A multi-visit characterization of absolutely noncircular attribute grammars

By E. GOMBÁS and M. BARTHA

1. Introduction

Simple multi-visit attribute grammars were introduced in [1]. An attribute grammar is simple multi-visit if each nonterminal has a fixed visit-number associated with it such that, during attribute evaluation, the attributes of a node which have visit-number j are computed in the j-th visit to the node. Putting in one class the attributes having the same visit-number we get an ordered partition of the attributes of the nonterminal. Thus, a visit to a node is a sequence of actions consisting of (a) computing all the inherited attributes contained in the class corresponding to the visit, (b) making some visits to the sons of the node and (c) computing all the synthesized attributes of the class. This evaluation strategy can be implemented in a natural way translating visits to recursive procedures, where inherited and synthesized attributes correspond to input and output parameters, respectively. It was proved in [1] that the problem whether an attribute grammar is simple multi-visit is NP (time) — complete.

In [2] Kastens introduced a subclass of simple multi-visit attributed grammars and called them ordered. The problem whether an attribute grammar is ordered can already be decided in polynomial time, but the choise of this subclass seems rather heuristic. A somewhat larger subclass, which is still decidable in polynomial time was investigated in [3]. However, because of the NP-completeness mentioned above there is no reason to work out further improvements.

In this paper we investigate a class of attribute grammars that can be evaluated by a multi-visit type strategy associating a fixed set of visits with each nonterminal, but the order of these visits need not be predetermined. We call these grammars generalized simple multi-visit. It turns out that this class of attribute grammars coincides with the class of absolutely noncircular attribute grammars, for which a more complicated tree-walking evaluator was given in [4].

It is known that the problem whether an attribute grammar is absolutely noncircular is decidable in polynomial time. However we shall show that the problem whether an attribute grammar is generalized simple m-visit for a fixed m (even for $m=2$) is still NP-complete. This means that, instead of trying to minimize the number of visits, it is more useful to execute them in parallel if possible. In section 5 we describe two such parallel evaluation strategies.

It was shown in [1] that an attribute grammar is simple multi-visit iff it is *l*-ordered, i.e. for each nonterminal a linear order of its attributes exists such that the attributes of a node can always be evaluated in that order. Let us call an attribute grammar top-down controlled *l*-ordered if there exists a finite state deterministic top-down tree automaton such that the state in which the automaton reaches a node of a derivation tree determines the evaluation order of the attributes of the node. As it is to be expected, the class of absolutely noncircular attribute grammars coincides with the class of top-down controlled *l*-ordered attribute grammars, too.

2. Preliminaries

Since we are dealing with almost the same notions as those defined in [1] (e.g. computation sequences, visit sequences, etc.), we adopt the main terminology introduced in that paper.

An attribute grammar G consists of:

(i) A reduced context free grammar $G_0=(T, N, P, Z)$, where T and N denote the set of terminal and nonterminal symbols, respectively, P is the set of productions (syntactic rules) and $Z\in N$ is the start symbol. We shall denote nonterminal symbols always by capital letters, while terminal strings by small ones.

(ii) A set A of attributes, which is the union of two disjoint finite sets A_s and A_i. The elements of A_s and A_i are called synthesized and inherited attributes, respectively (shortly *s*- and *i*-attributes). A function v assigns each nonterminal F a set $I(F)$ of inherited and a set $S(F)$ of synthesized attributes, i.e. $v(F)=I(F)\cup S(F)$. An attribute $a\in v(F)$ will be referenced as $a(F)$. The start symbol Z has only one *s*-attribute, and it does not occur on the right-hand side of any production.

(iii) A set $V(a)$ of possible values for each attribute a.

(iv) A set r_p of semantic rules associated with each production p. If p is of the form $p\colon F_0\to w_0F_1\dots F_{n_p}w_{n_p}$, then a semantic rule of r_p is an equation: $a_0(F_{i_0})=f(a_1(F_{i_1}), \dots, a_m(F_{i_m}))$, where $0\leqq i_j\leqq n_p$ and $f\colon V(a_1)\times\dots\times V(a_n)\to V(a_0)$ is a (recursive) function. This equation is interpreted by saying that attribute $a_0(F_{i_0})$ depends on attributes $a_1(F_{i_1}), \dots, a_m(F_{i_m})$ in p. We assume that G is in Bochmann normal from, i.e. the rules of r_p assign all and only the attributes in $S(F_0)$ and $I(F_j)$ $(j\geqq 1)$ using as argument only attributes in $I(F_0)$ and $S(F_j)$.

The production graph of $p\colon F_0\to w_0F_1\dots F_{n_p}w_{n_p}$ (denoted by $pg(p)$) has as nodes the disjoint union of $v(F_i)$, $0\leqq i\leqq n_p$, and there is an edge from $a_1(F_{i_1})$ to $a_2(F_{i_2})$ iff $a_2(F_{i_2})$ depends on $a_1(F_{i_1})$ in p. For a derivation tree t of G_0 we get the derivation tree graph of t (denoted by $dtg(t)$) by pasting together the pg's of all the productions t consist of. G is noncircular if none of the derivation trees of G_0 has an oriented cycle in its dtg. In the sequel, if no confusion arises we identify a nonterminal node of a derivation tree with its label.

3. The generalized simple multi-visit property

To describe an attribute evaluation strategy we define computation sequences similar to those introduced in [1]. A computation sequence is a sequence of so called basic actions, where each basic action is either the evaluation of some *i*-attributes of a node, called entering the node, or the evaluation of some *s*-attributes of a node,

called exiting the node. Thus, a basic action can be represented by a basic action symbol (ba-symbol) $i(n, A)$ or $s(n, A)$, where n denotes a nonterminal node in a derivation tree and A is a subset of $I(n)$ or $S(n)$, respectively. Our computation sequences, however, allow redundant computations, too.

Let $G=(G_0=(T, N, P, Z), A, v, \{V(a)|a\in A\}, \{r_p|p\in P\})$ be an attribute grammar, fixed in the rest of this section, and t a derivation tree of G_0. We always assume that a derivation tree is complete, i.e. its root is Z and its leaves are in T.

Definition 3.1. A computation sequence for t is a string h of ba-symbols, which satisfies the following four conditions.

(1) Start-end condition: the first and the last ba-symbols are $i(Z, \emptyset)$ and $s(Z, \emptyset)$.

(2) Sequentiality condition: for any two contiguous ba-symbols $x_1(n_1, A_1)$ $x_2(n_2, A_2)$ in h one of the following conditions holds.

(i) n_2 is a son of n_1 and $x_1=x_2=i$;

(ii) n_2 is the father of n_1 and $x_1=x_2=s$;

(iii) n_2 is a brother of n_1 and $x_1=s$ and $x_2=i$;

(iv) $n_2=n_1$ and $x_1\neq x_2$.

(3) Feasibility condition: For any production p consider an arbitrary occurrence of p in t, and let n_1 and n_2 be any such nonterminal nodes of this occurrence that $a_1(n_1)$ depends on $a_2(n_2)$ in p for some attributes a_1 and a_2. Then the first ba-symbol in h which contains $a_2(n_2)$ cannot precede the first such ba-symbol that contains $a_1(n_1)$.

(4) Completeness condition: For each node n of t, if $i(n, B_1), s(n, A_1), \ldots, \ldots, i(n, B_k), s(n, A_k)$ is the sequence of the ba-symbols of n occurring in h, then $\bigcup(\{A_i\cup B_i\}|i\in[k-1])$ is a partition of $v(n)$; furthermore $A_k\cup B_k=\emptyset$. The set $\Pi(n)=\bigcup(\{A_i\cup B_i\}|i\in[k])$ will be called the visit set of n and each $A_i\cup B_i$ $(i\in[k])$ a visit element. Since $\bigcup(\{A_i\cup B_i\}|i\in[k-1])$ is a partition, each visit element, except one is a nonvoid subset of $v(n)$. $v\in\Pi(n)$ will be referenced as $v(n)$ or $v(F)$, if F is the nonterminal label of n.

The (simple multi-visit) completeness condition in [1] required $\{A_i\cup B_i|i\in[k]\}$ to be an ordered partition of $v(n)$. The main point of our modification is that we allow making the same visit to a node several times.

Let n_0 be a node of t having sons $n_1, \ldots, n_m$, $v=A\cup B\in\Pi(n_0)$ and h a computation sequence for t. A visit trace of $v(n_0)$ in h, denoted by $tr(v(n_0))$ is a substring of h beginning with $i(n_0, B)$ and ending with $s(n_0, A)$ such that there is no further ba-symbol of n_0 between these two ones. By the definition of a computation sequence $tr(v(n_0))=tr(v_1(n_{i_1}))\ldots tr(v_l(n_{i_l}))$, where $v_1, \ldots, v_l$ are certain visit elements of $n_{i_1}, \ldots, n_{i_l}$, respectively. The sequence $v_1(n_{i_1})\ldots v_l(n_{i_l})$ will be called a visit sequence of $v(n_0)$. (Note that l might be 0, and a visit element might have several visit traces and visit sequences so far.)

Let Π: $N\to\mathcal{P}(\mathcal{P}(A))$ be a function such that for every $F\in N$ $\Pi(F)=\pi\cup\{\emptyset\}$, where π is a partition of $v(F)$. Furthermore, let ϱ be a set of functions $\{\varrho_p|p\in P\}$ such that if p: $F_0\to w_0F_1\ldots F_{n_p}w_{n_p}$, then ϱ_p assigns each $v\in\Pi(F_0)$ a sequence $v_1(F_{i_1})\ldots v_l(F_{i_l})$ $(l\geqq 0)$, where $v_m\in\Pi(F_{i_m})$ $(m\in[l], i_m\in[n_p])$ and all the v_m-s are different. Π will be called a collection of visit sets for G and ϱ a visit description function for Π.

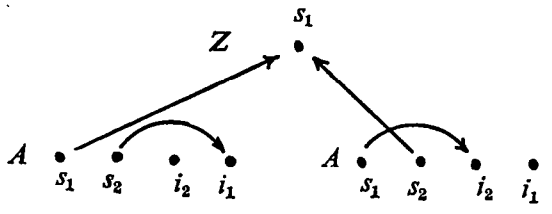
Definition 3.2. Given a collection Π of visit sets for G, a visit description function ϱ for Π and a derivation tree t, a computation sequence h for t respects ϱ if the following condition holds. For each nonterminal node n of t, if n is the left-hand side occurrence of a production $p: F_0 \to w_0 F_1 \dots F_{n_p} w_{n_p}$, then $\Pi(n) = \Pi(F_0)$ and each $v \in \Pi(n)$ has a unique visit sequence in h which is equal to $\varrho_p(v)$.

Note that there exists at most one computation sequence for t respecting ϱ. ϱ is called a generalized simple multi-visit description function for Π if each derivation tree of G has a computation sequence for it respecting ϱ.

Definition 3.3. G is a generalized simple multi-visit (gsmv) attribute grammar if there exist a collection Π of visit sets and a gsmv visit description function ϱ for Π. G is generalized simple m-visit ($m \in \mathbf{N}$) if $\|\Pi(F)\| \leqq m+1$ for each $F \in N$.

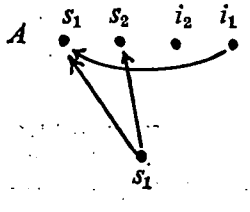
From this definition it is clear that the only essential difference between smv and gsmv evaluation strategies is that in the latter one each visit to a node must be made only if this has not been done before.

Example 3.4. Let Z, A, B and C be the nonterminals of the underlying grammar with the following attributes. $S(Z) = S(B) = \{s_1\}$, $I(Z) = I(B) = \emptyset$, $S(A) = \{s_1, s_2\}$, $I(A) = \{i_1, i_2\}$. The productions and the corresponding production graphs are listed below.



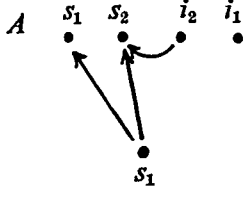
$1: Z \to AA$

a)



$2: A \to aB$

b)



$3: A \to bB$

c)

$4: B \to \lambda$

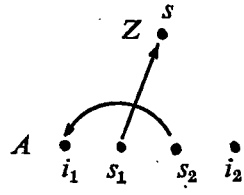
d)

If $\Pi(Z)=\Pi(B)=\{\{s_1\},\emptyset\}$ and $\Pi(A)=\{\{i_1,s_1\},\{i_2,s_2\},\emptyset\}$, then let $(\varrho_1(\{s_1\}=\{i_2,s_2\}(A_1)\{i_1,s_1\}(A_1)\{i_1,s_1\}(A_2)\{i_2,s_2\}(A_2)$, $\varrho_1(\emptyset)=\emptyset(A_1)\emptyset(A_2)$, $\varrho_2(\{i_1,s_1\})=\varrho_2(\{i_2,s_2\})=\{s_1\}(B)$, $\varrho_2(\emptyset)=\emptyset(B)$, $\varrho_3(\{i_1,s_1\})=\varrho_3(\{i_2,s_2\})=\{s_1\}(B)$, $\varrho_3(\emptyset)=\emptyset(B)$, $\varrho_4(\{s_1\})=\varrho_4(\emptyset)=\lambda$.

It is clear that ϱ is a gsmv description function for Π. However, G is not smv, because the visit elements of A cannot be evaluated in a fixed order. That is why we had to put the single visit element of B into both $\varrho_{2(3)}(\{i_1,s_1\})$ and $\varrho_{2(3)}(\{i_2,s_2\})$.

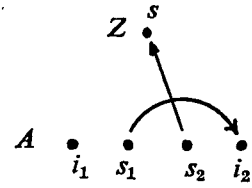
The following example shows that the void visit element is necessary.

Example 3.5. Let Z, A and B be the nonterminals of the underlying grammar with the following attributes $S(Z)=\{s\}$, $S(A)=S(B)=\{s_1,s_2\}$, $I(Z)=\emptyset$, $I(A)=\{i_1,i_2\}$, $I(B)=\{i,i_1,i_2\}$. The productions and the corresponding production graphs are listed below.



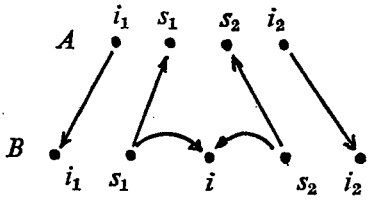
$1: Z \to aA$

a)



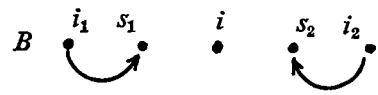
$2: Z \to bA$

b)



$3: A \to B$

c)



$4: B \to \lambda$

d)

G is clearly gsmv, but in the production $A \to B$ the useless attribute $i(B)$ can be evaluated only in the void visit of A.

To define the absolutely noncircular (anc) property we use the concept of induced production graph (ipg) introduced in [5]. These graphs can be obtained by adding some further edges to the production graphs. More exactly, considering a graph as a relation we get the ipg graphs by taking the least fixpoint of the following system of equations.

$ipg(p)=pg(p)\cup\{\langle a(F_i),b(F_i)\rangle | p\colon F_0\to w_0F_1\dots F_{n_p}w_{n_p},\ i\in[n_p]$ and there is a $q\in P$ with left-hand side F_i such that $\langle a(F_i),b(F_i)\rangle\in ipg(q)^+\}$.

G is anc if for each $p\in P$ $ipg(p)$ is acyclic.

Let Π be a collection of visit sets for G. Π defines for every $F\in N$ a set $B(F)$ of nonvoid basic action symbols as follows. If $\Pi(F)=\{v_1, \ldots, v_n\}$, then $B(F)=$ $=\{s(F, A_i)|i\in[n],\ v_i=A_i\cup B_i, B_i\subseteq I(F), A_i\neq\emptyset\}\cup\{i(F, B_j)|j\in[n], v_j=A_j\cup B_j,\ A_j\subseteq$ $\subseteq S(F),\ B_j\neq\emptyset\}$.

The production graph over the nonvoid ba-symbols of $p:F_0\to w_0F_1\ldots F_{n_p}w_{n_p}$ (denoted by $pgb_\pi(p)$) is the following graph. The set of its nodes is the disjoint union of $B(F_i)$, $0\leq i\leq n_p$, and there is an edge from $x_1(F_{i_1}, A_1)$ to $x_2(F_{i_2}, A_2)$ iff $a_2(F_{i_2})$ depends on $a_1(F_{i_1})$ in p for some $a_1\in A_1$, $a_2\in A_2$. From these graphs we construct the induced production graphs $\{ipgb_\pi(p)|p\in P\}$ as above, i.e. by taking the least fixpoint of the following system of equations.

$ipgb_\pi(p)=pgb_\pi(p)\cup\{\langle i(F_i, B), s(F_i, A)\rangle|0\leq i\leq n_p,\ A\neq\emptyset,\ B\neq\emptyset$ and $A\cup B\in$ $\in\Pi(F_i)\}\cup\{\langle i(F_i, B_1, s(F_i, A_2)\rangle, \langle s(F_i, A_1), i(F_i, B_2)\rangle|$ none of A_i and B_i $(i=1, 2)$ is $\emptyset$, $\{A_i\cup B_i|i=1, 2\}\subseteq\Pi(F_i)$ and there is a $g\in P$ with left-hand side F_i such that $\langle i(F_i, B_1), s(F_i, A_2)\rangle\in ipgb_\pi(q)^+\}$.

Remark 3.6. If for each $F\in N$ $\Pi(F)=\big\{\{a\}|a\in v(F)\big\}\cup\{\emptyset\}$, then $ipgb_\pi(p)\cong ipg(p)$ for every $p\in P$.

The following alogrithm can be used to compute the $ipgb_\pi$ relations.

Algorithm 3.7.

Step 1. For each $p\in P$ set $ipgb_0(p)=pgb_\pi(p)\cup\{\langle i(F, B), s(F, A)\rangle|F$ occurs in $p, A\neq\emptyset, B\neq\emptyset$ and $A\cup B\in\Pi(F)\}$

Step 2. If $j\geq 0$, then for each $p\in P$ let $ipgb_{j+1}(p)=ipgb_j(p)\cup\{\langle i(F, B_1),$ $s(F, A_2)\rangle, \langle s(F, A_1), i(F, B_2)\rangle|$ none of A_i, B_i $(i=1, 2)$ is $\emptyset$, $\{A_i\cup B_i|i=1, 2\}\subseteq$ $\subseteq\Pi(F)$, F occurs on the right-hand side of p and on the left-hand side of such a $q\in P$ for which $\langle i(F, B_1),\ s(F, A_2)\rangle\in ipgb_j(q)^+\}$

Repeat step 2 until such a j is found for which $ipgb_j(p)=ipgb_{j+1}(p)$ for all $p\in P$. Then $ipgb_\pi(p)=ipgb_j(p)$.

It is easy to see that the time complexity of this algorithm is polynomial in the parameters of G.

Lemma 3.8. Given a collection Π of visit sets for G, there exists a gsmv description function ϱ for Π iff $ipgb_\pi(p)$ is acyclic for every $p\in P$.

Proof. (a) (ϱ exists$\Rightarrow ipgb_\pi$ is acyclic)

It is enough to prove that if $\langle x_1(F, A_1), x_2(F, A_2)\rangle\in ipgb_\pi(p)\setminus pgb_\pi(p)$, then for every derivation tree t and F-labelled node n_0 of t the following statement holds. If h is the computation sequence for t respecting ϱ, then the first occurrence of $x_1(n_0, A_1)$ precedes that of $x_2(n_0, A_1)$ in h. We shall prove this statement by an induction following algorithm 3.7. First we make the following observation.

If $A_1\subseteq v_0^1(n_0)$ and $A_2\subseteq v_0^2(n_0)$ for some $v_0^1, v_0^2\in\Pi(n_0)$, then consider the sequence $n_0, \ldots, n_m$ $(m\geq 0)$ of nodes and the sequences $v_0^1(n_0), \ldots, v_m^1(n_m)$ and $v_0^2(n_0), \ldots, v_m^2(n_m)$ of visit elements with the following properties.

1) For each $0\leq i<m$, n_{i+1} is the father of n_i and $v_{i+1}^j(n_{i+1})$ $(j=1, 2)$ are those visit elements for which the ba-symbols of $v_i^j(n_i)$ can be found in the visit trace corresponding to $v_{i+1}^j(n_{i+1})$ as first occurrences in h.

2) a) $v_m^1(n_m) = v_m^2(n_m) = v_m$ but $v_i^1(n_i) \neq v_i^2(n_i)$ if $i < m$, or

b) $n_m = Z$ and $v_m^1(Z) \neq v_m^2(Z)$.

It is important to note that the correct choice of $v_{i+1}^j(n_{i+1})$ in 1) can also be done by stepping upwards in t, following the nodes $n_{i+1}, \ldots, n_m, \ldots, Z$.

Now it is clear that the first occurrence of $x_1(n_0, A_1)$ precedes that of $x_2(n_0, A_2)$ in h iff one of the following three conditions holds.

(i) $m=0$;

(ii) $m \geqq 1$, $v_m^1 = v_m^2$ and v_{m-1}^1 precedes v_{m-1}^2 in $\varrho(v_m)$;

(iii) $n_m = Z$, $v_m^1 = \{a\}$, $v_m^2 = \emptyset$.

This means that the order of the first occurrence of $x_1(n_0, A_1)$ and $x_2(n_0, A_2)$ in h does not depend on the subtree of t below n_0.

As a basis of our induction let $\langle x_1(F, A_1), x_2(F, A_2)\rangle \in ipgb_0(p)$. The only possible case $x_1=i$, $x_2=s$ and $A_1 \cup A_2 \in \Pi(F)$ is trivial.

Now let $\langle x_1(F, A_1), x_2(F, A_2)\rangle \in ipgb_{j+1}(p) \setminus ipgb_j(p)$ $(j \geqq 0)$, and suppose that the statement holds for every appropriate $\langle x_1(F', A_1'), x_2(F', A_2')\rangle \in ipgb_j(p')$. Again, we can restrict ourselves to the case $x_1=i$, $x_2=s$ and $A_1 \cup A_2 \notin \Pi(F)$. Then F is a right-hand side nonterminal of p. If n_0 is an occurrence of a right-hand side nonterminal of some $r \in P$, then by construction $\langle i(F, A_1), s(F, A_2)\rangle \in ipgb_{j+1}(r)$, too. Let $q \in P$ such that the left-hand side of q is F and $\langle i(F, A_1), s(F, A_2)\rangle \in ipgb_j(q)^+$. Then there exists a sequence of nonvoid ba-symbols $x_0(F_0, B_0) \ldots x_l(F_l, B_l)$ such that $x_0(F_0, B_0) = i(F, A_1)$, $x_l(F_l, B_l) = s(F, A_2)$ and $\langle x_{i-1}(F_{i-1}, B_{i-1}), x_i(F_i, B_i)\rangle \in ipgb_j(q)$ for each $i \in [l]$. Change the subtree below n_0 to an arbitrary subtree with top production q. If t' is the resulting subtree and h' is the computation sequence for t' respecting ϱ, then we have again that the order of the first occurrence of $i(F, A_1)$ and $s(F, A_2)$ is the same in h' as in h. On the other hand, the inductive hypothesis and the feasibility condition imply that for each $i \in [l]$ the first occurrence of $x_{i-1}(F_{i-1}, B_{i-1})$ precedes that of $x_i(F_i, B_i)$ in h', so we are through.

(b) ($ipgb_\pi$ is acyclic $\Rightarrow \varrho$ exists)

For $p\colon F_0 \to w_0 F_1 \ldots F_{n_p} w_{n_p} \in P$, $v_0 = A \cup B \in \Pi(F_0)$ $(A \subseteq S(F_0), B \subseteq I(F_0))$ we construct $\varrho(v_0)$ as follows. Let $s(F_{i_1}, A_1), \ldots, s(F_{i_k}, A_k)$, $i(F_{j_1}, B_1), \ldots, i(F_{j_m}, B_m)$ be all the nonvoid ba-symbols that can be reached on a reversed path from $s(F_0, A)$ in $ipgb_\pi(p)$ (provided it is a nonvoid ba-symbol). Then define $\varrho(v_0(F_0)) = v_1(F_{n_1}) \ldots v_l(F_{n_l})$ so that

(i) $v_1(F_{n_1}), \ldots, v_l(F_{n_l})$ are all the visit elements of the right-hand side nonterminals such that $F_{n_i} = F_{i_r}$ and $v_i(F_{n_i}) \cap A_r(F_{i_r}) \neq \emptyset$ for some $r \in [k]$, or $F_{n_i} = F_{j_s}$ and $v_i(F_{n_i}) \cap B_s(F_{j_s}) \neq \emptyset$ for some $s \in [m]$;

(ii) if $1 \leqq i < j \leqq l$, $x_1(F_{n_i}, A_1)$ and $x_2(F_{n_j}, A_2)$ are nonvoid ba-symbols of $v_i(F_{n_i})$ and $v_j(F_{n_j})$, respectively, then $\langle x_2(F_{n_j}, A_2), x_0(F_{n_i}, A_1)\rangle \notin ipgb_\pi(p)^+$.

Those nonvoid visit elements of the right-hand side nonterminals that are not listed in (i) for any $v \in \Pi(F_0)$ are put into $\varrho(\emptyset(F_0))$ in an arbitrary order satisfying (ii). Finally, the tail of $\varrho(\emptyset(F_0))$ is $\emptyset(F_1) \ldots \emptyset(F_{n_p})$. It is easy to check that such a ϱ is indeed a gsmv description function for Π.

The proof of the following lemma is left to the reader.

Lemma 3.9. If Π is a collection of visit sets for G such that $ipgb_\pi(p)$ is acyclic for every $p\in P$, then G is anc.

Our main theorem is now an immediate consequence of lemmas 3.8, 3.9 and remark 3.6.

Theorem 3.10. G is gsmv iff it is anc.

4. The top-down controlled l-ordered property

A deterministic finite state derivation tree automaton (dfsdt) for a context free grammar $G_0=(N, T, P, Z)$ is a triple $\mathbf{Q}=(Q, \bar{q}, \{u_p|p\in P\})$, where Q is a finite set (the set of states), $\bar{q}\in Q$ is the initial state, and if p: $F_0\to w_0F_1\dots F_{n_p}w_{n_p}\in P$, then u_p is a set of so called transition rules of the form $q_0(F_0)\to q_1(F_1)\dots q_{n_p}(F_{n_p})$ $(q_i\in Q)$, where each $q\in Q$ occurs at most once on the left-hand side of these rules. $\mathbf{Q}$ functions in the following well-known way on an input derivation tree t of G_0. It starts by assigning state $\bar{q}$ to the root of t, then, if a nonterminal node n_0 of t has already been assigned a state q_0, it assigns states (if possible) to the nonterminal sons of n_0 as follows. If n_0 is an occurrence of the left-hand side of p: $F_0\to w_0F_1\dots F_{n_p}w_{n_p}$, $n_1, \dots, n_{n_p}$ are the corresponding occurences of $F_1, \dots, F_{n_p}$, respectively and $q_0(F_0)\to q_1(F_1)\dots q_{n_p}(F_{n_p})\in u_p$, then for each $i\in[n_p]$ $\mathbf{Q}$ assigns q_i to n_i. t is accepted by $\mathbf{Q}$ iff it is able to assign a state to each nonterminal node of it in the above way.

Given an attribute grammar G and a dfsdt $\mathbf{Q}$ for G_0, a $\mathbf{Q}$-defined l-ordering of G is a partial function O which assigns for some pairs $\langle q, F\rangle$ $(q\in Q, F\in N)$ a linear order of $v(F)$.

Definition 4.1. An attribute grammar G is top-down controlled l-ordered (tcl-ordered) iff there is a dfsdt $\mathbf{Q}$ for G_0 and a $\mathbf{Q}$-defined l-ordering O of G such that for every derivation tree t of G_0 there exists a linear order O_t of the nodes of dtg (t) with the following properties.

(i) if a depends on b in a production occurring in t, then for the corresponding occurrences of a and b in dtg (t) we have $a<b$ in O_t (feasibility);

(ii) $\mathbf{Q}$ accepts all the derivation trees of G_0 (completeness);

(iii) if n is an F-labelled node in t and $\mathbf{Q}$ assigns state q to n, then $O(q, F)$ is defined, and $a<b$ in $O(q, F)$ iff $a<b$ in O_t (O is respected).

Theorem 4.2. G is gsmv iff it is tcl-ordered.

Proof. We show the equivalence of anc and tcl-ordered properties.

(a) (tcl-ordered$\Rightarrow$anc)

It is enough to prove that for every derivation tree t, if n is an F-labelled node in t and $\mathbf{Q}$ assigns state q to n, then the following statement holds. If $\langle a(F), b(F)\rangle\in ipg(p)$ for some $p\in P$ with $a\in I(F)$ and $b\in S(F)$, then $a<b$ in $O(q, F)$. We omit the proof of this statement since it is analogous to that of lemma 3.8/(a).

(b) (anc$\Rightarrow$tcl-ordered)

Let $Q=\bigcup(O(F)\ F\in N)$, where $O(F)$ is the set of l-orderings of $v(F)$. We construct $\{r_p|p\in P\}$ and $H\subseteq Q\times N$ by the following procedure.

Step 1. Put $(\bar{q}, Z)$ into H, where $\bar{q}$ is the unique l-ordering of $v(Z)$.

Step 2. If $(q_0, F_0) \in H$ (q_0 is an l-ordering of $v(F_0)$), $p: F_0 \to w_1 F_1 \dots F_{n_p} w_{n_p}$, then construct the graph $\overline{ipg}(p) = ipg(p) \cup \{(a(F_0), b(F_0)) | a \prec b \text{ in } q_0\}$. For each $i \in [n_p]$ choose an arbitrary l-ordering q_i of $v(F_i)$ that extends the partial order $\overline{ipg}(p)^+ | v(F_i)$. Then add $\{(q_i, F_i) | i \in [n_p]\}$ to H and $q_0(F_0) \to q_1(F_1) \dots q_{n_p}(F_{n_p})$ to r_p.

Repeat step 2 until no more new elements can be added to H.

It is easy to see that if G is anc, then the graph $\overline{ipg}(p)$ constructed in step 2 is always acyclic, so the automaton $\mathbf{Q} = (Q, \bar{q}, \{r_p | p \in P\})$ and the $\mathbf{Q}$-defined l-ordering $O = \{((q, F), q) | (q, F) \in H\}$ satisfy the requirements of definition 4.1.

5. Implementations of the gsmv strategy

1) Implementation using recursive procedures.

Let $G = (G_0 = (T, N, P, Z), A, v, \{r_p | p \in P\})$ a gsmv attribute grammar, Π a collection of visit sets of G and ϱ a gsmv visit description function for Π. We assume that before starting attribute evaluation we are given a structure tree, the nodes of which are represented by suitable data structures (e.g. records) with components for the attributes, references to the sons, a rule indicator that indicates the production applied to the node and a boolean flag for each visit element of the node which is true iff the visit has already been executed. The initial value of these flags is false. Then for each $p: F_0 \to w_0 F_1 \dots F_{n_p} w_{n_p} \in P$ and $v_0 \in \Pi(F_0)$ we have a

```
procedure VISIT-p-v0(n0); node (n0);
  comment ϱp(v0) = v1(Fm1) ... vl(Fml);
  begin
    compute all the i-attributes of v0 at n0
    for k = 1 to l
    comment let n1, ..., nnp be the sons of n0 having
      labels F1, ..., Fnp, respectively;
        begin
          if ¬ flag-vk · nmk then
            begin
              flag-vk · nmk = true;
              case rule indicator · nmk of
    comment let q1, ..., qj be all the productions with left-hand side Fmk;
                q1: VISIT-q1-vk(nmk);
                 ⋮
                qj: VISIT-qj-vk(nmk);
              esac
            end;
        end;
    compute all the s-attributes of v0 at n0
  end
```

2) Implementation using a system of tasks with parameters.

Here we assume that each node n of the structure tree has the following binary semaphores beyond the components (exept the flags) defined so far

(i) binary semaphores $i(n, v)$ and $s(n, v)$ for each $v \in \Pi(n)$ such that $i(n, v)$ $s((n, v))$ is *true* iff the evaluation of all the i-attributes (s-attributes, respectively) has already terminated. We do not need $i(n, v)$ $(s(n, v))$ if there are no i-attributes (s-attributes, respectively) in $v(n)$.

(ii) a binary semaphore $x(n, v)$, which is *true* iff the evaluation of v has already begun.

The initial value of these semaphores is clearly *false*. We treat semaphores $i(n, v)$ and $s(n, v)$ by the following primitives:

wait (w): L: *if* $w=0$ *then goto* L
send (w): w: $=1$

Semaphores $x(n, v)$ will be treated by the indivisible Boolean procedure $TS(x)$ (cf. [6]) of the form:

```
Boolean procedure TS(x)
  Boolean x;
    begin
      TS:=x;
      x:=true;
    end
```

Now for each p: $F_0 \to w_0 F_1 \ldots F_{n_p} w_{n_p}$ and $v \in \Pi(F_0)$ we have a task:

$T_{p,v}(n_0)$; *node* n_0;
wait $(i(m_0, v_1^0))$; ...; *wait* $(i(m_0, v_0^k))$;
comment: m_0 is the father of n_0, it is a left-hand side occurrence of $q \in P$ and $\{i(m_0, v_0^i) | i \in [k]\}$ are all the semaphores such that some $a \in v$ depends on some $b \in v_0^i$ in q;
wait $(s(m_{i_1}, v_{i_1}^1))$; ...; *wait* $(s(m_{i_l}, v_{i_l}^l))$;
comment: $m_1, \ldots, m_{n_q}$ are all the sons of $m_0 (n_0 = m_j$ for some $j \in [n_q])$, $i_s \in [n_q]$ for each $s \in [l]$ and $\{s(m_{i_s}, v_{i_s}) | s \in [l]\}$ are all the semaphores such that some $a \in v$ depends on some $b \in v_{i_s}^s$ in q;
compute all the i-attributes of v;
send $(i(n, v))$;
for $j=1$ *to* i
begin
comment: $\varrho(v) = v_1(F_{r_1}) \ldots v_t(F_{r_t}), n_1, \ldots, n_{n_p}$ are the sons of n_0 such that n_s is a left-hand side occurrence of $q_s \in P$;

if $\neg TS(x(n_{r_j}, v_j))$ *then activate* $T_{q_{r_j}, v_j}(n_{r_j})$;
end;
wait $(s(n_{r_1}, v_1))$; ...; *wait* $(s(n_{r_i}, v_i))$;
compute all the s-attributes of v;
send $s(n_0, v)$

We omitted the two case statements that should be applied to find the rules q and $\{q_{r_j} | j \in [i]\}$.

3) Implementation using a task system so that overlaps are allowed among the visit sets.

For any nonterminal F choose a production p_F such that F occurs on the right-hand side of p_F. For each s-attribute $a \in v(F)$ let $v_a = \{a, b_1, \ldots, b_m\}$, where $\{b_i | i \in [m]\}$ are all the i-attributes of F such that a depends on b_i in $ipg(p_F)$. Clearly, v_a does not depend on the choice of p_F. Let $\Pi(F) = \{v_a | a \in S(F)\}$. Now we assume that each attribute $a \in v(n)$ has own semaphores $x(n, a)$ and $y(n, a)$ at node n of the structure tree. $y(n, a)$ is true iff the evaluation of a has terminated at node n. If $a \in A_i$, then $x(n, a)$ is true iff the evaluation of a has already begun at n, and if $a \in A_s$, then $x(n, a)$ is true iff the evaluation of v_a has already begun. If p: $F_0 \to w_0 F_1 \ldots F_{n_p} w_{n_p} \in P$ and $v_a \in \Pi(F_0)$, then we have the following task:

$T_{p, v_a}(n_0)$; *node* (n_0);
  for $j = 1$ *to* m
  comment: $v_a = \{a, b_1, \ldots, b_m\}$;
  if $\neg TS(x(n_0, b_j))$ *then*
    begin wait $(y(m_0, b_1^j))$; ...; *wait* $(y(m_0, b_k^j))$;
      wait $(y(m_{i_1}^{(j)}; a_1^j))$; ...; *wait* $(y(m_{i_l}^{(j)}, a_l^j))$;
  comment: m_0 is the father of n_0, it is a left-hand side occurrence of $q \in P$, $m_1, \ldots, m_{n_q}$ are the sons of m_0 and $\{b_i^j | i \in [k]\} \cup \{a_s^j | s \in [l]\}$ are all the attributes b_j depends on in q;
    compute b_j;
    send $(y(n_0, b_j))$;
    end;
  comment: $a_1, \ldots, a_i$ are all the s-attributes a depends on in p, $n_1, \ldots, n_{n_p}$ are the sons of n_0 such that n_s is the left-hand side of $q_s \in P$, $a_j \in v(F_{r_j})$ for each $j \in [i]$;
  if $\neg TS(x(n_{r_j}, a_j))$ *then activate* $T_{q_{r_j}, a_j}$;
  wait $(y(n_{r_1}, a_1))$; ...; *wait* $(y(n_{r_i}, a_i))$;
  compute a;
  send $(y(n_0, a))$

6. NP-completeness

We have seen that the problem whether an attribute grammar is gsmv can be decided in polynomial time. However, we shall prove that it is NP-complete to decide whether an attribute grammar is generalized simple m-visit for a fixed $m \geqq 2$.

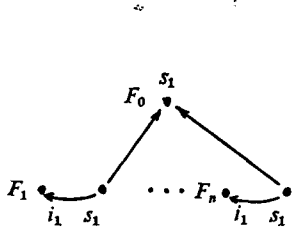
Theorem 6.1. The problem whether an attribute grammar is generalized simple 2 visit is NP-complete.

Proof: This problem is in NP, since we can guess an appropriate collection of visit sets Π by a nondeterministic algorithm in polynomial time and check whether $ipgb_\pi$ is acyclic using algorithm 3.7.

To prove NP-completeness we construct for every Boolean formula F_0 in conjunctive normal form an attribute grammar $G(F_0)$ such that F_0 is satisfiable iff $G(F_0)$ is gs-2-visit.

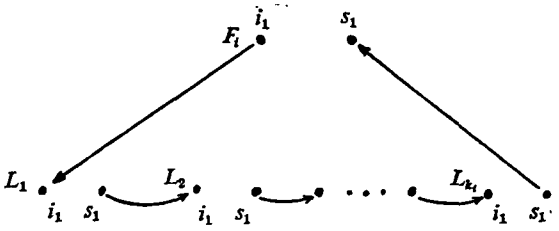
Let F_0 be a Boolean formula which is a product of sums of literals. A literal is either P or *not* P for some Boolean variable P. If $F_0 = F_1 * \ldots * F_n$ and $F_i = L_1^i + \ldots + L_{k_i}^i$, $i \in [n]$, then in $G(F_0)$ we have a nonterminal P for each variable P, nonterminals $F_1, \ldots,$

..., F_n for each factor, nonterminals L_1^i, ..., $L_{k_i}^i$, $i \in [n]$ for each literal and nontermina, F_0, which is the start symbol. If P is a variable nonterminal, then $I(P) = \{i_1, i_2\}$ $S(P) = \{s, s_1, s_2\}$, F_0 has s_1 and all the other nonterminals have i_1 and s_1. The productions and the corresponding production graphs are the following.



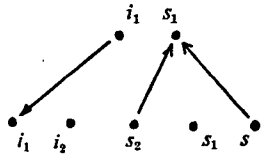
$F_0 \to F_1 * \ldots * F_n$

a)



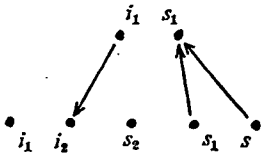
$F_i \to L_1 + \ldots + L_{k_i}$

b)



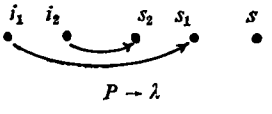
$L_j^i \to P$

c)



$L_j^i \to \underline{\text{not}}\ P$

d)

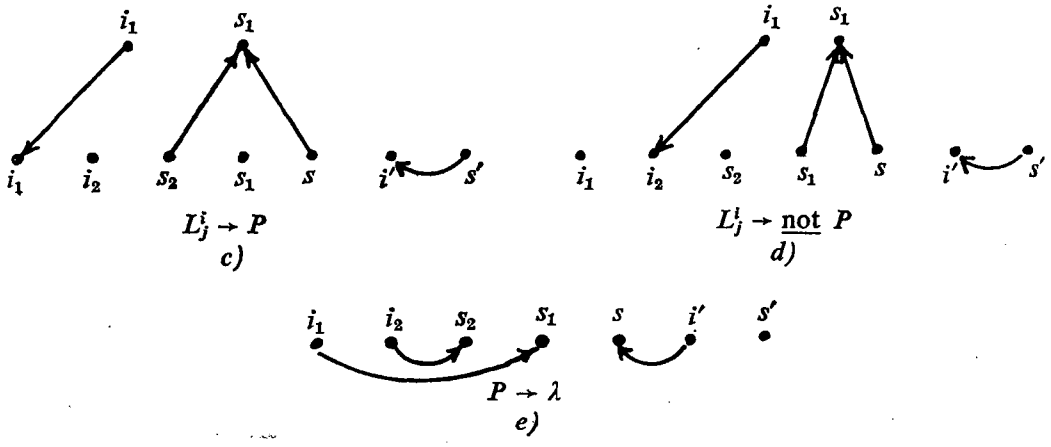


$P \to \lambda$

e)

Let $G'(F_0)$ be the attribute grammar which differs from $G(F_0)$ only in that $v(P) = \{i_1, i_2, s_1, s_2\}$ for each Boolean variable P of F_0. The *pg*'s of $G'(F_0)$ are the restrictions of the *pg*'s of $G(F_0)$ to this set of attributes. It was shown in [1] (Theorem 4.1) that $G'(F_0)$ is simple multi visit iff F_0 is satisfiable. On the other hand $G(F_0)$ is *gs*-2 visit iff $G'(F_0)$ is simple multi visit. Indeed, if $G'(F_0)$ is smv, then we can fix $\Pi(P) = \{\{i_1, s_1, s\}, \{i_2, s_2\}\}$ or $\Pi(P) = \{\{i_1, s_1\}, \{i_2, s_2, s\}\}$ for each variable P. If $G'(F_0)$ is not smv, then there are at least two occurrences of a variable P in the derivation tree of $G'(F_0)$ such that the visits $\{i_1, s_1\}$ and $\{i_2, s_2\}$ of P must be evaluated in different order at these occurrences. Hence s cannot be put into any of these two visit sets, it must be evaluated in a third visit.

To show NP-completeness for $m>2$ we only have to define $m-2$ preliminary visits to the variable nonterminals so that e.g. for $m=3$ the last three pg-graphs of the previons figure should be



Abstract

The concept of smv attribute grammar is generalized by using redundant computation sequences in the specification of visit sets and visit sequences. It is proved that these gsmv attribute grammars are the same as the absolutely noncircular ones.

An attribute grammar is top-down controlled l-ordered if there is a deterministic finite state top-down tree automaton such that for every node of every derivation tree, the order in which the attributes of the node must be evaluated is determined by the state in which the automaton passes through the node. It is proved that an attribute grammar is generalized smv iff it is top-down controlled l-ordered.

Finally it Is shown that the problem, whether an attribute grammar is gs m-visit for a fixed $m\geqq 2$ is NP-complete.

RESEARCH GROUP ON THEORY OF AUTOMATA
HUNGARIAN ACADEMY OF SCIENCES
SOMOGYI U. 7.
SZEGED, HUNGARY
H-6720

DEPT. OF COMPUTER SCIENCE
A. JÓZSEF UNIVERSITY
ARADI VÉRTANÚK TERE 1.
SZEGED, HUNGARY
H-6720

References

[1] ENGELFRIET, J. and FILE, G., Simple multi-visit attribute grammars, Journal of Computer and System Sciences, v. 24, 1982, pp. 283—314.
[2] KASTENS, U., Ordered attribute grammars, Acta Informatica, v. 13, 1980, pp. 229—256.
[3] GYIMÓTHY, T., SIMON, E. and MAKAY, A., An implementation of the HLP, Acta Cybernetica, v. 6, 1983.
[4] KENNEDY, K. and WARREN, S. K., Automatic generation of efficient evaluators for attribute grammars, Conf. Record of the Third ACM Symp. on Principles of Programming Languages, 1976, pp. 32—49.
[5] KNUTH, D. E., Semantics of context-free languages, Math. Systems Theory, v. 2, 1968, pp. 127—145.
[6] SHAW, A. C., The logical design of operating systems, Prentice-Hall, Inc., 1974.

(Received Nov. 17, 1983)