ATTRIBUTE GRAMMARS AND
RECURSIVE PROGRAM SCHEMES II*

Bruno COURCELLE and Paul FRANCHI-ZANNETTACCI
U.E.R. de Mathématiques, Université de Bordeaux-1, 33405 Talence, France

Communicated by M. Nivat
Received April 1980
Revised October 1980

Abstract. We show that an attribute system can be translated (in a certain way) into a recursive program scheme if and only if it is strongly noncircular. This property introduced by Kennedy and Warren [20] is decidable in polynomial time. We obtain an algorithm to decide the equivalence problem for purely synthesized attribute systems.

This is the second part of a work which has been divided for editorial reasons. Sections 1.2, 3, 4 can be found in the first part. The numbering of theorems, propositions, etc. . . . indicates the section: Theorem 3.16 can be found in Section 3.

5. Primitive recursive schemes and attribute systems

For all strongly noncircular attribute system, a primitive recursive scheme of same type can be constructed which defines in every interpretation $D$ the functions $\varphi_{a,S}^{(D)}$ introduced in Section 3. Then we characterize the p.r. schemes defined in this way.

Convention. In nearly all this section, we shall assume that every attribute corresponds to at most one nonterminal. Say $\mathcal{N}_a = \{ \eta(a) \}$ for each $a$ in $A$. This is clearly not a loss of generality. We shall write $\varphi_a$ instead of $\varphi_{a,S}$ (since $S$ is determined by $a$).

5.1. Construction of $\Sigma(I)$

Let $(I, \gamma)$ be a strongly noncircular attribute system of type $(P, F)$ (we shall use the notations of Section 3). The argument selector $\gamma$ will be fixed once for all. We construct a scheme $\Sigma(I)$ as follows:

1. We take $Y = A^{(h)}$ as set of parameters.
2. For each $a$ in $A^{(s)}$ such that $\mathcal{S} = \eta(a)$ and $\gamma(a, S) = y^{(1)}y^{(2)}\cdots y^{(m)}$ we introduce a function variable $\varphi_a$ of arity $S^{y^{(1)}y^{(2)}\cdots y^{(m)}}$ and of sort $a$. We let $\Phi$ denote the set of all such function variables.

* This work has been supported by the A.T.P. contract 4275 of C.N.R.S. It has been presented at the 21st Symposium on Foundations of Computer Science, Syracuse, New York, October 1980.
(3) For each \( \varphi_p \) in \( \Phi \) and each \( p \) in \( P \) of arity \( S_1 S_2 \cdots S_n \) and of sort \( S \) we shall define an equation

\[
\varphi_p(p(x_1, \ldots, x_n), y^{(1)}, \ldots, y^{(m)}) = \tau
\]

for some distinct variables \( x_1, \ldots, x_n \) of sort \( S_1, \ldots, S_n \) respectively and for some \( \tau \) in \( M(F \cup \Phi, \{x_1, \ldots, x_n, y^{(1)}, \ldots, y^{(m)}\}) \).

The set of all such equations will form our primitive recursive scheme with parameters. It is of type \( \langle P, F \rangle \). Let us now explain how we construct \( \tau \) in (5.1.1). To do so, we consider the semantic rule of \( \Gamma_p \) defining \( a(e) \). We assume it to be of the form

\[
a(e) = s[, \ldots, z(e), \ldots, b(j), \ldots]
\]

for some inherited attributes \( z \) in \( A^{(h)}_S \), for some \( j \)'s in \( [n] \), some synthesized attributes \( b \) in \( A^{(s)}_S \), and for some \( F \)-term \( s \).

**Fact 1.** \( z \) belongs to \( \{y^{(1)}, \ldots, y^{(m)}\} \).

Since (5.1.2) implies \( a(e) \rightarrow z(e) \). Hence \( z \in \gamma(a, S) = \{y^{(1)}, \ldots, y^{(m)}\} \).

**Fact 2.** \( b(j) \) can be 'defined' by some term \( \tau_{b,j} \) in \( M(F \cup \Phi, \{x_1, \ldots, x_n, y^{(1)}, \ldots, y^{(m)}\}) \).

Assuming for a while that Fact 2 has been proved and that we have defined all the \( \tau_{b,j} \) for all the \( b(j) \)'s occurring in the right-hand side of (5.1.2), we can define the term \( \tau \) as follows (where \( s \), the \( z \)'s and the \( (b, j) \)'s are as in (5.1.2)): \( \tau = s[, \ldots, z, \ldots, \tau_{b,j}, \ldots] \).

Now (5.1.1) looks like:

\[
\varphi_p(p(x_1, \ldots, x_n), y^{(1)}, \ldots, y^{(m)}) = s[, \ldots, z, \ldots, \tau_{b,j}, \ldots].
\]

**Proof of Fact 2.** Let us define a set \( \text{DEF} \) of values of the form \( c(k) \) for \( k \) in \( [n] \) and \( c \) in \( A_{S_k} \) that can be defined by some term \( \tau_{c,k} \) in \( M(F \cup \Phi, \{x_1, \ldots, x_n, y^{(1)}, \ldots, y^{(m)}\}) \). The set \( \text{DEF} \) is the union of an increasing sequence \( \langle \text{DEF}_i \rangle_{i \geq 0} \). We define \( \text{DEF}_i \) and \( \tau_{c,k} \) for all \( c(k) \) in \( \text{DEF}_i \) by induction on \( i \):

1. \( \text{DEF}_0 = \emptyset \).
2. \( \text{DEF}_{i+1} \) is \( \text{DEF}_i \) augmented with the set of \( c(k) \)'s satisfying one of the following exclusive conditions:
   - \( c \) is synthesized, \( \gamma(c, S_k) = z^{(1)} \ldots z^{(m)} \) and each of \( z^{(1)}(k), \ldots, z^{(m)}(k) \) belongs to \( \text{DEF}_i \), in this case we define:
     \[
     \tau_{c,k} = \varphi_c(x_k, \tau_{z^{(1)}, k}, \ldots, \tau_{z^{(m)}, k})
     \]
     (recall that \( \varphi_c \) stands for \( \varphi_{c,S_k} \) from our initial convention).
(ii) $c$ is inherited: the semantic rule of $I_p$ which defines $c(k)$ is

$$c(k) = s'[\ldots, z(e), \ldots, d(j'), \ldots]$$

for some $F$-term $s'$,
for some $z$'s in $\{y^{(1)}, \ldots, y^{(m)}\}$,
for some $j$'s in $[n]$ and some $d$'s in $A_{S_i}^{(s)}$
such that $d(j') \in \text{DEF}_i$;

in this case we define:

$$\tau_{c,k} = s'[\ldots, z, \ldots, \tau_{d,j'}, \ldots].$$

(5.1.5)

Going back to the question we started with, we need only show that $b(j)$ belongs to
$\text{DEF} = \bigcup_{i \geq 0} \text{DEF}_i$. Let us recall that we denote by $W_1(p)$ the set $\{b(i) | 1 \leq i \leq \rho(p)\}$ and $b \in A_{S_i}$. Let us show that $W' = \{w \in W_1(p) | b(j) \to^*_{p,\gamma} w \text{ and } w \notin \text{DEF}\}$ is empty (see Definition 3.12 for $\to_{p,\gamma}$). Assume the contrary. Since $W_1(p)$ is finite and $\to_{p,\gamma}$ has no cycle ($\gamma$ has been assumed noncircular) there exists some $c(k)$ in $W'$ such that $c(k) \to_{p,\gamma} w$ for no $w$ in $W'$.

Either we are in case (i) and some of the $z(i')^{(k)}$ is not in $\text{DEF}$. But $c(k) \to_{\gamma} z(i')^{(k)}$ hence $b(j) \to^*_{p,\gamma} c(k)$ and $z(i')^{(k)} \in W'$ contradicting our choice of $c(k)$.

Or we are in case (ii). At least one of the following two cases must occur:

(a) $z$ is not in $\{y^{(1)}, \ldots, y^{(m)}\}$ for some $z(e)$ occurring in the right-hand side of

(5.1.4). Hence we have

$$a(e) \to_{p} b(j) \to_{p,\gamma} c(k) \to_{p} z(e)$$

which implies $z \in \gamma(a, S) = \{y^{(1)}, \ldots, y^{(m)}\}$ since $\gamma$ is closed. Contradiction.

(b) $d(j')$ is not in $\text{DEF}$, hence $d(j') \in W'$ and this contradicts our choice of $c(k)$.

Hence $W' = \emptyset$. Since $b(j) \to^*_{p,\gamma} b(j)$ this implies that $b(j)$ is in $\text{DEF}$. This concludes the proof of Fact 2 and the construction of $\Sigma(I)$.

5.2. Theorem. Let $I$ be a strongly noncircular attribute system of type $(P, F)$. One can construct a primitive recursive scheme with parameter $\Sigma(I)$, of same type which defines the functions $\varphi_{a,\gamma}$ in every $F$-interpretation $D$.

Proof. The system of equations $\Sigma(I)$ defined in construction (5.1) is clearly a p.r. scheme with parameters of type $(P, F)$. Since for every $F$-interpretation $D$ such a system has a unique solution, we need only show that the functions $\varphi_{a,\gamma}^{(D)}$ satisfy $\Sigma(I)$ to achieve the proof.

This fact follows indeed from construction (5.1).

Let $a, S, p$ be as in (5.1), let $t$ be a tree of the form $p(t_1, \ldots, t_n)$. Let $\bar{y}^{(1)}, \ldots, \bar{y}^{(m)}$ be values given in the inherited attributes $y^{(1)}, \ldots, y^{(m)}$ at the root of $t$. Let $\bar{\varphi}_a = \varphi_{a}^{(D)}$
for all $\phi_a$ in $\Phi$. As in Definition 4.12 we shall consider $M(P) \oplus D$, the $(P \cup F \cup \Phi)$-magma associated with $(\phi_a)_{a \in \Phi}$. In particular we shall use

$$\text{derop}_{M(P) \oplus D} : M(P \cup F \cup \Phi, \{x_1, \ldots, x_n, y^{(1)}, \ldots, y^{(m)}\})_a$$

$$\to [M(P)_{\sigma(x_1)} \cdots \sigma(x_n)] \times D_{y^{(1)}} \times \cdots \times D_{y^{(m)}}, D_a].$$

Using the inductive construction of the $\tau_{c,k}$'s, one can show that

$$\text{derop}_{M(P) \oplus D}(\tau_{c,k})(t_1, \ldots, t_n, \tilde{y}^{(1)}, \ldots, \tilde{y}^{(m)}) = c_{D}(t)(\tilde{y}^{(1)}, \ldots, \tilde{y}^{(m)})$$

for all $c(k)$ in DEF.

From the construction of equation (5.1.1), one finds that

$$\text{derop}_{M(P) \oplus D}(\tau)(t_1, \ldots, t_n, \tilde{y}^{(1)}, \ldots, \tilde{y}^{(m)}) = a_{D}(i)(\tilde{y}^{(1)}, \ldots, \tilde{y}^{(m)})$$

$$= \phi_{a}^{(D)}(p(t_1, \ldots, t_n), \tilde{y}^{(1)}, \ldots, \tilde{y}^{(m)})$$

where $\tau$ is the right-hand side of $\Sigma_{p,\phi_a}$.

This proof holds for arbitrary $t_1, \ldots, t_n$ and $\tilde{y}^{(1)}, \ldots, \tilde{y}^{(m)}$ of appropriate sorts. Hence (5.1.1) is satisfied by the functions $\phi_{a}^{(D)}$'s. The same holds for all equations in $\Sigma(\Gamma)$. This shows that for all $a$ and $D$, the function $\phi_{a}^{(D)}$ defined by (3.23) coincides with $\phi_{a,D}$ defined by $\phi_a$ and $\Sigma(\Gamma)$ in the interpretation $D$. Similarly, $\phi_{a}^{(F)}$ (defined in (3.7)) coincides with $\phi_{a,F}$ associated with $\phi_a$ and $\Sigma(\Gamma)$ by (4.16).

5.3. Example. We define a strongly noncircular attribute system $\Gamma$ and construct $\Sigma(\Gamma)$ to illustrate construction (5.1).

Let $\mathcal{N} = \{S\}$ and $P$ be the ranked ($\mathcal{N}$-sorted) signature $\{p, q, r\}$ with $\rho(p) = 2$, $\rho(q) = \rho(r) = 0$.

Let $A^{(a)} = \{a, b\}$ and $A^{(b)} = \{y, z\}$. Let $F = \{f_1, f_2, f_3, f_4, f_5, f_6, g_1, g_2, h_1, h_2\}$ be a one-sorted signature. We define $\Gamma$ as follows:

For $p$ (of rank 2) we define:

1. $a(e) = f_1(b(2))$,
2. $b(e) = g_1(a(1), a(2))$,
3. $y(1) = f_2(y(e))$,
4. $z(1) = g_2(y(e), a(2))$,
5. $y(2) = f_3(z(e))$,
6. $z(2) = f_4(b(1))$;

for $q$ (of rank 0) we define:

$$a(e) = f_5(z(e)),$$
$$b(e) = h_1;$$

for $r$ (of rank 0) we define:

$$a(e) = h_2,$$
$$b(e) = f_6(y(e)).$$
The dependence graphs are shown in Fig. 5.1 with the same conventions as in Example 3.20.

Let us construct the sequence \((R_i)_{i \geq 0}\):

\[ R_0 = \emptyset, \quad R_1 = \{(y, b, S), (z, a, S)\}, \quad R_2 = R_1. \]

It is easy to check that the corresponding mapping \(\gamma_0\) is noncircular.

Hence we shall use \(\Phi = \{\varphi_a, \varphi_b\}\). From the semantic rules associated with \(q\) and \(r\) we get

\[
\varphi_a(q, z) = f_5(z), \quad \varphi_b(q, y) = h_1, \\
\varphi_a(r, z) = h_2, \quad \varphi_b(r, y) = f_6(y).
\]

Let us write the equations of \(\Sigma(\Gamma)\) associated with \(p\). The equation \(\Sigma(\Gamma)_{p, \varphi_a}\) will be

\[
\varphi_a(p(x_1, x_2), z) = f_1(\tau_{b, 2}) \quad \text{by (1)}
\]

where

\[
\tau_{b, 2} = \varphi_b(x_2, \tau_{y, 2}) \quad \text{and} \quad \tau_{y, 2} = f_3(z) \quad \text{by (5)}.
\]

Hence, finally, we get

\[
\varphi_a(p(x_1, x_2), z) = f_1(\varphi_b(x_2, f_3 z)).
\]

The equation \(\Sigma(\Gamma)_{p, \varphi_b}\) will be

\[
\varphi_b(p(x_1, x_2), y) = g_1(\tau_{a, 1}, \tau_{a, 2}) \quad \text{by (2)}
\]

with

\[
\tau_{a, 1} = \varphi_a(x_1, \tau_{z, 1}), \\
\tau_{a, 2} = \varphi_a(x_2, \tau_{z, 2}), \\
\tau_{z, 1} = g_2(y, \tau_{a, 2}) \quad \text{by (4)}, \\
\tau_{z, 1} = f_4(\tau_{h, 1}) \quad \text{by (6)}, \\
\tau_{h, 1} = \varphi_b(x_1, \tau_{c, 1}), \\
\tau_{c, 1} = f_2(y) \quad \text{by (3)}.
\]

Hence, we obtain

\[
\varphi_b(p(x_1, x_2), y) = g_1(\varphi_a(x_1, g_2(y, \varphi_a(x_2, f_4 \varphi_b(x_1, f_3 y))))), \varphi_b(x_2, f_4 \varphi_b(x_1, f_2 y))
\]


(We have omitted parentheses surrounding arguments of monadic function symbols for readability).

5.4. Examples. We try to apply construction (5.1) to the weakly noncircular attribute system $\Gamma$ of Example 3.20. Let us assume that the semantic rules of $\Gamma$ are:

\[
\begin{align*}
\Gamma_p: & \quad \begin{cases}
    a(e) = c(1), \\
    y(1) = g_1(u(e), c(1)), \\
    z(1) = h(1);
\end{cases} \\
\Gamma_q: & \quad \begin{cases}
    b(e) = h_1, \\
    c(e) = f_1(z(e));
\end{cases} \\
\Gamma_r: & \quad \begin{cases}
    b(e) = f_2(y(e)), \\
    c(e) = h_2.
\end{cases}
\end{align*}
\]

We let $\Phi = \{\varphi_a, \varphi_b, \varphi_c\}$ and we obtain immediately:

\[
\begin{align*}
\varphi_b(q, y) &= h_1, \\
\varphi_b(r, y) &= f_2(y), \\
\varphi_c(q, z) &= z, \\
\varphi_c(r, z) &= h_2.
\end{align*}
\]

We only need to define $\Sigma(\Gamma)_p, \varphi_a$:

\[
\begin{align*}
\varphi_a(p(x_1), u) &= \varphi_c(x_1, \tau_{z,1}), \\
\tau_{z,1} &= \varphi_b(x_1, \tau_{y,1}), \\
\tau_{y,1} &= g_1(u, \tau_{c,1}), \\
\tau_{c,1} &= \varphi_c(x_1, \tau_{z,1}).
\end{align*}
\]

Hence we need $\tau_{z,1}$ such that $\tau_{z,1} = \varphi_b(x_1, g_1(u, \varphi_c(x_1, \tau_{z,1})))$; this is impossible (unless we use the least fixed point operator $\mu$ as in [3]). Hence construction (5.1) fails for $\Gamma$ because of the circularity

\[
z(1) \rightarrow p b(1) \rightarrow c(1) \rightarrow p c(1) \rightarrow y_0 z(1)
\]

which precisely causes $y_0$ not to be noncircular.

For the second system $\Gamma'$ of (3.20) we obtain: $\tau_{y,1} = g_1(u)$ and finally

\[
\varphi_a(p(x_1), u) = \varphi_c(x_1, \varphi_b(x_1, g_1(u))).
\]

Note that $\varphi_a$ never depends on $u$:

- if $x_1 = q$, then $\varphi_a(p(q), u) \rightarrow \varphi_c(q, \varphi_b(q, g_1(u))) \rightarrow h_1$;
- if $x_1 = r$, then $\varphi_a(p(r), u) \rightarrow \varphi_c(r, \varphi_b(r, g_1(u))) \rightarrow h_2$.

We are aiming to characterize the schemes associated with strongly noncircular attribute systems. We first consider the special case of schemes associated with purely
synthesized systems i.e. attribute systems such that $A^{(h)} = \emptyset$. Note that purely synthesized systems are always strongly noncircular (Proposition 3.22).

5.5. Proposition. The class of primitive recursive schemes of type $(P, F)$ coincides with the class of schemes associated with purely synthesized attribute systems of same type.

Proof. It is obvious that $\Sigma(\Gamma)$ has no parameter if $\Gamma$ is purely synthesized. We give the proof of the converse as a preparation for the proof of Theorem 5.10 which generalizes the present proposition and is more difficult.

Let $\Sigma$ be a p.r. scheme of type $(P, F)$ with a set $\Phi$ of function variables.

We construct $\Gamma$ by taking $A^{(h)} = \emptyset$ and $A^{(s)} = \Phi$. For $a$ in $\Phi$ we define $\mathcal{N}_a = \{a(a)\}$ and $a = \sigma(a)$. With each equation $\Sigma_{a,p}$ in $\Sigma$ of the form:

$$a(p(x_1, \ldots, x_n)) = s[\ldots, b(x_i), \ldots]$$

(where $s$ is an $F$-term and the $b$'s are in $\Phi$, i.e. in $A^{(s)}$) we associate the semantic rule

$$a(e) = s[\ldots, b(i), \ldots]$$

and put it in $\Gamma_P$.

In this way, we build a purely synthesized attribute system $\Gamma$. By mapping $a$ in $\Phi$ onto $\phi$, one establishes an isomorphism, (i.e. a renaming of function variables) between $\Sigma$ and $\Sigma(\Gamma)$.

In order to characterize the schemes associated with attribute systems in the general case, we need a definition.

5.6. Definition. A p.r. scheme with parameters, of type $(P, F)$ is well-presented$^1$ if the following two conditions hold.

(5.6.1) For any two equations of $\Sigma$,

$$\phi(p(x_1, \ldots, x_n), y^{(1)}, \ldots, y^{(m)}) = \tau,$$

$$\phi(q(x_1, \ldots, x_n), z^{(1)}, \ldots, z^{(m)}) = \tau'$$

the parameter lists $\langle y^{(1)}, \ldots, y^{(m)} \rangle$ and $\langle z^{(1)}, \ldots, z^{(m)} \rangle$ are the same. In other words, there exists a mapping $\text{PAR}: \Phi \rightarrow Y^*$ such that

$$\text{PAR}(\phi) = y^{(1)}y^{(2)} \ldots y^{(m)}$$

for all equation $\Sigma_{p,\phi}$ written as above. We shall denote by $\text{PAR}_i(\phi)$ the $j$th element of $\text{PAR}(\phi)$.

(5.6.2) For any two equations of $\Sigma$,

$$\phi(p(x_1, \ldots, x_n), y^{(1)}, \ldots) = \cdots \psi(x_{n_1}, \nu_1, \nu_2, \ldots, \nu_m) \cdots,$$

$$\phi'(p(x_1, \ldots, x_n), y'^{(1)}, \ldots) = \cdots \psi'(x_{n_1}, \nu'_1, \nu'_2, \ldots, \nu'_m) \cdots$$

$^1$ Note: with respect to the problem in question! Nothing more.
(our notation means that \(\psi(x_i, v_1, \ldots, v_m)\) is a subterm of the right-hand side of \(\Sigma_{\varphi, \varphi'}\) and similarly for \(\psi'(x_i, v'_1, \ldots, v'_m)\), if \(z\) is a parameter such that \(z = \text{PARi} = \text{PARi}'(\psi')\), then \(v_j = v'_j\). In other words, there exists a partial mapping \(\mu : P \times N^+ \times Y \rightarrow M((F \cup \Phi, X \cup Y))\) such that:

\[
\forall j \in [m] \quad v_i = \mu(p, i, \text{PARi}(\psi))
\]

for all equations \(\Sigma_{\varphi, \varphi'}\) written as above.

Note that \(\varphi\) and \(\varphi'\) may be the same. In this case our conditions concern the subterms of the right-hand side of a same equation. Note also that \(\psi\) and \(\psi'\) may be the same. In this case our second condition implies that \(v_j = v'_j\) for all \(j\), i.e. that the subterms \(\psi(x_i, v_1, \ldots, v_m)\) and \(\psi'(x_i, v'_1, \ldots, v'_m)\) are the same.

Finally, a p.r. scheme with parameters is well-presentable if it can be transformed into a well-presented scheme by a renaming of the parameters in each equation. Clearly this transformation does not modify the functions computed by the scheme.

### 5.7. Remarks

1. Any p.r. scheme without parameters is well-presented. Since the set \(Y\) is empty, conditions (5.6.1) and (5.6.2) are trivially true.

2. Schemes containing equations of the forms

\[
\varphi(p(x_1, x_2), y) = \cdots \varphi(x_1, \ldots, \varphi(x_1, A) \cdots) \cdots
\]

or

\[
\varphi(p(x_1, x_2), y) = \cdots \varphi(x_1, h(y)) \cdots \varphi(x_1, k(y)) \cdots
\]

are certainly not well-presentable. The value \(\mu(p, 1, y)\) cannot be defined to satisfy (5.6.2). In the first case we need

\[
\mu(p, 1, y) = \mu(p, 1, y) = A = \cdots \varphi(x_1, A) \cdots
\]

and in the second case we need

\[
\mu(p, 1, y) = h(y) = k(y).
\]

3. Whether a given scheme is equivalent to some well-presentable scheme will be left as an open question.

4. The value of \(\mu(p, i, y)\) has a meaning only if \(i\) belongs to \([n]\) and \(y\) to \(\text{PAR}(\psi)\) for some \(\psi\) such that \(\beta(\psi) = S\) where \(\alpha(p) = S_1S_2 \cdots S_n\).

### 5.8. Example

Let us consider the following p.r. scheme \(\Sigma:\)

1. \(\psi(p(x_1, x_2), y) = f_1 \theta(x_2, f_3 y),\)
2. \(\psi(q, y) = f_3 y,\)
3. \(\psi(r, y) = h_2,\)
4. \(\theta(p(x_1, x_2), y) = g_1(\psi(x_1, g_2(y, \psi(x_2, f_4 \theta(x_1, f_2 y))))), \psi(x_2, f_4 \theta(x_1, f_2 y))),\)
5. \(\theta(q, y) = h_1,\)
6. \(\theta(r, y) = f_3 y.\)

Condition (5.6.1) is obviously satisfied.
Let us now check condition (5.6.2) by constructing the mapping \( \mu \). We shall only have to define \( \mu(p, 1, y) \) and \( \mu(p, 2, y) \). We get

\[
\mu(p, 1, y) = g_2(y, \psi(x_2, f_4 \theta(x_1, f_2 y))) \quad \text{from (4)}
\]

\[
= f_2 y \quad \text{from (4)};
\]

\[
\mu(p, 2, y) = f_3 y \quad \text{from (1)}
\]

\[
= f_4 \theta(x_1, f_2 y) \quad \text{from (4)}.
\]

Hence \( \mu \) is not a mapping.

Let us now rename \( y \) into \( z \) in equations (1), (2) and (3). This gives us a scheme \( \Sigma' \).

In particular, equation (1) becomes:

\[
(1') \quad \psi(p(x_1, x_2), z) = f_1 \theta(x_2, f_3 z).
\]

Now, we have to define \( \mu(p, i, y) \) and \( \mu(p, i, z) \) for \( i = 1, 2 \). We find:

\[
\mu(p, 1, y) = f_2 y \quad \text{from (4)},
\]

\[
\mu(p, 2, y) = f_3 z \quad \text{from (1')},
\]

\[
\mu(p, 1, z) = g_2(y, \psi(x_2, f_4 \theta(x_1, f_2 y))) \quad \text{from (4)}
\]

\[
\mu(p, 2, z) = f_4 \theta(x_1, f_2 z) \quad \text{from (4)}.
\]

Hence our new scheme is well-presented. The former is well-presentable. (The reader will note that \( \Sigma' \) is isomorphic to \( \Sigma(\Gamma) \) constructed in Example 5.3.

5.9. Proposition. Let \( \Sigma \) be a primitive recursive scheme with parameters. There exists an algorithm to decide whether \( \Sigma \) is well-presentable.

Proof. Let \( \Sigma \) be given. Let us first rename the variables of \( Y \) in all equation of \( \Sigma \) in such a way that:

- sorts are preserved,
- condition (5.6.1) is satisfied,
- for any two function variables \( \varphi \) and \( \psi \neq \varphi \), there is no variable common to \( \text{PAR}(\varphi) \) and \( \text{PAR}(\psi) \).

This is clearly possible (we can assume that \( Y \) contains sufficiently many variables).

We start now constructing the various \( \mu(p, i, y) \) (as in Example 5.8). As soon as we are faced with a pair of equalities:

\[
(1) \quad \mu(p, i, y) = \nu, \quad \mu(p, i, y) = \nu' \quad \text{such that } \nu' \neq \nu,
\]

we consider the following two possibilities:

(i) \( \nu' = \nu[y^{(i1)} / y^{(1)}, \ldots, y^{(im)} / y^{(m)}] \) with \( y^{(i1)} \neq y^{(ij)} \) for all \( i \neq j \) (note that this implies that \( y^{(i1)} \) and \( y^{(i)} \) are of the same sort). Then we substitute \( y^{(i1)} \) for \( y^{(i)} \) for all \( i \) in \( [m] \) everywhere in \( \Sigma \). Note that the new scheme satisfies (5.6.1). Then we repeat
from the beginning the construction of $\mu$ for the new scheme and the search for a pair of equalities as (1) above.

(ii) If case (i) does not hold, then we can stop: the scheme we started with is not well-presentable.

If no pair of equalities of the form (1) can be found, then this means that (5.6.2) is satisfied and that we have obtained a well-presented scheme, and, more precisely a renaming of the variables of the initial scheme showing it to be well-presentable.

The proof of the correctness of this algorithm is left to the reader.

5.10. Theorem. A primitive recursive scheme $\Sigma$ is well-presented if and only if it is isomorphic to $\Sigma(\Gamma)$ for some strongly noncircular attribute system of same type.

Proof. Let us first show that the system $\Sigma(\Gamma)$ associated with an attribute system $\Gamma$ is well-presented.

Let us consider an equation of $\Sigma(\Gamma)$ of the form:

$$\varphi(p(x_1, \ldots , x_n), y^{(1)}, \ldots , y^{(m)}) = \tau.$$ 

Note first that $\text{PAR}(\varphi(a)) = y^{(1)} \cdots y^{(m)} = \gamma(a, \eta(a));$ whence (5.6.1) holds. Then we have to consider subterms of $\tau$ of the form $\varphi_c(x_i, \nu_1, \ldots , \nu_{m_i})$ (not that $\eta(c) = S_i$ where $\alpha(p) = S_1, \ldots , S_n$). Such a subterm corresponds to a variable $c(i)$ and is denoted by $\tau_{c,i}$ in construction (5.1.). Let us take $\gamma(c, \eta(c)) = z^{(1)} \cdots z^{(m_i)}$. Then $v_j$ is of the form $\tau_{p^k,i}$ for all $j$ in $[m']$. It follows that $v_j$ depends only on $p_i$ and $x^{(i)}$. This fact establishes (5.6.2). Hence we have shown that $\Sigma(\Gamma)$ is well-presented. Conversely, let $\Sigma$ be a well-presented primitive recursive scheme of type $(P, F)$ with a set $\Phi$ of function variables.

Let us define an attribute system $\Gamma$ as follows:

(1) Let $A^{(s)} = \Phi$; for each $a$ in $\Phi$ let us define $a = \sigma(a)$ (in $\mathcal{A}$) and $N_a = \{\beta(a)\}$.

(2) Let us now define $A^{(b)} = Y$ (the set of parameters) and for each $y$ in $Y$ let us define $y = \sigma(y)$ (in $\mathcal{A}$) and $N_y = \{S \in N | y$ appears in $\text{PAR}(a)$ for some $a$ in $A^{(s)}\}$. Note that if $\Sigma$ is well-presented after application of the algorithm of Proposition 5.9 $\text{PAR}(\varphi)$ and $\text{PAR}(\varphi')$ are disjoint for any two function variables $\varphi$ and $\varphi'$ such that $\sigma(\varphi) \neq \sigma(\varphi')$. In this case $N_y$ is singleton.

(3) We shall now define the semantic rules of $\Gamma$. Let us fix some $p$ in $P$ and define $\Gamma_p$. Let us consider an equation $\Sigma_{a,p}$ for some $a$ in $\Phi$, assumed to be of the form:

$$a(p(x_1, \ldots , x_n), y^{(1)}, \ldots , y^{(m)}) = s[\mu_1/v_1, \ldots , \mu_k/v_k]$$

for some $s$ in $\mathcal{M}(F, V_k)$ and some terms $\mu, \nu$ each of them being of the forms:

($\alpha$) either $y^{(i)}$ for some $i$ in $[m]$; in this case we define $\mu_i = y^{(i)}(e)$,

($\beta$) or $b(x_i, \nu_1, \ldots , \nu_{m_i})$ for some $b$ in $\Phi$; in this case we define $\mu_i = b(i)$.

From all this we construct the following semantic rule and put it in $\Gamma_p$:

$$a(e) = s[\mu_1/v_1, \ldots , \mu_k/v_k].$$

By doing the same for all $a$ in $A^{(p)}$ and all $p$ in $P$ we obtain semantic rules defining the synthesized attributes.
Going back to the equation $\Sigma_{a,p}$ (see (5.10.1)), let us consider all the subterms of its right-hand side which are of the form $b(x_0, \nu_1, \ldots, \nu_m)$ for some fixed $i$. Each of the $\nu_j$'s can be written:

$$\nu_j = s[\mu_1/\nu_1, \ldots, \mu_k/\nu_k]$$

where $s, \mu_1, \ldots, \mu_k$ are as in (5.10.1).

We construct $\hat{\mu}_1, \ldots, \hat{\mu}_k$ exactly as above, using clauses (a) and (b), and we obtain the semantic rule

$$y^{(r)}(i) = s[\hat{\mu}_1/\nu_1, \ldots, \hat{\mu}_k/\nu_k]$$

and put it in $\Gamma_p$.

Note that we put it in $\Gamma_p$ since we started from an equation $\Sigma_{a,p}$ that $i$ comes from the consideration of $b(x_0, \nu_1, \ldots, \nu_m)$ and that $j$ is the rank of the subterm, namely $\nu_n$, from which we have really defined our semantic rule. This is illustrated in Fig. 5.2 where the elements of $\Sigma_{a,p}$ which are relevant to the construction of (5.10.4) are circled with dots.

![Fig. 5.2.](image)

For each $p, i$ and $j$ we must have at most one semantic rule in $\Gamma_p$ defining $y^{(r)}(i)$. But the existence of a mapping $\mu$ satisfying condition (5.6.2) guarantees that the right-hand side of (5.10.3) is $\mu(p, i, z)$ where $z = \text{PAR}_i(b)$; hence that the right-hand side of (5.10.3) is associated in a unique way with $(p, i, j)$. Hence there exists at most one semantic rule in $\Gamma_p$ defining $y^{(r)}(i)$ in $\Gamma_p$.

By Definition 3.1 we need also at least one such semantic rule. in $\Gamma_p$ for all $i \in [\rho(p)]$ and all $y^{(r)}$ in $A_{\nu(i,p)}^{(h)}$. And this is not necessarily the case yet from our construction.

In fact, if there is no semantic rule in $\Gamma_p$ defining $y(i)$ in $\Gamma_p$, we simply put anything instead, say

$$y(i) = B$$

for some constant $B$ in $F$ of sort $y$. 

---

*Attribute grammars and recursive program schemes*
These artificial semantic rules will never be used to compute any synthesized attribute in any situation. An example will be presented below (see Example 5.12).

Let us finally note that for all $a$ in $A'$ and all $p$ of sort $\sigma(a)$ we have one and only one semantic rule in $\Gamma$, defining $a(\varepsilon)$. Hence we have obtained an attribute system $\Gamma$ of type $(P, F)$.

We must now show that $\Gamma$ is strongly noncircular. Let us take for $\gamma$ the mapping $\text{PAR}$, namely:

$$\gamma(a, \mathcal{N}(a)) = \text{PAR}(a)$$

for $a$ in $A'(= \Phi)$, otherwise the value of $\gamma$ is of no interest.

Let us consider some $p$ in $P$ and the relation $\rightarrow^{\mathcal{E}}_{\gamma}$ on $W(p)$ defined in (3.1).

**Claim.** For all $w$ in $W(p)$ and all $y$ in $A^{(\mathcal{E})}$, if $w \rightarrow^{\mathcal{E}}_{\gamma} y(\varepsilon)$, then either

(i) $w = a(\varepsilon)$ for some $a$ in $A'$ and $y \in \text{Var}_{\gamma}(\tau)$ where $\tau$ is the right-hand side of $\Sigma_{a,p}$ i.e. of (5.10.1), or

(ii) $w = y'(i)$ for some $y'(j)$ in $A^{(\mathcal{E})}$ and some $i$ in $[p(p)]$ and $y \in \text{Var}_{\gamma}(\tau)$ where $\tau$ is the right-hand side of (5.9.3), or

(iii) $w = b(i)$ for some $b$ in $A'$ and $w \rightarrow_{\gamma} w'$ for some $w'$ satisfying (i) or (ii) above.

The proof is an induction on the number of steps between $w$ and $y(\varepsilon)$. We obtain immediately that $\gamma$ is closed: by case (i) of the claim, $a(\varepsilon) \rightarrow_{\gamma}^* y(\varepsilon)$ implies that $y$ belongs to $\text{Var}_{\gamma}(\tau)$ which is included in $\text{PAR}(a)$. Similar remarks can show that $\rightarrow^{\mathcal{E}}_{\gamma}$ has no cycle. Hence $\Gamma$ is strongly noncircular. Finally, by mapping $a$ onto $\varphi_a$ one defines an isomorphism of $\Sigma$ onto $\Sigma(\Gamma)$ (i.e. a renaming of the function variables).

The proof of this point is left to the reader.

**5.11. Example.** Let us construct the attribute system $\Gamma'$ associated with the well-presented scheme $\Sigma'$ of Example 5.8.

We let $A^{(\mathcal{E})} = \{y, z\}$ and $A^{(\mathcal{E})} = \{\psi, \theta\}$. We now give $\Gamma'$ and indicate how each semantic rule has been obtained

$$\Gamma'': \begin{cases} 
\psi(\varepsilon) = f_1(\theta(2)) & \text{by (1'),} \\
\theta(\varepsilon) = g_1(\psi(1), \psi(2)) & \text{by (4)}, \\
y(1) = f_2(y(\varepsilon)) & \text{by using $\mu(p, 1, y)$,} \\
z(1) = g_2(y(\varepsilon), \psi(2)) & \text{by using $\mu(p, 1, z)$,} \\
y(2) = f_3(z(\varepsilon)) & \text{by using $\mu(p, 2, y)$,} \\
z(2) = f_4(\theta(1)) & \text{by using $\mu(p, 2, z)$;} 
\end{cases}$$

$$\Gamma'': \begin{cases} 
\psi(\varepsilon) = h_1 & \text{by (2'),} \\
\theta(\varepsilon) = h_2 & \text{by (3'),} \\
\psi(\varepsilon) = f_5(z(\varepsilon)) & \text{by (6).} 
\end{cases}$$

Note that $\Gamma''$ is exactly $\Gamma$ up to the renaming of $\psi$ into $a$ and of $\theta$ into $b$. 
5.12. Example. Let \( P = \{ p, q \} \) be \( \{ S \} \)-sorted with \( \rho(p) = 2 \) and \( \rho(q) = 0 \).
Let \( A^{(a)} = \{ a, a' \} \) and \( A^{(b)} = \{ y \} \) both \( \{ s \} \)-sorted. Let us define \( \Gamma' \) as follows:

\[
\begin{align*}
\Gamma_p: & \quad \begin{cases} 
  a(\varepsilon) = f(a(1), a'(2)), \\
  a'(\varepsilon) = g(a'(1), a'(2)), \\
  y(1) = h(y(\varepsilon)), \\
  y(2) = h(y(\varepsilon)), 
\end{cases} \\
\Gamma_q: & \quad \begin{cases} 
  a(\varepsilon) = i(y(\varepsilon)), \\
  a'(\varepsilon) = j 
\end{cases}
\end{align*}
\]

with help of \( F = \{ f, g, h, i, j \} \), also \( \{ s \} \)-sorted. Noting that \( \gamma_0(a, S) = y \) and \( \gamma_0(a', S) = 0 \), we introduce \( \phi_a \) and \( \phi_{a'} \) and define \( \Sigma \):

\[
\begin{align*}
(1) & \quad \phi_a(p(x_1, x_2), y) = f(\phi_a(x_1, h(y)), \phi_{a'}(x_2)), \\
(2) & \quad \phi_{a'}(p(x_1, x_2)) = g(\phi_a(x_1), \phi_{a'}(x_2)).
\end{align*}
\]

Note at this point that the last equation of \( \Gamma_p \) has not been used (and will not be). The reason is that for all \( t \) in \( M(P) \) neither \( a(\varepsilon) \) nor \( a'(\varepsilon) \) calls \( y(2) \) in \( K(t) \). Going on we get

\[
\begin{align*}
(3) & \quad \phi_a(q, y) = i(b), \\
(4) & \quad \phi_{a'}(q) = j.
\end{align*}
\]

Let us now apply the algorithm of Theorem 5.10 and reconstruct \( \Gamma \) from \( \Sigma \):

\[
A^{(a)} = \{ \phi_a, \phi_{a'} \}; A^{(b)} = \{ y \}. \]

We identify \( \phi_a \) with \( a \) and \( \phi_{a'} \) with \( a' \).

Thus we obtain for \( \Gamma_p \)

\[
\begin{align*}
  a(\varepsilon) & = f(a(1), a'(2)) \quad \text{by (1),} \\
  a'(\varepsilon) & = g(a'(1), a'(2)) \quad \text{by (2),} \\
  y(1) & = h(y(\varepsilon)) \quad \text{since } \mu(p, 1, y) = h(y) \text{ by (1),} \\
  y(2) & = \text{anything} \quad \text{since } \mu(p, 2, y) \text{ is not defined.}
\end{align*}
\]

This is not surprising: the semantic rule \( y(2) = h(y(\varepsilon)) \) has not been used to construct \( \Sigma \) as noticed above. Hence there is nothing in \( \Sigma \) to recover this semantic rule.

\[
\begin{align*}
\Gamma_\phi: & \quad \begin{cases} 
  a(\varepsilon) = i(y(\varepsilon)) \quad \text{by (3)} \\
  a'(\varepsilon) = j \quad \text{by (3)}
\end{cases}
\end{align*}
\]

This shows that requiring the inherited attribute \( y \) to be defined at all nodes of sort \( S \) whenever \( S \in \mathcal{N}_y \) is two strong. A more subtle condition should be found to make an isomorphism between well-presented p.r. schemes and strongly noncircular attribute systems. This question is left for a further work.
5.13. Chirica and Martin's construction revisited

In this appendix we want to show that conditions (1) and (2) of Definition 3.1 (closure and noncircularity of an argument selector) can be naturally found if one wants to eliminate the \( \mu \)-operator (denoting the least fixed point of a function) from the general construction of [3].

Rather than proving this point in full generality we shall consider a typical case.

Let us consider a production \( p \) of sort \( S \) and of arity \( S_1S_2 \). We assume that:

\[
A_s^{(s)} = \{a_1, \ldots, a_m\}, \quad A_s^{(i)} = \{x_1, \ldots, x_n\},
\]

\[
A_{s_1}^{(s)} = \{b_1, \ldots, b_m\}, \quad A_{s_1}^{(i)} = \{y_1, \ldots, y_n\},
\]

\[
A_{s_2}^{(s)} = \{c_1, \ldots, c_m\}, \quad A_{s_2}^{(i)} = \{y_{n+1}, \ldots, y_{2n}\}
\]

and we assume that the semantic rules associated with \( p \) are:

\[
a_i = A_i(x_1, \ldots, x_n, b_1, \ldots, b_m, c_1, \ldots, c_m), \quad 1 \leq i \leq m,
\]

\[
y_j = A_j(x_1, \ldots, x_n, b_1, \ldots, b_m, c_1, \ldots, c_m), \quad 1 \leq j \leq 2n
\]

and \( A_i, Y_j \) are terms over \( F \).

Of course, we have replaced \( a_i(\varepsilon) \) by \( a_i, y_j(1) \) by \( y_j \) if \( 1 \leq j \leq n \), \( y_j(2) \) by \( y_j \) if \( n + 1 \leq j \leq 2n \), \( x_i(\varepsilon) \) by \( x_i \) etc. without ambiguity by our choice of names for the attributes.

We have to consider a tree \( t \) of the form \( p(t_1, t_2) \) and we let

- \( \tilde{a}_i \) denote the mapping \( \lambda x_1, \ldots, x_n \cdot \varphi_a(t_1)(x_1, \ldots, x_n) \) for \( 1 \leq i \leq m \),

- \( \tilde{b}_j \) denote the mapping \( \lambda y_1, \ldots, y_n \cdot \varphi_b(t_1)(y_1, \ldots, y_n) \) for \( 1 \leq j \leq m \),

- \( \tilde{c}_l \) denote the mapping \( \lambda y_{n+1}, \ldots, y_{2n} \cdot \varphi_c(t_1)(y_{n+1}, \ldots, y_{2n}) \) for \( 1 \leq l \leq m \).

In the following equations, which are the translation of equations (16) and (17) of [3, p. 13] we use:

- new variables \( a'_1, \ldots, a'_m \) of sort \( a_1, \ldots, a_m \) respectively,

- new variables \( \tilde{y}_1, \ldots, \tilde{y}_{2n} \) of sort \( y_1, \ldots, y_{2n} \) respectively.

We let \( x \) stand for \( (x_1, \ldots, x_n) \) and similarly for \( y \) and \( \tilde{y} \). We let \( (y)_j = (y_1, \ldots, y_n) \) and \( (y)_2 = (y_{n+1}, \ldots, y_{2n}) \) and similarly for \( \tilde{y} \).

We obtain the following equations:

\[
(1) \quad (\ldots, \tilde{a}_i(x), \ldots) = \mu(\ldots, a'_i, \ldots) \cdot (\ldots, A_i(\ldots, x_i, \ldots, \tilde{b}_i((\tilde{y})_1), \ldots, \tilde{c}_l((\tilde{y})_2), \ldots), \ldots)
\]

where \( \tilde{y} \) is defined by

\[
(2) \quad \tilde{y} = \mu_y \cdot (\ldots, Y_i(\ldots, x_i, \ldots, \tilde{b}_i((y)_1), \ldots, \tilde{c}_l((y)_2), \ldots), \ldots).
\]

Let us now consider how \( \mu \) can be avoided in (1) and (2). Note that \( \mu \) is not needed in (1) since \( a'_1, \ldots, a'_m \) do not appear in the expression one takes a least fixed point of, and \( \tilde{y} \) is expressible (possibly with \( \mu \)) by (2).

Hence \( \mu \) is avoidable if and only if it is in (2). Since the variables \( y_1, \ldots, y_{2n} \) appear at the right, the only possibility to avoid \( \mu \) is when the \( \tilde{b}_i \)'s are known to depend on
certain of the variables in \( \{y_1, \ldots, y_n\} \) and the \( \tilde{c}_i \)'s on certain of the variables in \( \{y_{n+1}, \ldots, y_{2n}\} \), i.e. when some argument selector \( \gamma \) can be found, such that \( \gamma(b_i, S_1) \subseteq \{y_1, \ldots, y_n\} \) and \( \gamma(c_i, S_2) \subseteq \{y_{n+1}, \ldots, y_{2n}\} \) for all \( i = 1, \ldots, m \). This allows to eliminate \( \mu \) from (2) if and only if \( \gamma \) is noncircular (see (3.12)).

If this holds, then (2) can be formally solved and \( \tilde{y}_i \) can be defined by a finite term \( Y'_i \) written with the symbols of \( F \), the \( \tilde{b}_i \)'s, the \( \tilde{c}_i \)'s and the variables \( x_1, \ldots, x_n \).

Putting this in (1) we obtain

\[
\tilde{a}_i(x) = A_i(\ldots, x_i, \ldots, \tilde{b}_j(\ldots, Y'_j, \ldots), \ldots, \tilde{c}_i(\ldots, Y'_i, \ldots))
\]

for \( 1 \leq i \leq m \).

But \( \tilde{a}_i \) is not allowed to depend on all the variables \( x_1, \ldots, x_n \) only on those in \( \gamma(a_i, S) \) (which must also be specified since \( S \) can appear in the arity of some other production). This means that in (3), all variables of \( \{x_1, \ldots, x_n\} \) appearing in the right-hand side must be in \( \gamma(a_i, S) \). This holds for all productions if and only if \( \gamma \) is closed.

Hence we state (without any more proof)

5.14. Theorem. An attribute system is strongly noncircular if and only if the elimination of the \( \mu \)-operator from Chirica and Martin's construction described in (5.13) can be performed.

5.15. Remark. If we allow a 'bottom' ('undefined') element in the right-hand side of the equations of our recursive program schemes, the \( \mu \)-operator can be eliminated in the case of benign attribute systems ([25]).

6. The equivalence problem for attribute systems

We have shown in Section 3 that the equivalence problem for attribute systems reduces to an equivalence problem for certain tree transducers, namely the mappings \( \varphi_{a,i}^{F,S} : M(P)_S \rightarrow M(F, Y) \).

We show in this section that the equivalence problem for purely synthesized attribute systems is decidable. We give a proof in terms of primitive recursive schemes (without parameters) and hope to generalize it in future work to larger classes of p.r. schemes (with parameters), at least to the class of schemes associated with nonnested attribute systems (as defined in (3.21)). This decidability result could also be obtained as a corollary of the decidability of the equivalence of finite-state deterministic top-down tree transducers proved by Esik [11]. But we claim our method to be better suited to generalization to other classes of p.r. schemes (hence of attribute systems).

6.1. Theorem. The equivalence problem for primitive recursive schemes is solvable.
The proof needs a number of definitions.

6.2. Definitions. Let $\Sigma$ be a p.r. scheme of type $(P, F)$ with a set $\Phi$ of function variables.

Every function variable $\varphi$ defines a mapping $\varphi_F : M(P)_{\sigma(\varphi)} \rightarrow M(F)_{\sigma(\varphi)}$. Every element of $M(F \cup P \cup \Phi, \{x_1, \ldots, x_n\})$ defines a mapping

$$\text{derop}_{M(P) \cup M(F)}(t) : M(P)_{\sigma(x_1)} \times \cdots \times M(P)_{\sigma(x_n)} \rightarrow M(F)_{\sigma(t)}$$

that we shall simply denote by $t_F$ in this section.

Our problem is to decide whether $\varphi_F = \psi_F$ for given $\varphi, \psi$ in $\Phi$, and more generally whether $t_F = t'_F$ for $t, t'$ in $M(F \cup P \cup \Phi, \{x_1, \ldots, x_n\})$.

Let us define an equivalence as a pair $E = (t, t')$ of terms of $M(F \cup \Phi, \{x_1, \ldots, x_n\})$. It may be true if $t_F = t'_F$ or false if $t_F \neq t'_F$. It is valid for an assignment $\langle x_1 = u_1, \ldots, x_n = u_n \rangle$ if $t_F(u_1, \ldots, u_n) = t'_F(u_1, \ldots, u_n)$.

By the constraints due to arities, each hand side $t$ of an equivalence is of the form

$$t = s[\varphi_1(x_{i_1}), \ldots, \varphi_k(x_{i_k})],$$

for some linear $F$-term $s$,

some $\varphi_1, \ldots, \varphi_k$ in $\Phi$ (not necessarily distinct),

some sequence $i_1, \ldots, i_k$ in $[n]$.

The transduction associated with such a term $t$ is

$$t_F : M(P)_{\sigma(x_1)} \times \cdots \times M(P)_{\sigma(x_n)} \rightarrow M(F)$$

such that for all $u_1, u_2, \ldots, u_n$ in $M(P)$ of proper sort:

$$t_F(u_1, u_2, \ldots, u_n) = s[\varphi_{1F}(u_{i_1}), \ldots, \varphi_{kF}(u_{i_k})].$$

An equivalence $E = (t, t')$ is unsplittable if there exists a variable $x$ in $X$ such that $t \in \Phi(\{x\})$ and $t' \in M(F, \Phi(\{x\}))$. In other words, the typical unsplittable equivalence is of the form

$$\varphi(x), s[\varphi_1(x), \ldots, \varphi_k(x)]$$

for some $F$-term $s$ and some $\varphi, \varphi_1, \ldots, \varphi_k$ in $\Phi$. We shall use $\Lambda$ to denote a false equivalence, typically $(a, b)$ for two different constants $a$ and $b$ in $F$. We shall consider $\Lambda$ as unsplittable too.

With any equivalence $E$ we shall associate a finite set of unsplittable equivalences, denoted by $\text{SPLIT}(E)$ and defined by Algorithm 6.3 below. In order to define this algorithm, we choose $w_S$ in $M(P)_S$ for each $S$ in $\mathcal{N}$. This is possible by the assumption we have made in Remark 2.6.
6.3. Algorithm (Definition of SPLIT(E)).
1. If $E$ is unsplittable, then $\text{SPLIT}(E) = \{E\}$.
2. If $E = (\varphi(x), t)$ but $E$ is not unsplittable, this means that other variables than $x$ occur in $t$. Hence $t = s[\varphi_1(x) \cdots \varphi_k(x), \psi_1(x_1), \ldots, \psi_l(x_l)]$ for some $F$-term $s$, some $\varphi_1, \ldots, \varphi_k, \psi_1, \ldots, \psi_l$ in $\Phi$, some $x_1, \ldots, x_l$ different from $x$.
   We let $u_i = \psi_{IF}(w_{\alpha(\varphi_i)})$ for all $i$ in $[l]$ and define:
   $\text{SPLIT}(E) = \{(\varphi(x), s[\varphi_1(x), \ldots, \varphi_k(x), u_1, \ldots, u_l], (\psi_1(x_1), u_1), \ldots, (\psi_l(x_l), u_l))\}$.
3. If $E = (f(t_1, \ldots, t_n), g(t_1', \ldots, t_m'))$ for some $f$ in $F$, some $t_1, \ldots, t_n, t_1', \ldots, t_m'$ in $M(F \cup \Phi, X)$ we have the following cases:
   3.1. If $f \neq g$, then $\text{SPLIT}(E) = \{A\}$.
   3.2. If $f = g$ and $n = m = 0$, then $\text{SPLIT}(E) = \emptyset$.
   3.3. Otherwise, i.e. if $f = g$, $n = m \neq 0$, we let $\text{SPLIT}(E) = \text{SPLIT}(E_1) \cup \ldots \cup \text{SPLIT}(E_n)$ where $E_i = (t_i, t_i')$ for all $i \in [n]$.
4. Delete from $\text{SPLIT}(E)$ all equations of the form $(t, t)$.

6.4. Lemma. An equivalence $E$ is true if and only if all equivalences in $\text{SPLIT}(E)$ are true. If $\text{SPLIT}(E)$ includes $A$, then $E$ is false. If $\text{SPLIT}(E) = \emptyset$, then $E$ is true. If all equivalences of $\text{SPLIT}(E)$ are valid for some assignment, then $E$ is valid for the same assignment.

Proof. We shall only consider case 2 of the definition of $\text{SPLIT}(E)$. If $E = (\varphi(x), t)$ is true, this means that the functions $\psi_{IF_1}, \ldots, \psi_{IF}$ are constant, hence that $\psi_{IF}(u) = \psi_{IF}(w_{\alpha(\varphi_i)}) = u_i$ for all $u$ in $M(P)_{\alpha(\varphi)}$, hence that the equivalence $(\psi_1(x), u_1)$ is true. Then for all $u$ in $M(P)_{\alpha(\varphi)}$, $\varphi_F(u) = s[\varphi_1F(u), \ldots, \varphi_kF(u), \psi_1F(w_{\alpha(\varphi_1)}), \ldots, \psi_lF(w_{\alpha(\varphi_l)})]$, i.e. the equivalence $(\varphi(x), s[\varphi_1(x), \ldots, \varphi_k(x), u_1, \ldots, u_l])$ is true. Hence all equivalences in $\text{SPLIT}(E)$ are true.

The other parts are simpler to prove.

We shall now develop every unsplittable equivalence into a finite set of equivalences $\text{DEV}(E)$ the definition of which will depend on $\Sigma$.

6.5. Algorithm (Definition of $\text{DEV}(E)$).
1. If $E = A$, then $\text{DEV}(E) = \{A\}$.
2. Otherwise, $E$ is of the form $(\varphi(x), t)$; define $\text{DEV}(E, p)$ for all $p$ in $P$ of sort $\alpha(\varphi)$ as follows:
   2.1. substitute $p(x_1, \ldots, x_m)$ for $x$ in both hand sides of $E$ where $x_1, \ldots, x_m$ are variables different from $x$ and of appropriate sort; this substitution defines a pair $(s, s')$ of terms in $M(F \cup P \cup \Phi, X)$;
   2.2. define now $\text{DEV}(E, p)$ as the pair $(\text{nf}_2(s), \text{nf}_2(s'))$ (note that $\text{nf}_2(s)$ and $\text{nf}_2(s')$ belong to $M(F \cup \Phi, X)$).
Then define $\text{DEV}(E) = \{\text{DEV}(E, p) | p \in P, \alpha(p) = \alpha(\Phi)\}$. 
6.6. Lemma. An unsplittable equivalence \( E \) is true if and only if all equivalences in \( \text{DEV}(E) \) are true.

Let us define \( \text{SDEV}(E) \) as the union of all the \( \text{DEV}(E') \) for all \( E' \) in \( \text{SPLIT}(E) \).

Hence \( \text{SDEV} \) is a computable function: \( \mathcal{E} \rightarrow P_0(\mathcal{E}) \) where \( \mathcal{E} \) is the set of all unsplittable equivalences (and \( P_0(\mathcal{E}) \) is the set of finite subsets of \( \mathcal{E} \)). We shall use variable renamings. Hence our unsplittable equivalences will be written with one variable \( x \) for each sort in \( \mathcal{N} \). A subset \( \mathcal{C} \) of \( \mathcal{E} \) is closed if \( \text{SDEV}(E) \subseteq \mathcal{C} \) for all \( E \) in \( \mathcal{C} \).

Hence by Lemmas 6.4 and 6.6 the set \( \mathcal{F} \) of all the true unsplittable equivalences is closed. Conversely,

6.7. Proposition. If a subset \( \mathcal{C} \) of \( \mathcal{E} \) is closed and does not contain \( \Lambda \), then \( \mathcal{C} \subseteq \mathcal{F} \).

Proof. Let \( \mathcal{C}_0 \) be the set of all equivalences of \( \mathcal{C} \) of the form \((\varphi(x), u)\) for \( u \) in \( M(F) \) and \( \varphi \) in \( \Phi \). (Such an equation expresses that \( \varphi_F \) is the constant function with value \( u \).)

Since \( \Lambda \in \mathcal{C} \), the set \( \mathcal{C}_1 = \mathcal{C} - \mathcal{C}_0 \) is formed of equivalences of the form \((\varphi(x), u)\) with \( \text{Var}(u) = \{x\} \) (hence \( u \) has occurrences of function symbols of \( \Phi \)).

Note that for all \( E \) in \( \mathcal{C}_0 \) \( \text{SDEV}(E) \subseteq \mathcal{C}_0 \), hence \( \mathcal{C}_0 \) is closed.

Let \( Q_0 \) be \( \bigwedge_{E \in \mathcal{C}_0} Q_E \), the conjunction of all properties \( Q_E \) for all \( E \) in \( \mathcal{C}_0 \) where, if \( E = (\varphi(x), u) \), the property \( Q_E \) is

\[
\forall t \in M(P)(\sigma_{\{x\}} [\varphi_F(t) = u]).
\]

If \( \mathcal{C}_0 \) is closed (and does not contain \( \Lambda \)), then \( Q_0 \) is provable by structural induction; we omit the proof of this point (see the proof for \( Q_1 \) below). Hence \( Q_0 \) is true.

Let now \( Q_1 = \bigwedge_{E \in \mathcal{C}_1} Q_E \) where for \( E = (\varphi(x), u) \) we denote by \( Q_F \) the following property:

1) \( \forall t \in M(P)(\sigma_{\{x\}} [\varphi_F(t) = u_F(t)]) \).

By using the fact that \( Q_0 \) is true, we shall show that \( Q_1 \) is provable by structural induction. For each \( E = (\varphi(x), u) \) in \( \mathcal{C}_1 \), let us prove:

2) \( \forall p \in P_{(x, \sigma_{\{x\}})} [\varphi_F(p) = u_F(p)] \).

We have to show that for each \( p \) in \( P_{(x, \sigma_{\{x\}})} \) the equivalence \( \text{DEV}(E, p) \) is true. Note that \( \text{DEV}(E, p) = (t, t') \) for some \( t \) and \( t' \) in \( M(F) \). Hence either \( \text{SPLIT}(\{t, t'\}) = \emptyset \) and \( t = t' \) i.e. \( (t, t') \) is true, or \( \text{SPLIT}(\{t, t'\}) = \{\Lambda\} \) and \( t \neq t' \) i.e. \( (t, t') \) is false.

Since \( \mathcal{C} \) is closed and does not contain \( \Lambda \) the former holds. Hence we have proved (2) and the validity of the inductive assertion for the constants.

Let us now prove (1) for all \( t \) of the form \( p(t_1, \ldots, t_n) \) by assuming the validity of

\[
\psi_F(t_I) = w_F(t_I)
\]

for all \( t \) in \( [n] \), all \((\psi(x), w)\) in \( \mathcal{C}_1 \) such that \( \alpha(\psi) = \sigma(t_I) \). We have to prove the validity of

3) \( \varphi_F(p(x_1, \ldots, x_n)) = u_F(p(x_1, \ldots, x_n)) \)
for the assignment \( \tau: (x_1 = t_1, \ldots, x_n = t_n) \). It will be a consequence of the validity of each \( E' \) in \( \text{DEV}(E, p) \) for the same assignment. Let \( E' \) be such an equivalence; its validity will be a consequence of the validity of each equivalence \( E'' \) in \( \text{SPLIT}(E') \) (by Lemma 6.4). Note that

(a) if \( \text{SPLIT}(E') = \emptyset \), then \( E' \) is true;
(b) \( \text{SPLIT}(E') \) cannot contain \( \Lambda \) (\( \text{SPLIT}(E') \subseteq \mathcal{C} \) since \( \mathcal{C} \) is closed and \( \mathcal{C} \subseteq \mathcal{C} - \{\Lambda\} \));
(c) let \( E'' \) be in \( \text{SPLIT}(E') \). It may be in \( \mathcal{C}_0 \) hence is true since \( Q_0 \) is true. Otherwise \( E'' \) is of the form \((\psi(x_i), w)\) for some \( i \) in \( \lbrack n \rbrack \), some \( \psi \in \Phi \), some \( w \) in \( M(F, \Phi(\{x_i\})) \). By our inductive assumption \( E'' \) is valid for the assignment \( \tau \).

This shows the validity of \( E' \) for the assignment \( \tau \) and the same for (3). Finally, we have shown that the inductive step of the principle of structural induction holds (namely property (3) of Definition 2.4).

Hence \( Q_1 \) is true. We have shown that \( \mathcal{C} \subseteq \mathcal{T} \).

6.8. Lemma. There are finitely many true unsplittable equivalences (up to variable renaming).

Proof. We shall use the elements \( w_s \) of \( M(P)_s \) defined in (6.2) for \( S \) in \( \mathcal{N} \).

Let \((\varphi(x), t)\) be a true unsplittable equivalence with \( i = s[\varphi_1(x), \ldots, \varphi_k(x)] \) for some \( F \)-term \( s \) and some \( \varphi_1, \ldots, \varphi_k \) in \( \Phi \).

We have

\[
\varphi_F(w_s) = s[\varphi_1F(w_s), \ldots, \varphi_kF(w_s)]
\]

where \( S = \sigma(x) = \alpha(\varphi) = \alpha(\varphi_1) = \cdots = \alpha(\varphi_k) \).

Hence \(|s| \leq |\varphi_F(w_s)| \leq \max\{|\varphi_F(w_{\alpha(\psi)})|\mid \psi \in \Phi\} \), which shows that there are finitely many equivalences as above since \( \Phi \) is finite.

Proof of Theorem 6.1. In order to decide the truth of an equivalence \( E_0 \) we can use the following algorithm.

6.9. Algorithm.

1. Input: \( E_0 \) in \( M(F \cup \Phi, \{x_1, \ldots, x_n\}) \).
2. Compute a sequence \( \mathcal{C}_i \) of finite subsets of \( \mathcal{C} \) as follows:
   2.1. \( \mathcal{C}_0 = \text{SPLIT}(E_0) \),
   2.2. \( \mathcal{C}_{i+1} = \mathcal{C}_i \cup \text{SDEV}(\mathcal{C}_i) \).
3. As soon as some \( \mathcal{C}_i \) containing \( \Lambda \) is found, stop with answer: “\( E_0 \) is false”.
4. As soon as \( \text{SDEV}(\mathcal{C}_i) \subseteq \mathcal{C}_i \) with \( \Lambda \not\in \mathcal{C}_i \), stop with answer: “\( E_0 \) is true”.

Let us prove the correctness of this algorithm.

Case 1: There exists \( i \) such that \( \Lambda \in \mathcal{C}_i \). Then the algorithm stops with answer: “\( E_0 \) is false” and the answer is correct by Lemmas 6.4 and 6.6.

Case 2: For no \( i \) we have \( \Lambda \in \mathcal{C}_i \). Note that the sets \( \mathcal{C}_i \) are defined for all \( i \) by clauses 1 and 2 of the algorithm, independently of the termination of the algorithm.
Let us consider the (a priori infinite) set \( C = \bigcup_{i=0}^{\infty} C_i \). It is closed and does not contain \( \Lambda \).

Hence \( C \subseteq T \) by Proposition 6.7, hence \( C \) is finite since \( T \) is by Lemma 6.8. Since \( C_i \) is an increasing sequence, there exists \( i \) such that \( C = C_i \) whence \( \text{SDEV}(C_i) \subseteq C_i \). Hence the algorithm stops with answer "\( E_0 \) is true". This answer is correct by Lemma 6.4 since \( \text{SPLIT}(E_0) \subseteq C \subseteq T \). Our algorithm stops in all cases with a correct answer, hence is correct.

**6.10. Remark.** As in the algorithm of Korenjak and Hopcroft (see Harrison et al. [14] and Courcelle [4]) the set \( C \) constructed in Algorithm 6.9 can be made into a finite tree, defined as follows:

1. Its root is labelled by \( E_0 \).
2. The successors of \( E_0 \) are labeled by the elements of \( \text{SPLIT}(E_0) \).
3. Each node is labelled by an unsplittable equivalence say \( E \). Then two cases may happen:
   3.1. this node is *developed*, i.e. it has one successor for each \( E' \) in \( \text{SDEV}(E) \),
   3.2. this node is not developed, hence is a leaf, but some other node labelled by \( E \) is developed (or \( \text{SDEV}(E) = \emptyset \)).
4. No node is labelled by \( \Lambda \).

Hence \( E_0 \) is true if and only if such a tree (necessarily finite by the proof of correctness of our algorithm) can be constructed.

**6.11. Example.** Let \( P = \{ p, q \} \) be one-sorted, \( \rho(p) = 2 \) and \( \rho(q) = 0 \). Let \( F = \{ f, g, h, i \} \) be also one-sorted and \( \rho(f) = 2, \rho(g) = 1, \rho(h) = \rho(i) = 0 \).

Let now \( \Phi = \{ \varphi, \psi, \varphi', \psi' \} \) and \( \Sigma \) be the following p.r. scheme of type \( (P, F) \):

\[
\begin{align*}
\varphi p(x_1, x_2) &= f(g\psi x_1, \varphi x_2), & \varphi q &= k, \\
\psi p(x_1, x_2) &= g\psi x_2, & \psi q &= h, \\
\varphi' p(x_1, x_2) &= f(\psi' x_1, \varphi' x_2), & \varphi' q &= k, \\
\psi' p(x_1, x_2) &= g\psi' x_2, & \psi' q &= gk.
\end{align*}
\]

Let \( E_0 \) be \( (\varphi x_1, \varphi' x_1) \). Let us start the construction of the tree described in (6.10).

For more clarity, equivalences will be written \( t = t' \) instead of \( (t, t') \), (which does not mean that they are true).

Note that \( \text{SPLIT}(E_0) = \{ E_0 \} \).

We start the construction of the successors of \( E_0 \)

\[
\text{DEV}(E_0) = \{ f(g\psi x_1, \varphi x_2) = f(\psi' x_1, \varphi' x_2), k = k \}
\]

whence

\[
\text{SDEV}(E_0) = \{ g\psi x_1 = \psi' x_1, \varphi x_2 = \varphi' x_2 \}.
\]
Since the second equivalence of $SDEV(E_0)$ is exactly $E_0$ up to a variable renaming, the corresponding node will not be developed.

We only have to develop $E_1$, the equivalence $g\psi x_1 = \psi' x_1$:

$$DEV(E_1) = \{gg\psi x_2 \equiv g\psi' x_2, gh \equiv gk\}, \quad SDEV(E_1) = \{g\psi x_2 \equiv \psi' x_2, \Lambda\}.$$

The $\Lambda$ comes from $gh = gk$ which is obviously false. Hence we get the answer: “$E_0$ is false”. Let us now modify $\Sigma$ by replacing the last equation by $\psi' q = gh$.

Then

$$SDEV(E_0) = SDEV(E_1),$$
$$DEV(E_1) = \{gg\psi x_2 \equiv g\psi' x_2, gh \equiv gh\},$$
$$SDEV(E_1) = \{g\psi x_2 = \psi' x_2\}.$$

But the equivalence in $SDEV(E_1)$ is $E_1$ up to a variable renaming, hence the corresponding node will not be developed. Our tree is completed (see Fig. 6.1(b) and gives the answer “$E_0$ is true”.

![Fig. 6.1.](image)

**6.12. Corollary.** *The equivalence problem for purely synthesized attribute systems is decidable.*

**Proof.** This result follows from Proposition 5.5 and Theorem 6.1.

**6.13. Definition.** An attribute system is *quasi-purely synthesized* if the semantic rules defining inherited attributes are of the form

$$y(k) = z(e)$$

instead of the general form (3.1.2).
Hence such an attribute system is nonnested and strongly noncircular (by Proposition 3.2).

From construction (5.1) it follows that all equations of the recursive scheme \( \Sigma(\Gamma) \) associated with a quasi purely synthesized attribute system \( \Gamma \) are of the form:

\[
\varphi(p(x_1, \ldots, x_n), y^{(1)}, \ldots, y^{(m)}) = \psi[\tau_1, \ldots, \tau_k]
\]

where each \( \tau_i \) belongs to \( \{y^{(1)}, \ldots, y^{(m)}\} \cup \mathcal{D} \).


Proof. Algorithm 6.9 applies exactly as before with a set \( \mathcal{E}' \) of unsplittable equivalences defined as follows:

\[
\mathcal{E}' = \{(t, t') | t \in \mathcal{D}(\{x\} \cup Y) \text{ and } t' \text{ is either in } \mathcal{D}(\{x\} \cup Y) \text{ or is } M(F, \mathcal{D}(\{x\} \cup Y) \cup Y)\}.
\]

We only have to add to the definition of \( \text{SPLIT}(E) \) the following cases:
- if \( E = (y, t') \) or \( (t', y) \) for some \( y \) in \( Y \) and some \( t' \) such that: First \( (t') \in F \cup Y - \{y\} \), then \( \text{SPLIT}(E) = \{A\} \);
- if \( E = (y, y) \), then \( \text{SPLIT}(E) = \emptyset \).

In the proof of the result corresponding to (6.7) we shall use \( \mathcal{C}_0 = \{(\varphi(x), u) \in \mathcal{C} | u \in M(F, Y)\} \). The other details are left to the reader.

6.15. Remark. It could be proved that the mappings \( \varphi_F: \mathcal{T}(P)_{\alpha(\varphi)} \rightarrow M(F, Y)_{\alpha(\varphi)} \) associated with a quasi-purely synthesized attribute system and defined in (4.16) are DT-transductions. Hence Proposition 6.14 can be stated as a corollary of (6.16) below. We have preferred to show that our algorithm extends in a straightforward manner to recursive schemes associated with quasi-purely synthesized attribute systems.

We hope to provide other extensions in the future.


Proof. This result follows from Proposition 4.7 and Theorem 6.1.

Note added at revision. Algorithm 6.9 is in the family of algorithms described in Courcelle [4]. An extension of Proposition 6.14 can be found in Courcelle and Franchi-Zanettacci [5].
References

[12] I. Fang, FOLDS, a declarative formal language definition system, Computer Science Department, Stanford University, Palo Alto, California, STAN-CS-329 (1972).