')$) as something like "$(t, t')$ belongs to the congruence generated by $\mathcal{C}$." That is, $\vdash$ is the least relation on $\mathcal{P}(\text{ASSERT}) \times \text{ASSERT}$ such that:

(3.8.1) $\mathcal{C} \vdash t \equiv t$,

(3.8.2) $\mathcal{C} \vdash t \equiv t'$ if $\mathcal{C} \vdash t' \equiv t$,

(3.8.3) $\mathcal{C} \vdash t \equiv t''$ if $\mathcal{C} \vdash t \equiv t''$ and $\mathcal{C} \vdash t'' \equiv t'$,

(3.8.4) $\mathcal{C} \vdash f(t_1, \ldots, t_k) \equiv f(t_1', \ldots, t_k')$ if $\mathcal{C} \vdash t_i \equiv t_i'$ for $i = 1, \ldots, k$ and $f$ in $F \cup G \cup \Phi$.

(3.8.5) $\mathcal{C} \vdash t \equiv t'$ if $t = \delta(u), t' = \delta(u')$ for some $(u, u')$ in $\mathcal{C}$ and some substitution $\delta: W \rightarrow M(G, W)$.

(3.8.6) $\mathcal{C} \vdash t \equiv t'$ if $t = \mu(u), t' = \mu(u')$ for some $(u, u') \in \mathcal{C}$ and some substitution $\mu: X \rightarrow X$.

(3.8.7) $A \in \mathcal{C}$ implies $\mathcal{C} \vdash A$.

(3.8.8) $\mathcal{C} \vdash \mathcal{C}'$ and $\mathcal{C}' \vdash A$ imply $\mathcal{C} \vdash A$. 
The domain $\text{ASSERT}_{\text{EXP}}$ of the function $\text{EXP}: \text{ASSERT}_{\text{EXP}} \to \mathcal{P}(\text{ASSERT})$ is the set of monadic equivalences (abbreviated m.e.) i.e., the set of equivalences $(u, u')$ such that $\text{Var}_X(u) \cup \text{Var}_X(u')$ is a singleton. We now define EXP. Let $x$ be the unique variable of $X$ appearing in $(u, u')$, assumed to be of sort $S$. We shall say that $(u, u')$ is of sort $S$. Any $p$ in $P$ of sort $S$ will be said compatible with $(u, u')$.

For all $p$ compatible with $(u, u')$, we define $\text{EXP}(p, u, u')$ as a pair of terms as follows:

Let $u_1 = u[p(x_1, \ldots, x_n)/x]$ and $u_1' = u'[p(x_1, \ldots, x_n)/x]$, where $x_1, \ldots, x_n$ are variables in $X$ such that $p(x_1, \ldots, x_n)$ is well formed. (3.9.1)

Let $u_2 = \text{nf}_x(u_1)$ and $u_2' = \text{nf}_x(u_1')$. (Note that $u_2$ and $u_2'$ have no occurrences of symbols of $P$). (3.9.2)

Let $\text{EXP}(p, u, u') = \{(u_2, u_2')\}$. (3.9.3)

Finally, we let $\text{EXP}(u, u')$ be the union of the sets $\text{EXP}(p, u, u')$ for all $p$ compatible with $(u, u')$. An assertion $(t, t')$ is strongly true if it is true and in the triple $(u, u', \tau) = \text{REP}(t, t')$, $\tau$ is a base of $\text{UNIF}(u, u')$. An elementary assertion is a monadic equivalence $(t, t')$ such that:

(i) $\text{FIRST}(t) \in \Phi$, and

(ii) $(t, t')$ is strongly true.

Finally, we have to define SPLIT: $(\text{ASSERT-\text{ELEM}}) \to \mathcal{P}(\text{\text{ELEM}})$. It will be convenient to define it: $\text{ASSERT} \to \mathcal{P}(\text{\text{ELEM}})$. We shall do so with help of auxiliary functions SPL, SPLO, SPL1: $\text{ASSERT} \to \mathcal{P}(\text{ASSERT})$.

Let us define SPL first. Let $M = M(F \cup G \cup \Phi, X \cup W) = \bigcup_{a \in \mathbb{A}} M_a$. For each $E$ in $M^2$ we define a subset $\text{SPL}(E)$ of $M^2 \cup \{A\}$ by the following recursive definition:

If $t = t'$, then $\text{SPL}(t, t') = \emptyset$. (3.10.1)

If $t = f(t_1, \ldots, t_h)$ and $t' = f(t_1', \ldots, t_h')$ for $f$ in $F \cup G \cup \Phi$, then $\text{SPL}(t, t') = \text{SPL}(t_1, t_1') \cup \cdots \cup \text{SPL}(t_h, t_h')$. (3.10.2)

If $t$ and $t'$ are of different sort, or if $\text{FIRST}(t) = f$ and $\text{FIRST}(t') = f'$ for $f, f'$ in $F \cup G \cup X$ such that $f \neq f'$, then $\text{SPL}(t, t') = \{A\}$. (3.10.3)

If $t \in W$ and $t' \in M(G, W)$ or vice-versa with $t \neq t'$, then $\text{SPL}(t, t') = \{(t, t')\}$. (3.10.4)

If $\text{FIRST}(t) \in F$ and $\text{FIRST}(t') \in \Phi$, then $\text{SPL}(t, t') = \text{SPL}(t', t)$. (3.10.5)
If none of these cases hold, then \( t = \varphi(x, t_1, \ldots, t_k) \) and we have still several cases to distinguish:

If \( \text{VAR}_x(t') \) contains other variables than \( x \), we choose in \( t' \) some subterm of the form \( \psi(x', t'_1, \ldots, t'_j) \), we choose some \( u \) in \( M(P)_{\sigma(x')} \) we compute

\( w = \psi^*(u, t'_1, \ldots, t'_j) \) we define \( t'' \) by substituting \( w \) for \( \psi(x', t'_1, \ldots, t'_j) \) in \( t' \) and we define

\[
\text{SPL}(t, t') = \text{SPL}(t, t'') \cup \text{SPL}(\psi(x', t'_1, \ldots, t'_j), w). \tag{3.10.6}
\]

Finally, we are left with a unique case

\[
t = \varphi(x, t_1, \ldots, t_k) \text{ and } t' \in M(F \cup \Phi \cup G, \{x\} \cup W)_{\sigma(a)},
\]

then \( \text{SPL}(t, t') = \{(t, t')\}. \tag{3.10.7}

It is easy to see that every element of \( \text{SPL}(t, t') \) is of one of the possible three forms:

1. \( A \),
2. \( (v, s) \) or \( (s, v) \) for \( v \) in \( W \) and \( s \) in \( M(G, W) \),
3. \( (t, t') \) with \( t \) and \( t' \) as in case (3.10.7), i.e., \( (t, t') \) is a monadic equivalence,

We let \( \text{SPLO}(t, t') \) denote the set of elements of type (1) or (2) and \( \text{SPL1}(t, t') \) the remaining, i.e., the elements of type (3).

We define an auxiliary mapping \( H: \text{ASSERT}_A \to \text{ASSERT}_A \).

\[
H(A) = A,
\]

\[
H(t, t') = A \quad \text{if } \ \text{UNIF}(\mathcal{E}_{u,u'}) = \emptyset
\]

\[
= (v(u), v(u')), \quad \text{where } v \text{ is a base of } \text{UNIF}(\mathcal{E}_{u,u'}) \neq \emptyset
\]

and \( \text{REP}(t, t') = (u, u', r) \).

Finally, we define \( \text{SPLIT}: \text{ASSERT} \to \mathcal{P}(\text{ELEM}) \) by

\[
\text{SPLIT}(E) = \{A\} \quad \text{if } \ E \text{ is false}
\]

\[
\text{SPLIT}(E) = H(\text{SPL1}(E)) = \{H(E')/E' \in \text{SPL1}(E)\} \quad \text{if } \ E \text{ is true.}
\]

Note that \( H \) and \( \text{SPLIT} \) are not defined in an effective way. On the contrary, \( \text{SPL} \), \( \text{SPL0} \) and \( \text{SPL1} \) are computable. Let us prove some properties of these functions.
3.11. Lemma. If $E$ is true, then $H(E)$ is strongly true and $H(E) \vdash E$.

Proof. Let $(t, t')$ be true, let $(u, u', \tau) = \text{REP}(t, t')$ and $v$ be a base of UNIF($\delta'_{u, u'}$). Then $\tau \circ v = \tau$, hence $(v(u), v(u'))$ proves $(\tau(v(u)), \tau(v(u')))$, i.e., $(t, t')$. The equivalence $(v(u), v(u'))$ is strongly true by the definitions. \[\square\]

3.12. Lemma. (1) Let $\delta: W \rightarrow M(G, W)$. If $\delta \models E$, then $\delta \models \text{SPL}(E)$.

(2) If $\not\models E$, then $\not\models \text{SPL}(E)$ and $\text{SPL}_{0}(E) = \emptyset$.

(3) If $\text{SPL}_{0}(E) = \emptyset$, then $\text{SPL}_{1}(E) \models E$.

Proof. The proof will be an induction on the computation of SPL$(E)$ in the following sense:

For each case (3.10.1) to (3.10.6) we show the proposition for $E$ by assuming it true for $E_{1}, \ldots, E_{k}$ appearing as argument of SPL in the definition of SPL$(E)$.

Let us consider case (3.10.2) briefly.

Assertion 1. Let $\delta \models f(t_{1}, \ldots, t_{k}) = f(t'_{1}, \ldots, t'_{k})$,

then

$\delta \models t_{i} \equiv t'_{i} \quad \text{for} \quad i = 1, \ldots, k$.

Hence

$\delta \models \text{SPL}(t_{i}, t'_{i})$

by inductive hypothesis, hence

$\delta \models \text{SPL}(t_{1}, t'_{1}) \cup \cdots \cup \text{SPL}(t_{k}, t'_{k})$,

i.e.,

$\delta \models \text{SPL}(f(t_{1}, \ldots, t_{k}), f(t'_{1}, \ldots, t'_{k}))$.

Assertion 2 is a consequence of Assertion 1, and Assertion 3 can be proved similarly.

We now consider case (3.10.6) is some more detail. By (3.10.6) we have SPL$(t, t') = \text{SPL}(t, t'') \cup \text{SPL}(s, w)$ with $s = \psi(x', t'_{1}, \ldots, t'_{k})$.

Assertion 2. Let $\delta \models t \equiv t'$ with $t = \varphi(x, t_{1}, \ldots, t_{k})$ and $\text{Var}_{x}(t') = \{x, x', \ldots\}$. Then the functions $\delta(t)^{*}$ and $\delta(t')^{*}$ are the same and $\delta(t')^{*}$ is constant in its other arguments than $x$, i.e., in particular in $x'$.

This means that, then the function $\delta(s)^{*}$ associated with the subterm $s = \psi(x', t'_{1}, \ldots, t'_{k})$ of $t'$ is constant, i.e., that $\psi^{*}$ is constant in its first argument.
We have \( w = \psi^*(u, t'_1, \ldots, t'_n) \) for some \( u \) in \( M(P, x'_2 \ldots, x'_{n+1}) \) and \( t'' \) is defined by substituting \( w \) for \( s \), i.e., for \( \psi(x', t_1, \ldots, t_n) \) in \( t' \), and \( \delta \models t = t'' \) and \( \delta \models s \equiv w \). Then the inductive hypothesis allows us to conclude that \( \delta \models \text{SPL}(t, t') \) as for case (3.10.2).

Assertion 2 is a consequence of Assertion 1. We now consider

Assertion 3. Let \( t \) and \( t' \) be as above and let us assume that \( \text{SPL}_0 (t, t') = \emptyset \), i.e., that \( \text{SPL}_0 (t, t'') = \text{SPL}_0 (s, w) = \emptyset \).

By the inductive hypothesis

\[ \text{SPL}_1 (t, t'') \models t \equiv t'' \quad \text{and} \quad \text{SPL}_1 (s, w) \models s \equiv w. \]

By several uses of clauses (3.8.1), (3.8.3), (3.8.4), of the definition of \( \models \) we can show that

\[ \text{SPL}_1 (s, w) \models t' \equiv t''. \]

Hence by (3.8.8)

\[ \text{SPL}_1 (t, t') = \text{SPL}_1 (t, t'') \cup \text{SPL}_1 (s, w) \models \{ t \equiv t'', t' \equiv t'' \} \]

and by (3.8.2) and (3.8.3)

\[ \text{SPL}_1 (t, t') \models t \equiv t'. \]

3.13. LEMMA. (1) \( E \) is true if and only if \( A \in \text{SPLIT}(E) \).

(2) If \( E \) is true, then \( \text{SPLIT}(E) \) is true and \( \text{SPLIT}(E) \models E \).

Proof. If \( E \) is false, then \( A \in \text{SPLIT}(E) \) by the definition of \( \text{SPLIT} \). If \( E \) is true, then \( \text{SPL}_1 (E) \) is true by Lemma 3.12(2) and \( H(\text{SPL}_1 (E)) \) = \( \text{SPLIT}(E) \) is true by Lemma 3.11. Hence \( A \notin \text{SPLIT}(E) \), Lemma 3.12 shows that \( \text{SPL}_1 (E) \models E \) hence \( H(\text{SPL}_1 (E)) \models E \) by Lemma 3.11 and (3.8.8).

Q.E.D.

3.14. PROPOSITION. \( D \) is a decision system such that \( \models \) is recursively enumerable and \( \emptyset \notin \text{ELEM} \), the set of true elementary assertions, is finite.

This proposition with Theorem (3.5) immediately proves

3.15. THEOREM. The truth of an equivalence is semi-decidable.

The main theorem follows from Proposition 3.14 and Theorem 3.5, since by Remark 3.2.4 the falsity of an equivalence is semi-decidable.

Proof of Proposition 3.14. We have to show that \( D \) is a well-formed decision system, i.e., that the various conditions of definition 3.4 are satisfied. Most of them are very easy to establish.
Conditions (3.4.1) and (3.4.2) concerning the truth of equivalences follow from the definitions. Conditions (3.4.3) and (3.4.4) concerning \( \vdash \) follow from the definition. Condition (3.4.5) can be proved by induction on the length of a derivation establishing that \( \mathcal{E} \vdash A \) by means of clauses (3.8.1)–(3.8.8).

Note on the way that \( \vdash \) is recursively enumerable: for any given sequence of derivation steps concerning pairs \( (\mathcal{E}, A) \) in \( P_0(\mathcal{D}) \times \mathcal{D} \), we can check whether it really uses clauses (3.8.1)–(3.8.8).

It is clear that \( \text{ASSERT}_{\text{EXP}} \) is recursive and that \( \text{EXP} \) is a total and computable mapping. Conditions (3.4.6) and (3.4.7) are easy to verify.

The conditions concerning SPLIT follow immediately from Lemma 3.13. Let us establish that \( \mathcal{E} \text{ELEM} \) is finite. Let \((t, t') \in \mathcal{E} \text{ELEM}\), \(x\) be its unique variable in \(X\), let \(\text{REP}(t, t') = (u, u', \tau)\). Since \(\tau\) is determined by \((u, u')\) we only have to show that there is only finitely many possible pairs \((u, u')\). We can assume that

\[
\begin{align*}
\phi(x, v_1, \ldots, v_k) \quad \text{(hence is in } \Phi(X \cup V_k)) \quad u' = s[w_1, \ldots, w_e] \\
\text{for some } F\text{-term } s \text{ and some } w_1, \ldots, w_e \text{ in } \\
V_{k,m} \cup \Phi(X \cup V_{k,m}), V_{k,m} = \{v_{k+1}, \ldots, v_m\}.
\end{align*}
\]  

(3.14.1)

Then \(\tau: V_m \rightarrow M(G, V_m)\) is such that \(\tau \vdash u \equiv u'\). Let \(t\) be any element of \(M(P)_{\phi(x)}\) (\(\neq \emptyset\) by our initial assumptions concerning signatures). Then \(\tau \vdash u^*(t) \equiv u'^*(t)\) and

\[
\begin{align*}
u^*(t) &= \phi^*(t, v_1, \ldots, v_k) \\
u'^*(t) &= s[w_1^*(t), \ldots, w_e^*(t)] \\
\text{and } w_1^*(t), \ldots, w_e^*(t) \text{ belong to } M(F, V_{k,m}).
\end{align*}
\]  

It follows that \(\tau(u^*(t)) = \tau(u'^*(t))\), hence since \(F \cap G = \emptyset\) (\(\Sigma\) being separated)

\[
|s|_F \leq |u^*(t)|_F,
\]  

(3.14.2)

where \(|z|_F\) denotes the number of occurrences of symbols in \(F\) in a element \(z\) of \(M(F, V)\).

Since \(X\) and \(\Phi\) are finite, there exist only finitely many possible terms \(u\) satisfying (3.14.1). By (3.14.2), for given \(u\), there exist only finitely many possible terms \(s\), hence finitely many terms \(u'\) satisfying (3.14.1). Hence the number of possible pairs \((u, u')\) satisfying (3.14.1) is finite and \(\mathcal{E} \text{ELEM}\) is finite.

3.15. Remark. The decidability result we have just proved concerns primitive recursive program schemes which are non-nested and separated. The latter condition is crucial for inequality (3.14.2). Otherwise, i.e., if we do not have \(F \cap G = \emptyset\) examples can be given where \(\mathcal{E} \text{ELEM}\) is infinite. And this condition is only used here.
The former, namely the restriction to non-nested schemes, is essential for the definition of the class \( \text{ELEM} \), that of \( \text{SPLIT} \) and Lemma 3.13(2).

A more complicated way to "split" equivalence must be found if we want to extend the method to arbitrary (possibly separated) primitive recursive program schemes.

4. AN ALGORITHM

In this section, we provide an algorithm which constructs a smallest self-proving subset of \( \mathcal{E} \text{ELEM} \) which proves a given equivalence if it is true and stops with a FAILURE answer if it is not true.

4.1. DEFINITIONS. (1) We shall manipulate triples \((u, u', \tau)\) called facts such that:

\begin{enumerate}
  \item \((u, u')\) is a \(V\)-linear pair of elements of \(M(F \cup \Phi, \{x\} \cup V)\), for some \(x\),
  \item the variables of \(V\) occurring in \((u, u')\) are numbered from left to right in a canonical way, as in (3.1.7).
  \item \(\tau\) is a substitution \(V \rightarrow M(G, V)\) satisfying (2.1.1). Hence \((\tau(u), \tau(u'))\) is a monadic equivalence.
\end{enumerate}

A fact \((u, u', \tau)\) is true if \(\text{UNIF}(\mathcal{E}u, u, \tau) \subseteq \text{INVAR}(\tau)\). It is optimal if \(\text{UNIF}(\mathcal{E}u, u, \tau) = \text{INVAR}(\tau) \neq \emptyset\). This means the following:

If \((u, u', \tau)\) is true, then, for all \(v\) such that \(\models v(u) \equiv v(u')\), \(v \in \text{INVAR}(\tau)\).
If \((u, u', \tau)\) is optimal, then further more, if \(v \in \text{INVAR}(\tau)\), then \(\models v(u) \equiv v(u')\).

4.1.1. Remark. The following properties of any fact \((u, u', \tau)\) are equivalent:

\begin{enumerate}
  \item \((u, u', \tau)\) is true and \(\models \tau(u) \equiv \tau(u')\),
  \item \((u, u', \tau)\) is optimal,
  \item \((\tau(u), \tau(u'))\) is a strongly true equivalence.
\end{enumerate}

Note finally that \((u, u', id)\) is always true even if \(\text{UNIF}(\mathcal{E}u, u, \tau) = \emptyset\) (and \(id: V \rightarrow V\) is such that \(id(v) = v\)).

(2) Sets of facts are ordered as follows:

\(\mathcal{F} \leq \mathcal{F}'\) if and only if for all \((u, u', \tau)\) in \(\mathcal{F}\) there exists \((u, u', \tau)\) in \(\mathcal{F}'\) such that \(\tau \leq \tau'\), i.e. (see Section 2) such that \(\text{INVAR}(\tau') \subseteq \text{INVAR}(\tau)\).
(3) We know from Proposition 2.2 that every infinite chain: \( r_0 \leq r_1 \leq r_2 \leq \cdots \leq r_n \leq \cdots \) stabilizes, i.e., \( r_{n_0} \sim r_n \) for all \( n \) greater than some \( n_0 \). Hence we can define the least upper bound \( \text{Sup}_{i \geq 0}(F_i) \) of an increasing chain

\[
F_0 \leq F_1 \leq \cdots \leq F_n \leq \cdots
\]
as the set of facts \( (u, u', r) \) in \( F_i \) for some \( i \) which is the first integer such that, for all \( j \geq i \) and all substitution \( r' \) if \( (u, u', r') \in F_j \), then \( r' \sim r \).

(4) A set of facts \( F \) is reduced if for all pair \( (w, w') \) there exists at most one \( v \) such that \( (w, w', v) \) is in \( F \).

The least upper bound of an increasing sequence of reduced sets of facts is reduced. With a set of facts \( F \) we associate a set of equivalences, \( F = \{(r(u), r(u'))|(u, u', r) \in F\} \). Our algorithm can be sketched as follows:

Given \( E \), we define a finite set \( F_0 \) of facts such that if \( E \) is true, then \( F_0 \) is true. Then we define an increasing sequence \( (F_i)_{i \geq 0} \) of sets of facts such that \( F_{i+1} = \text{REFINE}(F_i) \), where \( \text{REFINE} \) is some procedure defined below. We shall show that one of the following two cases must happen:

(1) \( \text{REFINE} (F_i) \) stops with a FAILURE answer; we conclude then that \( E \) is not true,

(2) \( \text{REFINE}(F_i) = F_i \) for some \( i \); then \( F_i \) is self-proving and \( E \) is true if and only if \( F_i \vdash E \) (which is decidable).

The definition of \( \text{REFINE} \) will rest upon SPL defined in 3.10. (We shall use SPL with \( W = V \).)

We shall also use the following notation:

For substitutions \( \delta, \delta' : V \rightarrow M(G, V) \) we denote by \( \text{EQ}(\delta, \delta') \) the set of equations \( (\delta(v), \delta'(v)) \) for all \( v \) in \( V \) and by \( \text{EQ}(\delta) \) the set \( \text{EQ}(\delta, id) \), i.e., the set \( (\delta(v), v) \) for all \( v \) in \( V \). Hence \( \text{UNIF} \left( \text{EQ}(\delta) \right) = \text{INVAR}(\delta) \).

We are now ready to define \( \text{REFINE} \) which takes for argument a finite reduced set of facts.

4.2. DEFINITION.

Procedure \( \text{REFINE}(F) \)

1. Let \( \mathcal{D} = F \).
2. For all \( (w, w', v) \) in \( \mathcal{D} \) do
   2.1 Let \( \mathcal{E}_0 = \text{SPL}_0(\text{EXP}(w, w')) \) and \( \mathcal{E}_1 = \text{SPL}_1(\text{EXP}(w, w')) \)
   2.2 If \( \mathcal{E} \in \mathcal{E}_0 \) or if \( \text{UNIF}(\mathcal{E}_0 \cup \text{EQ}(v)) = \emptyset \), then exit \( \text{REFINE} \) with answer: FAILURE.
   2.3 Otherwise let \( v' \) be a base of \( \text{UNIF}(\mathcal{E}_0 \cup \text{EQ}(v)) \); if \( v \not\sim v' \), then modify \( F \) by replacing \( (w, w', v) \) by \( (w, w', v') \).
2.4 For all \((t, t')\) in \(\mathcal{F}\), do

2.4.1 Let \((u, u', \delta) = \text{REP}(t, t')\)

2.4.1 If \(\mathcal{F}\) does not contain any fact of the form \((u, u', \tau)\), then let \(\mathcal{F} = \mathcal{F} \cup \{(u, u', \text{id})\}\)

2.4.3 Otherwise let \((u, u', \tau)\) be already in \(\mathcal{F}\); let \(S = \text{UNIF}(\text{EQ}(\delta \circ \tau, \delta) \cup \text{EQ}(v))\); if \(S = \emptyset\), then exit \text{REFINE} with answer: FAILURE.

2.4.4 Otherwise let \(v'\) be a base of \(S\); if \(v' \not\sim v\), then modify \(\mathcal{F}\) by replacing \((w, w', v)\) by \((w, w', v')\).

3. Return \text{REFINE} = \mathcal{F}.

4. End \text{REFINE}.

The next three propositions state the properties of \text{REFINE} we shall use in the sequel.

4.3. Proposition. Let \(\mathcal{F}\) be a finite reduced set of facts.

(1) \text{REFINE}(\mathcal{F}) always terminates.

(2) If \text{REFINE}(\mathcal{F}) stops with a FAILURE answer, then \(\mathcal{F}\) is not true.

(3) If \text{REFINE}(\mathcal{F}) = \mathcal{F}', then \(\mathcal{F} \leq \mathcal{F}'\) and \(\mathcal{F}'\) is reduced. If \(\mathcal{F}\) is true, then \(\mathcal{F}'\) is true.

Proof. (1) Termination is easy to check.

(2) Let \((w, w', v)\) in \(\mathcal{F}\) causing a failure either at step 2.2 or 2.4.3. By assuming that \(\mathcal{F}\) is true we shall get a contradiction. Let \(\theta\) such that \(\theta \models w \equiv w'\). Then, for all \(p\), all \((z, z')\) in \(\text{EXP}(p, w, w')\) we have \(\theta \models z \equiv z'\). We cannot have \(\text{SPL}(z, z') = \{A\}\) since this would arise from the hypotheses of clause (3.10.3) of the definition of \(\text{SPL}\). For all \((z, z')\) in \(\text{SPL}(\text{EXP}(w, w'))\) we have \(\theta \models z \equiv z'\). We also know that \(\theta \circ v = \theta\) (since \((w, w', v)\) is assumed true), hence \(\text{UNIF}(\text{EQ}(v) \cup \emptyset) \neq \emptyset\). This eliminates the possibility of a failure at step 2.2. Assume now that \((t, t')\) in \(\text{SPL}(\text{EXP}(w, w'))\) causes a failure at step 2.4.3. It is clear that \(\theta \models t = t'\) hence \(\theta \models \delta(u) \equiv \delta(u')\) (where \((u, u', \delta) = \text{REP}(t, t')\)) in other words

\[\theta \circ \delta \models u \equiv u'.\]

Since \((u, u', \tau)\) is true this implies

\[\theta \circ \delta \circ \tau = \theta \circ \delta.\]
i.e., \( \theta \in \text{UNIF}(\text{EQ}(\delta \circ \tau, \delta)) \). Hence, since \( \theta \in \text{INVAR}(v) = \text{UNIF}(\text{EQ}(v)) \) the set \( S \) cannot be empty, and we get a contradiction.

(3) Let us assume that \( \text{REFINE}(\mathcal{F}) = \mathcal{F}' \). This means that no failure has occurred and that either \( \mathcal{F}' = \mathcal{F} \) or that \( \mathcal{F}' \) is the result of a series of modifications of \( \mathcal{F} \), say, \( \mathcal{F} = \mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k = \mathcal{F}' \).

Each modification of \( \mathcal{F}_i \) into \( \mathcal{F}_{i+1} \) is performed at steps 2.3, 2.4.2, or 2.4.5 and, in each case \( \mathcal{F}_i \leq \mathcal{F}_{i+1} \). Hence \( \mathcal{F} \leq \mathcal{F}' \). Similar arguments show that \( \mathcal{F}_{i+1} \) is reduced (resp. true) if \( \mathcal{F}_i \) is, hence that \( \mathcal{F}' \) is reduced (resp. true) if \( \mathcal{F} \) is.

Remark that if \( \text{First}(u) \in \Phi \) for all \((u, u', \tau)\) in \( \mathcal{F} \), then \( \mathcal{F} \) is a set of elementary equivalences.

4.4. PROPOSITION. Let \( \mathcal{F}_0 \) be a finite reduced set of true facts such that \( \text{First}(u) \in \Phi \) for all \((u, u', \tau)\) in \( \mathcal{F}_0 \). Let \( \mathcal{F}_0 \leq \mathcal{F}_1 \leq \mathcal{F}_2 \leq \cdots \leq \mathcal{F}_i \leq \cdots \) be an increasing chain of sets of facts such that \( \mathcal{F}_{i+1} = \text{REFINE}(\mathcal{F}_i) \). Let \( \mathcal{F} \) be its least upper bound.

(1) \( \mathcal{F} \) is a self-proving set of elementary equivalences.

(2) There exists \( i \) such that \( \mathcal{F}_j = \mathcal{F}_i \) for all \( j \geq i \) and \( \mathcal{F} = \mathcal{F}_i \).

Proof. From the definition of \( \text{REFINE} \), it is easy to see that if \( \text{First}(u) \in \Phi \) for all \((u, u', \tau)\) in \( \mathcal{F}_0 \), then the same holds for all \((u, u', \tau)\) in \( \mathcal{F}_i \) for all \( i \geq 0 \) hence for all element of \( \mathcal{F} \).

Hence \( \mathcal{F} \) is a set of monadic equivalences. We first show that \( \mathcal{F} \) is self-proving, i.e., since \( \mathcal{F} \subseteq \text{ASSERT}_{\exp} \) that \( \mathcal{F} \vdash \exp(\mathcal{F}) \).

Let \((w, w', v)\) be in \( \mathcal{F} \). We must show that \( \mathcal{F} \vdash \exp(v(w), v(w')) \).

Let \( A = \{(w, w')\} \cup \{(u, u')/\delta = \text{REP}(t, t') \text{ for some } (t, t') \in \text{SPL}_1(\exp(w, w'))\} \).

The set \( A \) is finite. There exists an integer \( i_A \) such that, for all \((u, u')\) in \( A \)

(1) there exists \((u, u', \tau)\) in \( \mathcal{F}_{i_A} \) for some \( \tau \),

(2) for all \( i > i_A \), the unique \( \tau' \) such that \((u, u', \tau') \in \mathcal{F}_i \) is equivalent to \( \tau \).

The existence of \( i_A \) is a consequence of Lemma 2.1 and definition (4.1.3).

We let \( \mathcal{F}' = \mathcal{F}_{i_A} \) and show

\( \mathcal{F}' \vdash \exp(v(w), v(w')) \).

Note that \((w, w', v) \in \mathcal{F}' \). By Lemma 3.12(3) it suffices to show that

(4) \( \text{SPL}_0(\exp(v(w), v(w'))) = \emptyset \), and
(5) \( \mathcal{F}' \vdash \text{SPL}_1(\exp(v(w), v(w'))) \).
Remark that (4) follows from

(6) \( v(t) = v(t') \) for all \((t, t')\) in \( \mathcal{E}_0 = \text{SPL}_0(\text{EXP}(w, w')) \).

But there exists \( v' \) such that \((w, w', v') \in \text{REFINE}(\mathcal{F}) \) and

(7) \( \text{INVAR}(v') = \text{UNIF}(\text{EQ}(v) \cup \mathcal{E}_0) \).

Property (2) implies \( v \sim v' \), i.e., \( \text{INVAR}(v) = \text{INVAR}(v') \) hence with (7)

(8) \( \text{INVAR}(v) \subseteq \text{UNIF}(\mathcal{E}_0) \)

which implies (6) and property (4) is proved.

Let us now consider \((t, t') \in \text{SPL}(v(w), v(w'))\), i.e., \( t = v(s), t' = v(s') \) for some \((s, s') \in \text{SPL}_1(\text{EXP}(w, w'))\). Let \((u, u', \delta) = \text{REP}(s, s') \) hence \((u, u') \in A\). Let \((u, u', \tau) \) be in \( \mathcal{F}' \). We know that there exists some \( v' \) such that \((w, w', v') \in \text{REFINE}(\mathcal{F}') \) and

(9) \( \text{INVAR}(v') = \text{UNIF}(\text{EQ}(v) \cup \text{EQ}(\delta \circ \tau, \delta)) \).

As above we conclude that \( v' \sim v \) hence that

(10) \( \text{INVAR}(v) \subseteq \text{UNIF}(\text{EQ}(\delta \circ \tau, \delta)) \).

This shows that for all term \( z \)

\[ v \circ \delta \circ \tau(z) = v \circ \delta(z), \]

hence in particular for \( u \) and \( u' \), whence

(11) \( \emptyset \vdash v \circ \delta \circ \tau(u) \equiv v \circ \delta(u), \)

(12) \( \emptyset \vdash v \circ \delta \circ \tau(u') \equiv v \circ \delta(u'). \)

On the other hand, since \((u, u', \tau) \in \mathcal{F}' \)

\[ \mathcal{F}' \vdash \tau(u) \equiv \tau(u'), \]

hence

(13) \( \mathcal{F}' \vdash v \circ \delta \circ \tau(u) \equiv v \circ \delta \circ \tau(u') \)

we can conclude from (11–13) that

\[ \mathcal{F}' \vdash v \circ \delta(u) \equiv v \circ \delta(u'), \]

i.e.,

(14) \( \mathcal{F}' \vdash t \equiv t' \)

since \( t = v(\delta(u)), t' = v(\delta(u)) \). This holds for all \((t, t') \) in \( \text{SPL}_1(\text{EXP}(v(w), v(w'))\). We have established (5) and this concludes the proof of part (1).

Let us now prove part (2). Since \( \mathcal{F} \) is a set of true equivalences (by Lemma 3.6). Note that for all \( i \), all \((u, u', \tau) \in \mathcal{F}_i \), \( \text{First}(u) \in \Phi \), (by induction on \( i \)), hence the same holds for \( \mathcal{F} \).

Since \( \mathcal{F}_0 \) is true, Proposition 4.3(3) shows that \( \mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_i, \ldots \), and finally \( \mathcal{F} \) are true (as sets of facts). From Remark 4.1(1) we can conclude
that every equivalence \((t, t')\) in \(\mathcal{F}\) is strongly true. Since we also have \(\text{First}(t) \in \Phi\), we finally have: \(\mathcal{F} \subseteq \mathcal{F} \text{ ELEM}\).

Then Proposition 3.9 shows that \(\mathcal{F}\) is finite. Hence there exists an integer \(i\) such that \(\mathcal{F} \subseteq \mathcal{F}_i\). Hence \(\text{REFINE}(\mathcal{F}_i) \subseteq \mathcal{F}_i\) and, by Propositions 4.3 and 4.4 \(\text{REFINE}(\mathcal{F}_i) = \mathcal{F}_i\). For all \(i' > i\), \(\mathcal{F}_i' = \mathcal{F}_i\) and \(\mathcal{F} = \mathcal{F}_i\).

Let us call \(\text{CLOSURE}\) the function which associates \(\mathcal{F}\) with \(\mathcal{F}_0\) as in Proposition 4.4. Hence \(\text{CLOSURE}\) is computable mapping. It is also total in the sense that we obtain an answer FAILURE within a finite time (when evaluating some \(\text{REFINE}(\mathcal{F}_i)\)) when \(\text{CLOSURE}(\mathcal{F}_0)\) is undefined.

4.5. COROLLARY. Let \(\mathcal{F}\) be a finite reduced set of facts such that \(\text{First}(u) \in \Phi\) for all \((u, u', \tau)\) in \(\mathcal{F}\). Then

1. \(\text{REFINE}(\mathcal{F}) = \mathcal{F}\) if and only if \(\mathcal{F}\) is self-proving.
2. \(\text{CLOSURE}(\mathcal{F})\) is a smallest set of facts \(\mathcal{F}'\) such that \(\mathcal{F} \subseteq \mathcal{F}'\) and \(\mathcal{F}'\) is self-proving, if there exists any such set \(\mathcal{F}'\).

Proof. Note that the proof of Proposition 4.4(1) does not assume that \(\mathcal{F}_0\) is true. The details are omitted. \(\square\)

We can now write our algorithm to check the truth of an equivalence \(E\) in \(M(F \cup G \cup \Phi, X \cup Y)^2\).

We let \(\text{RSPL}(E)\) denote \(\{\text{REP}(t, t')/(t, t') \in \text{SPL}(E)\}\).

4.6. ALGORITHM.

0. Input \(E\).

1. Let \(\mathcal{E}_0 = \text{SPL}_0(E)\) and \(\mathcal{E}_1 = \text{RSPL}(E)\)
2. If \(\mathcal{E}_0 = \text{SPL}_0(E)\) and \(\mathcal{E}_1 = \text{RSPL}(E)\)
2. If \(\mathcal{E}_0 \neq \emptyset\), then stop with answer: "\(E\) IS FALSE"
3. If \(\mathcal{E}_0 = \mathcal{E}_1 = \emptyset\), then stop with answer: "\(E\) IS TRUE"
4. Otherwise let \(\mathcal{E}_0 = \{(u, u', id)/(u, u', \delta) \in \mathcal{E}_1\) for some \(\delta\}\)
5. For all \(i \geq 1\) do
   5.1. Let \(\mathcal{F}_{i+1} = \text{REFINE}(\mathcal{F}_i)\)
   5.2. in case of FAILURE, then stop with answer: "\(E\) IS FALSE"
   5.3. as soon as \(\mathcal{F}_{i+1} = \mathcal{F}_i\) go to 6.
6. Check whether for all \((u, u', \delta)\) in \(\mathcal{E}_1\) there exist \((u, u', \tau)\) in \(\mathcal{F}_i\) such that \(\delta \in \text{INVAR}(\tau)\).
7. If true, then stop with answer: "\(E\) IS TRUE" else stop with answer: "\(E\) IS FALSE".
8. End.
4.7. THEOREM. Algorithm 4.6 says correctly whether $E$ is true.

Proof. Remark first that $F_0$ satisfies the hypotheses of Proposition 4.4. Hence, there exists some $i$ such that, either the computation of $\text{REFINE}(F_i)$ stops with a FAILURE or gives $\text{REFINE}(F_i) = F_i$. This shows the termination of Algorithm 4.6.

Propositions 4.3 and 3.12 show that the answers at steps 2, 3, and 5.2 of Algorithm 4.6 are correct. Hence we only have to consider what happens at step 7.

If the result of step 6 is \textit{true} this means that

$$\mathcal{F}_i \models \text{SPL} 1(E)$$

Since we know that $\text{SPL}_0(E) = \emptyset$ and $\mathcal{F}_i \subseteq E$ (by the proof of 4.4(2)), then Lemma 3.12(3) shows that $\mathcal{F} \models E$, i.e., that $E$ is true. The answer given at step 7 is correct.

Let us now assume that $E$ is true. This means that $\text{SPL}_1(E)$ is true. But for all $(u, u', \delta) = \text{REP}(t, t')$ for some $(t, t')$ in $\text{SPL}_1(E)$ there exist $(u, u', \tau)$ in $\mathcal{F}_i$. Since $\delta \vdash u = u'$ and $(u, u', \tau)$ is optimal (by 4.4(2) again), $\delta \in \text{INVAR}(\tau)$ and the result of step 6 is \textit{true}.

Hence we have shown that after step 7 the result of Algorithm 4.6 is "$E$ IS TRUE" if and only if $E$ is true. Hence our algorithm is correct. \[ \square \]

4.8. Remark. Slight modifications in Algorithm 4.6 allow to determine $\text{UNIF}(g_{t', t})$ for given $t, t'$ in $M(F \cup \Phi \cup \emptyset, X \cup V)$:

(1) Let $E = (t, t')$ and compute $g_0, g_1$ and $F_0$ as in 4.6.

(2) Perform loop 5 of Algorithm 4.6 and exit with answer "UNIF$(g_{t', t}) = \emptyset$" in case of FAILURE in the computation of some $\text{REFINE}(F_i)$.

(3) Having obtained $F_i$, at step 5.3 such that $F_i = \text{REFINE}(F_i)$; let $S = \text{SPL}_0(E) \cup \{\text{EQ}(\delta \circ \tau, \delta)/(u, u', \delta) \in \mathcal{F}_1 \land (u, u', \tau) \in \mathcal{F}_i\}$.

(4) Then $\text{UNIF}(g_{t', t}) = \text{UNIF}(S)$.

Since $\mathcal{F}_i$ in finite, $\text{UNIF}(S)$ can be computed by classical algorithms.

4.9. Example

In order to improve the readability, we shall use the following variants of our notations:

$u \equiv u'$ for an equivalence $(u, u')$,

$(u = u', \delta)$ for a fact $(u, u', \delta)$,

A set $\mathcal{E}$ of equations on $M(G, V)$ is denoted $\{t_1 = t'_1, t_2 = t'_2, \ldots, t_k = t'_k\}$, where $\mathcal{E} = \{(t_1, t'_1), \ldots, (t_k, t'_k)\}$.
Here is our example. Let $\eta$ consists of two sorts $S$ and $S'$. Let $x, x'$ be variables of sorts $S$ and $S'$, respectively. Let $\Sigma$ be the p.r. scheme given below, written with $P = \{p, q_1, q_2, r\}$ such that:

\begin{align*}
\alpha(p) &= S, & \sigma(p) &= S', \\
\alpha(q_1) &= \alpha(q_2) = S, & \sigma(q_1) &= \sigma(q_2) = S, \\
\alpha(r) &= e, & \sigma(r) &= S.
\end{align*}

It is non-nested and separated (with $F = \{k\}$ and $G = \{a, f, g\}$).

\begin{align*}
\varphi(p(x), y, z) &= \psi(x, y, z), \\
\psi(q_1(x), y, z) &= \psi(x, g(y, a), z), \\
\psi(q_2(x), y, z) &= \psi(x, g(g(y, z), a), z), \\
\psi(r, y, z) &= k(g(y, a), z), \\
\varphi'(p(x), y, z) &= \psi'(x, g(fy, a), z), \\
\psi'(q_1(x), y, z) &= \psi'(x, g(y, a), z), \\
\psi'(q_2(x), y, z) &= \psi'(x, g(g(y, a), z), z), \\
\psi'(r, y, z) &= k(y, z).
\end{align*}

We shall determine a base (if any) of $g', s'$ with $s = \varphi(x', v_1, v_2)$ and $s' = \varphi'(x', v_3, v_4)$, by computing $\text{CLOSURE} (\{(s \equiv s', \text{id})\})$.

The corresponding sequence $\langle \mathcal{F}_i \rangle_{i \geq 0}$ is the following:

\[ \mathcal{F}_0 = \{(s \equiv s', \text{id})\}, \quad \mathcal{F}_1 = \{(s \equiv s', \text{id})\} \cup \{(t \equiv t', \text{id})\}, \]

where

\[ t = \psi(x, v_1, v_2) \quad t' = \psi'(x, v_3, v_4), \]

\[ \mathcal{F}_2 = \{(s \equiv s', \text{id})\} \cup \{(t \equiv t', \tau)\}, \]

where

\[ \tau = \{v_2 \rightarrow a, v_3 \rightarrow g(v_1, a), v_4 \rightarrow a\}, \]

\[ \mathcal{F}_3 = \{s \equiv s', \sigma), (t \equiv t', \tau)\}, \]

where

\[ \sigma = \{v_1 \rightarrow f, v_3, v_2 \rightarrow a, v_4 \rightarrow a\} \quad \text{and} \quad \mathcal{F}_4 = \mathcal{F}_3. \]
Hence

\[
\text{CLOSURE}(s \equiv s', id) = \mathcal{F}_3.
\]

This means that for all \(u_1, u_2, u_3, u_4\),

\[
\models \varphi(x', u_1, u_2) \equiv \varphi'(x', u_3, u_4)
\]

if and only if \((u_1, u_2, u_3, u_4)\) satisfies \(\text{EQ}(\sigma)\), i.e., if and only if:

\[
u_1 = f u_3, \quad u_2 = a, \quad u_4 = a.\]

Hence

\[
\models \varphi(x', f g(y, z), a) \equiv \varphi'(x', g(y, z), a)
\]

and

\[
\not\models \varphi(x', a, a) \equiv \varphi'(x', a, a).
\]

5. Conclusions

We have not tried to reduce the equivalence problem for non-nested separated primitive recursive schemes to a decidable equivalence problem such that the equivalence of finite-turn DPDAs although we think that such a reduction is possible. We think that a direct algorithm as Algorithm 4.6 shows better the reasons for which our schemes have a decidable equivalence problem. Nevertheless we think that there are some deep relations between primitive recursive schemes and DPDAs (via the constructions of Courcelle [2]) and we omit the very technical proof of


Here are some conjectures and open problems.

5.2. Conjecture. The equivalence problem for non-nested primitive recursive schemes is solvable.

The hypothesis of separated-ness has been crucial for the proof of Proposition 3.1.4. We think that it can be lifted. Probably much more difficult is

5.3. Conjecture. The equivalence problem for separated strongly noncircular attribute systems is solvable.
We are also left with the following questions:

5.4. **Open Problem.** Are the equivalence problems for p.r. schemes and DPDAs interreducible?

5.5. **Open Problem.** Is the equivalence problem of attribute systems decidable?

Let us finally point out that our arguments, especially Propositions 3.10 and 4.4(1), could be formalized in formal proof systems similar to those introduced in DeBakker (1971), or Courcelle and Vuillemin (1976) or in the implemented system LCF (Gordon *et al.*, 1979).

This is not true for decidability results for program schemes obtained by reduction to decidable results in language theory, e.g., of Valiant (1974).

**References**


