PASSES, SWEEPS AND VISITS

(extended abstract)

Joost Engelfriet and Gilberto Filè
Twente University of Technology
Enschede, The Netherlands

Introduction

This paper is a survey on recent theoretical results on attribute grammars (AG). In particular we consider multi-pass AG (both left-to-right and alternating), multi-sweep, and multi-visit AG, and for each of them we distinguish two types: the pure and the simple type. This gives rise to eight different classes of AG which are defined in section 1 by means of attribute-evaluation routines. The routines defining the pure classes are nondeterministic and, by making them deterministic, those characterizing the simple classes are obtained. In section 2 we consider the time complexity of deciding, for each of the eight classes, whether an arbitrary AG belongs to it. In particular, the decision problems for the simple classes are either polynomial-time or NP-complete. In section 3 a comparison of the formal power of the classes is made:

(i) corresponding pure and simple classes have the same power,

(ii) the formal power increases in this order: passes, alternating passes, sweeps, and visits,

(iii) the number of passes (alternating passes, sweeps, and visits) gives rise to a proper hierarchy.

In this paper no full proofs of the new results are given: those concerning simple multi-visit and pure multi-pass AG can be found in [EF2] and [EF3], respectively. The references for the other results are as follows. The simple multi-pass and alternating pass AG were introduced in [Bo] and [JW], respectively, and are studied also in [Al, RU]. The notion of sweep was considered in [Ma, Pa, EF1] and simple multi-sweep AG were defined in [Pa]. The notion of simple multi-visit AG is based on ideas in [Ka 1,2]. The idea of distinguishing between pure and simple classes originates from [Al], where the pure multi-pass/alternating pass AG are considered. The pure multi-visit (and pass) AG are studied in [Ri, RS].

Terminology

We assume the concept of attribute grammar (AG) as known and give simply some useful terminology.

The set of attributes of nonterminal X is A(X), where the inherited and the synthesized subsets are I(X) and S(X), respectively. We say that an attribute b depends on attribute a in a production p if there is a semantic rule of p defining b and using a as an argument. In each production p of the form $X_0 \to w_0X_1w_1\cdots w_{n-1}X_nw_n$, the semantic rules define all and only the attributes in $S(X_0)$ and $I(X_i)$ using as arguments

only attributes in $I(X_0)$ and $S(X_i)$, $i \in [1,n]$, (see also [Bo]). The production-graph of a production p is a graph having as nodes the attributes of the nonterminals of p and in which there is an edge from a to b iff b depends on a in p. The derivation tree-graph of a derivation tree t is obtained by pasting together, in the appropriate way, the production-graphs of the productions used in t. An AG G is noncircular if no derivation tree-graph of a tree of G contains a cycle, [Kn]. We recall briefly the definitions of left-to-right one-pass AG (L-AG), [Bo], right-to-left one-pass AG, (R-AG), [JW], and that of one-visit AG [EF1].

(i) An AG G is L (R) iff in all its production-graphs there are no dependencies which go from right to left (left to right).

(ii) An AG G is one-visit iff for each production $X_0 \to w_0X_1w_1\ldots w_{n-1}X_nw_n$, there is an ordering $X_{i_1},\ldots,X_{i_n}$ of the nonterminals $X_1,\ldots,X_n$ such that, for each $j \in [1,n]$, the attributes of X_{i_j} depend only on attributes of nonterminals X_{i_k} with $k < j$.

1. Eight classes of attribute grammars

In this section we define eight classes of AG, each of them by means of a particular attribute-evaluation strategy which can be used for every AG belonging to that class. The strategies are defined by way of programs in order to avoid more precise, but complicated, theoretical definitions. In the deterministic cases the programs are also used as practical attribute evaluation algorithms. First we stress that we consider only tree-walking evaluation strategies and, to make this idea more precise, in Fig. 1 we describe, by means of a nondeterministic recursive routine, the general form of a tree-walking strategy.

```
N(ondeterministic)-strategy-V
   proc V-evaluate (X0); node X0;
  {Let X0 → w0X1w1...wn-1Xnwn be the production applied to X0}
1. begin c(X0) := c(X0) + 1;
2.       compute some inherited attributes of X0;
3.       guess a sequence v = <v(1),...,v(m)>, such that m ≥ 0 and for each
         i ∈ [1,m], v(i) ∈ [1,n];
4.       for i := 1 to m do
5.           V-evaluate (Xv(i));
6.       compute some synthesized attributes of X0
   end;
{Main program} begin while c(root) < k0 do V-evaluate (root)  end
```

Figure 1. General tree-walking attribute-evaluation strategy.

Let t be a derivation tree of an AG G and assign to each nonterminal node X of t a counter c(X), initially set to 0. If the routine V-evaluate is called with the root

of t as parameter, the effect of any possible computation is the evaluation of some attributes of t. At the end the counter c(X), of any node X of t, will contain the number of calls V-evaluate(X) which have been executed, or, in other words, how many visits have been paid to the subtree rooted in X, during the computation. Observe that the routine V-evaluate is nondeterministic in two ways:

Nondeterminism of type a: at each visit to a node X_0, it can choose which attributes of X_0 to evaluate (lines 2 and 6).

Nondeterminism of type b: it guesses in which order to visit the sons of X_0, i.e., it guesses $v = \langle v(1),\ldots,v(m)\rangle$, which we call a visit-sequence (cfr. line 3).

The first class of AG we define is that of pure multi-visit AG which uses the full power of N-strategy-V. An AG G is pure k-visit iff, for each derivation tree t of G, there is a possible computation of N-strategy-V on t which computes all attributes of t and such that for no node X of t the counter c(X) becomes larger than k (in particular, $k_0 \leq k$). An AG is pure multi-visit if it is pure k-visit for some $k \geq 1$. In [RS] it has been proven that every noncircular AG G is pure k-visit for some k. In fact, if nonterminal X has n attributes, then at most n visits to X (i.e. calls of V-evaluate(X)) are necessary.

The remainder of the section consists of two parts. First we restrict the nondeterminism of type b in N-strategy-V obtaining three other nondeterministic strategies (N-strategies-S, A, and P). Secondly, by making these four strategies deterministic (eliminating the remaining nondeterminism of type b and all nondeterminism of type a) we obtain D-strategies-V, S, A, and P. Let $Y \in \{\text{visit, sweep, alter.pass, pass}\}$. The nondeterministic strategies define the classes of pure multi-Y AG, whereas the deterministic ones define the classes of simple multi-Y AG. Considering that the second strategies are obtained restricting the first ones, it is obvious that simple multi-Y AG $\subseteq$ pure multi-Y AG.

Restricting nondeterminism of type b

We modify the routine V-evaluate of Fig. 1 so as to obtain from it three routines, S-, R-, and L-evaluate which we will use later for defining N-strategies-S, A, and P. The changes of V-evaluate are as follows: eliminate line 1 and restrict the guess of the visit-sequence $v = \langle v(1),\ldots,v(n)\rangle$ of line 3 in one of the following ways:

(i) for S-evaluate: the visit-sequence must be a permutation of the sequence $1,\ldots,n$, i.e. $v = \langle v(1),\ldots,v(n)\rangle$, where for each $i \in [1,n]$, $v(i) \in [1,n]$ and for $j \in [1,n]$ and $j \neq i$, $v(j) \neq v(i)$,

(ii) for R-evaluate: the visit sequence can be only $v_R = \langle n,\ldots,1\rangle$,

(iii) for L-evaluate: the visit-sequence can be only $v_L = \langle 1,\ldots,n\rangle$.

The routine S-evaluate, when called with the root of a derivation tree t as argument, visits each node of t exactly once. We call such a walk through t a sweep. The routines R-, and L-evaluate are particular cases of S-evaluate: they perform sweeps through t such that t is visited in a depth-first fashion from right-to-left or left-to-right,

respectively. Such sweeps we call a right-to-left (R-) and a left-to-right (L-) pass, respectively.

N-strategy-S. {Main program} begin for c := 1 to k do S-evaluate(root) end.

N-strategy-A. Fix some $M = \langle M_1,\ldots,M_k \rangle$, where $M_i \in \{L,R\}$, for each $i \in [1,k]$.
{Main program} begin for c := 1 to k do M_c-evaluate(root) end.

N-strategy-P. {Main program} begin for c := 1 to k do L-evaluate(root) end.

Note that the alternating passes need not really alternate between L and R passes, as defined in [JW], but in M we allow a free sequence of L's and R's for the k-passes. It is easy to see that for both cases the concepts of pure multi-alternating pass are the same, but the concepts of pure k-alternating pass are, in general, different.

The classes of AG corresponding to the above three nondeterministic strategies are defined as follows. An AG G is pure k-sweep (alternating pass, pass) iff all attributes of any derivation tree t of G can be evaluated by a possible computation of N-strategy-S (A, P, respectively). An AG is pure multi-sweep (alternating pass, pass) if it is pure k-sweep (alternating pass, pass) for some $k > 0$.
Note that for any possible computation of N-strategy-S on a tree t (N-strategies-A, and P are particular cases) the variable c of the loop in the main program plays the role of the counters at each node of t (in N-strategy-V). This is the reason for eliminating line 1 of V-evaluate in constructing S-, R-, and L-evaluate.

Eliminating all nondeterminism

Nondeterminism of type a is eliminated from all four N-strategies by fixing a partition $A_1^{X_0},\ldots,A_{k(X_0)}^{X_0}$ of the attributes of each nonterminal X_0, and changing the four strategies in such a way that in the j-th visit to X_0, $j \in [1,k(X_0)]$, all attributes of $A_j^{X_0}$ are evaluated instead of some attributes of X_0 (cfr. lines 2 and 6 of V-evaluate, Fig. 1). Eliminating nondeterminism of type a makes N-strategies-A and P deterministic, but, for N-strategies-V and S we must take care also of the type b nondeterminism as follows. For each production $X_0 \to w_0X_1w_1\ldots w_{n-1}X_nw_n$ and $j \in [1,k(X_0)]$, we fix, (i) for N-strategy-V, a visit-sequence $v_j = \langle v_j(1),\ldots,v_j(m_j)\rangle$, where $m_j \geq 0$ and, for each $i \in [1,m_j]$, $v_j(i) \in [1,n]$; (ii) for N-strategy-S a visit-sequence $v_j = \langle v(1),\ldots,v(n)\rangle$, which is a permutation of the sequence $1,\ldots,n$. Then, we change V- and S-evaluate in such a way that, when visiting X_0 for the j-th time, the nonterminals $X_1,\ldots,X_n$ are visited according to the visit-sequence v_j.

We now define the classes of simple multi-Y AG (Y $\in$ {visit, sweep, alter. pass, pass}), by means of D-strategies-V,S,A, and P, respectively.

An AG G is simple k-visit iff, (a) for each nonterminal X of G there is a partition $A_1^X,\ldots,A_{k(X)}^X$ of its attributes, where $k(X) \leq k$; (b) for each production

$X_0 \to w_0X_1w_1,\ldots,w_{n-1}X_nw_n$ of G and $j \in [1,k(X_0)]$ there is a visit-sequence $v_j = \langle v_j(1),\ldots,v_j(m_j)\rangle$ as in point (i) above; such that D-strategy-V, described below, using the partitions of (a) and the visit-sequences of (b) evaluates all attributes of any derivation tree of G.

D-strategy-V

```
   proc sV-evaluate (X0); node X0;
   {Let X0 → w0X1w1...wn-1Xnwn be the production applied to X0}
1. begin c(X0) := c(X0) + 1; let c = c(X );
2.       compute the inherited attributes in A_c^X0;
3.       for i := 1 to mc do {the visit-sequence vc has length mc}
4.           sV-evaluate (Xvc(i));
5.       compute the synthesized attributed in A_c^X0
   end
  {Main program} begin while c(root) < k(root) do sV-evaluate(root); end
```

Note that in sV-evaluate the value of $c(X_0)$ is not allowed to become larger than $k(X_0)$. The class of simple multi-visit AG is inside that of absolutely noncircular AG (ANC-AG) [KW]. The concept of plan used in the attribute evaluation algorithm for ANC-AG is very close to the attribute partitions and the visit-sequences needed in D-strategy-V, but the algorithm for ANC-AG, in order to decide which plan to use, needs to store more information at each node than just the visit number (cfr. [EF2, Ka2]).

As we did before with V-evaluate, we transform now sV-evaluate in order to obtain three routines, sS-, sR-, and sL-evaluate which we will use later for defining D-strategies-S, A, and P.
These are the changes on sV-evaluate: (a) eliminate line 1; (b) line 3 becomes, for i := 1 to n do ; (c) line 4 becomes:

(i) for sS-evaluate: sS-evaluate $(X_{v_c(i)})$, where v_c is a permutation of $1,\ldots,n$ and c is the global counter of the main program,

(ii) for sR- and sL-evaluate: sR-evaluate$(X_{v_R(i)})$ and sL-evaluate$(X_{v_L(i)})$, respectively, where v_R and v_L are the visit-sequences $v_R = \langle n,\ldots,1\rangle$ and $v_L = \langle 1,\ldots,n\rangle$.

An AG G is simple k-sweep iff (a) for each nonterminal X there is a partition $A_1^X,\ldots,A_k^X$ of the attributes of X; (b) for each production $X_0 \to w_0X_1w_1\ldots w_{n-1}X_nw_n$ of G and $j \in [1,k]$, there is a visit-sequence $v_j = \langle v_j(1),\ldots,v_j(n)\rangle$, which is a permutation of the sequence $1,\ldots,n$; such that D-strategy-S, described below, using the partitions of (a) and the visit-sequences of (b), evaluates all attributes of any derivation tree of G.

D-strategy-S. {Main program} begin for c := 1 to k do sS-evaluate (root) end

Observation 1.1. Consider the two nondeterministic strategies we would obtain from N-strategies-V and S eliminating type a nondeterminism from them, like we have just done, but keeping type b nondeterminism. Call them V_1 and S_1, respectively. It is proved in [EF2] that the class of AG characterized by V_1 is still the class of simple multi-visit AG. It is possible to prove that also S_1 defines the class of simple multi-sweep AG (cfr. [EF1] for the one-sweep case). This implies that what is really important, in going from pure to simple classes, is eliminating nondeterminism of type a.

An AG G is simple k-alternating pass iff there exists a sequence $M = \langle M_1,\ldots,M_k\rangle$, where $M_i \in \{L,R\}$, and a partition $A_1^X,\ldots,A_k^X$ of the attributes of each nonterminal X of G such that D-strategy-A, described below, using this sequence M and these partitions, evaluates all attributes of any derivation tree of G.

D-strategy-A. {Main program} begin for c := 1 to k do sM_c-evaluate (root) end

An AG G is simple k-pass iff for each nonterminal X there is a partition $A_1^X,\ldots,A_k^X$ of its attributes such that D-strategy-P, which is as follows:
{Main program} begin for c := 1 to k do sL-evaluate (root) end
evaluates all attributes of any derivation tree of G.

As in the pure case, an AG is simple multi-visit (sweep, alter. pass, pass) if it is simple k-visit (sweep, alter. pass, pass) for some k > 0.
Note that pure 1-pass = simple 1-pass = L-AG, that pure 1-alter.pass = simple 1-alter.pass = L-AG ∪ R-AG, and that pure 1-visit = pure 1-sweep = simple 1-visit = = simple 1-sweep = one-visit AG (cfr. Observation 1.1).

2. Time-complexity

In this section, for each of the eight classes defined in part 1, we will consider the time-complexity of the following three problems. Let Y ∈ {visit, sweep, alternating pass, pass}.
Problem 1. Decide whether any given AG is pure/simple multi-Y.
Problem 2. Decide for a fixed k > 0, whether any given AG is pure/simple k-Y.
Problem 3. Decide whether, given an AG G and an integer k > 0, G is pure/simple k-Y.
Note that Problem 3 is closely related to the following optimization problem.
Problem 4. Find, for a given pure/simple multi-Y AG G, the smallest k such that G is pure/simple k-Y. Actually, the results about Problem 3 in both tables of Fig. 2 also hold for Problem 4. (Problems 3 and 4 can be reduced to each other).
The results of Fig. 2 concerning simple multi-pass, simple multi-alternating pass and pure multi-visit are (or can be derived) from [Bo], [JW, RU] and [RS], respectively. The other results are ours. Those about pure multi-pass/alternating pass/sweep are or can be obtained from [EF1,3]. The results about simple multi-sweep/visit

Table 1. Simple classes.

simple multi-	Problem 1	Problem 2	Problem 3
pass	polynomial (1)	polynomial (2)	polynomial (2)
alter. pass		polynomial (3)	NP-complete (6)
sweep		NP-complete for k≥2, for k=1 polynomial (5)	NP-complete (5)
visit	NP-complete (4)	NP-complete for k≥2, for k=1 polynomial (4)	NP-complete (4)

Table 2. Pure classes.

pure multi-	Problem 1	Problem 2	Problem 3
pass	exponential lower and upper bounds (1)	polynomial (1)	decidable (2)
alter. pass			
sweep	exponential lower bound decidable? (1)	decidable (2)	
visit	trivial assuming noncircularity (3)		

Figure 2. Time-complexity of Problems 1, 2 and 3 for the eight classes of AG we are studying. The number at the bottom-right corner of each box indicates where the corresponding result is explained in the remainder of the section.

are new. We start by considering table 1 of Fig. 2.

Table 1: Simple classes.

Result 1. We show in one stroke that Problem 1 is polynomial for simple multi-pass/alter.pass/sweep. Actually we prove a more general result which applies to all those AG which can be called simple multi-Yphase, where Y is some class of AG. We need first some terminology and then we define precisely what a simple multi-Yphase AG is.

For an AG G, we denote by A^G the set of all attributes of G. The simple dependency graph of G (sdg(G)), see [Al, RU], is the graph whose nodes are the attributes in

A^G and in which there is an edge from node a to node b iff attribute b depends on attribute a in some production of G. Observe that, by considering all the strongly connected components of sdg(G), we define a partition of the attributes of G. We call this the strong partition of the attributes of G.

For $B \subseteq A^G$, the subgrammar G_B of G is the AG defined as follows: (a) G_B has the same underlying CFG as G; (b) the attributes of any nonterminal X in G_B are those attributes of X in G which are in B; (c) the semantic rules in G_B of any production $p: X_0 \to w_0 X_1 w_1 \ldots w_{n-1} X_n w_n$, are those semantic rules of p in G which define those attributes of $S(X_0)$ and $I(X_i)$ which are also in B, $i \in [1,n]$. If some of these semantic rules of p in G use as arguments attributes which are not in B, then these arguments are changed into appropriate constants (cfr. [Ri]).

Definition 2.1. Let Y be a class of AG. An AG G is simple k-Yphase iff there is a partition $B_1,\ldots,B_k$ of the attributes of G such that:
(1) the subgrammar G_{B_i} is in Y;
(2) let phase(a) = i iff attribute $a \in B_i$; if attribute b depends on attribute a in some production of G, then phase(a) $\leq$ phase(b).

□

Intuitively this means that for simple k-Yphase AG there is an attribute-evaluation strategy which consists of k consecutive applications of the strategy for the class Y. In the following Theorem, which generalizes results of [Al,RU] about simple multi-pass/alter.pass AG, we show that Problem 1 can be solved for simple multi-Yphase AG.

Theorem 2.1. Let Y be a class of AG closed under the operation of taking subgrammars. An AG G is simple multi-Yphase iff $G_{A_i} \in Y$ for all $i \in [1,m]$, where $A_1,\ldots,A_m$ is the strong partition of the attributes of G.

Proof: $\Longrightarrow$ Assume that G is simple multi-Yphase. Then there is a partition $B_1,\ldots,B_k$ of the attributes of G satisfying Definition 2.1. Definition 2.1(2) implies that all attributes in a strongly connected component of sdg(G) must have the same phase number, i.e., for every $i \in [1,m]$ there is a $j \in [1,k]$ such that $A_i \subseteq B_j$. Hence because $G_{B_j} \in Y$ by Definition 2.1(1), also $G_{A_i} \in Y$ (Y is closed under taking subgrammars).
$\Longleftarrow$ There is at least one ordering $A_{i_1},\ldots,A_{i_m}$ of the elements $A_1,\ldots,A_m$ of the strong partition of G such that, if an attribute in A_{i_j} depends (in some production of G) on an attribute in A_{i_k}, then $k < j$. Now, let $B_j = A_{i_j}$ for all $j \in [1,m]$. This partition satisfies Definition 2.1.

□

Corollary 2.1. Let $T_Y(n) \geq n$ be the time taken for deciding whether an AG G of size n is in the class Y. The time for deciding whether any given AG G of size m is simple multi-Yphase is bounded by $p(T_Y(m))$, where p is a polynomial.

□

Corollary 2.2. The time complexity of Problem 1 for simple multi-pass/alter. pass/sweep AG is polynomial.

Proof: It is easy to see from their D-strategies that these three classes are simple multi-Yphase, where Y is the class of L-, L- or R-, and one-visit AG, respectively. Note also that these classes are closed under the operation of taking subgrammars and that Problem 1 can be decided in polynomial time for them (cfr. [Bo, JW, EF1], respectively). □

Result 2. The fact that, for the class of simple multi-pass AG, Problems 2 and 3 are also polynomial, is immediate from the existence of a polynomial algorithm which finds the optimal pass-assignment to the attributes of any AG in this class, [Bo, Al].

Result 3. Problem 2 can be decided in polynomial time for simple multi-alter. pass AG in the following way: let k be the constant and G any AG; (i) consider all possible sequences $M = \langle M_1,\ldots,M_k\rangle$, where each M_i is either L or R, there are 2^k such sequences, (ii) it is possible, with an algorithm similar to those in [Bo, JW, Al], to test in polynomial time whether one such sequence works (cfr. D-strategy-A).

Result 4, 5, and 6. First we show that all six problems are in NP with a similar argument. This is because deciding each instance of these problems, for a given AG G, can be done, roughly, (1) by guessing a partition of the attributes of each nonterminal of G, and (2) by testing whether it satisfies the necessary conditions. Operation (1) for Problems 2 and 3 takes polynomial time because the k specified in the problems gives a bound on the number of elements of the partitions to be guessed. In [EF2] it is proved that a similar bound exists also for Problem 1 in case of simple multi-visit AG. For passes and sweeps, operation (2) takes polynomial time because it takes polynomial time to decide whether a given partition satisfies conditions (1) and (2) of Definition 2.1. A polynomial test exists also for simple multi-visit AG, [EF2].

Result 4. Problems 1, 2 and 3 are NP-hard for simple multi-visit AG. This can be shown by proving that, for each instance of a boolean formula F, it is possible to construct an AG G_F such that, G_F is simple multi-visit $\Longrightarrow$ F is satisfiable $\Longrightarrow$ G_F is simple 2-visit [EF2].

Result 5. Problems 2 and 3 are NP-hard for simple multi-sweep AG. This result is shown by reducing the k-colorability problem (for fixed $k \geq 3$) to Problem 2, also for $k \geq 3$ (the case $k = 2$ needs a different proof). The idea of the reduction is as follows. In the k-colorability problem two nodes which are connected cannot have the same color. It is possible to construct in polynomial time, for any given undirected graph H, an AG G such that in the sdg(G) there is one strongly connected component A_n for each node n of H and, furthermore, there is an edge (n, n') in H iff the subgrammar $G_{A_n \cup A_{n'}}$ of G (corresponding to the attributes in $A_n \cup A_{n'}$) is not one-visit.

Result 6. Problem 3 is NP-hard for simple multi-alter.pass AG. The proof [RU] consists of reducing the shortest common supersequence problem to this problem.

Table 2: Pure classes.

Result 1. The exponential lower and upper bounds for Problem 1 in the case of pure multi-pass AG have already been stated in [EF1]. Both these results can be extended to pure multi-alter.pass AG, whereas only the first can be extended to pure multi-sweep AG for which we do not know yet any algorithm for deciding Problem 1. In [EF1] we also mentioned that Problem 2 for pure multi-pass AG was polynomial. That the same holds also for pure multi-alter.pass AG is shown similarly.

Result 2. Problem 3 for pure multi-pass/alter.pass AG is shown to be decidable in [EF3] and a decision method for Problems 2 and 3 for pure multi-visit AG is given in [RS] . Here we describe a uniform decision method for all Problems 2 and 3. Let dt(G) denote the set of derivation trees of an AG G.

Lemma 2.1. Let Z $\in$ {visit, sweep, alter.pass, pass} and let G be an AG. It is possible to construct, for any k > 0 (and in case Z = alter.pass, for a sequence of k L's and R's), an AG G' which is simple k-Z and such that, for some projection Π : nonterminals of G' $\rightarrow$ nonterminals of G, $\Pi(dt(G')) = S_k$, where S_k is the set of those derivation trees t of G such that there is a computation of N-strategy-Z on t which evaluates all the attributes of t and performs k Z's.

Proof: We only consider the case Z $\in$ {sweep, alter.pass, pass} . For Z = visit a similar proof can be given. The intuition behind the construction is as follows. Consider a derivation tree t of G such that all its attributes can be evaluated through a computation Ψ of N-strategy-Z on t which performs k Z's. Relabel each node X of t with (X,par^X), where par^X is the partition $A_1^X,\ldots,A_k^X$ of the attributes of X such that the attributes in each A_i^X have been evaluated during the i-th visit of Ψ to X. Let t' be the so obtained tree, and let production $p : X_0 \rightarrow w_0X_1w_1\ldots w_{n-1}X_nw_n$ of t correspond in t' to p': $(X_0,par^{X_0}) \rightarrow w_0(X_1,par^{X_1})\ w_1\ldots w_{n-1}(X_n,par^{X_n})w_n$. Assume that each nonterminal (X_j,par^{X_j}) of p' has the same attributes as X_j of p and that p' has the same semantic rules as p, apart from the renaming of the nonterminals. If we consider p' as the only production of an AG and we use Definition 2.1 for it, then, the partition of all attributes of p' determined by $par^{X_0},\ldots,par^{X_n}$, satisfies conditions (1) and (2) of Definition 2.1, where Y is L, L or R, one-visit if Z is pass, alter.pass, sweep, respectively. This is because the partitions $par^{X_0},\ldots,par^{X_n}$ originate from the actual k-Z computation Ψ. We say that p' is a simple k-Yphase production.

The construction of G' from G is as follows.

(1) G' has nonterminals (X,par^X) where X is a nonterminal of G and par^X a partition of the attributes of X.

(2) For each production $p : X_0 \rightarrow w_0X_1w_1\ldots w_{n-1}X_nw_n$ of G, G' has all the productions $p' : (X_0,par^{X_0}) \rightarrow w_0(X_1,par^{X_1})w_1\ldots w_{n-1}(X_n,par^{X_n})w_n$ which are simple k-Yphase productions with the indicated partitions. The semantic rules of p' are obtained from those of p by the appropriate renaming of the nonterminals.

Condition (2) with Y equal to L, L or R, one visit, guarantees that G' is simple multi-pass/alter.pass/sweep, respectively. The above intuition should be sufficient to see that $\Pi(dt(G')) = S_k$, where Π simply maps each (X,par^X) of G' into X of G. □

To decide whether an AG G is pure k-Z (Z ∈ {visit,sweep,alter.pass,pass}) use Lemma 2.1 to construct from G the AG G' which is simple k-Z, and test whether $\Pi(dt(G')) = dt(G)$. By Lemma 2.1 the answer is yes iff G is pure k-Z. It is well known that such questions on derivation trees of CFG's are decidable [Th]. Note that in case Z = alternating pass, we must do the two operations above for all sequences of k L's and R's (for each of them there is a G'). G is pure k-alter.pass iff at least for one such sequence the answer of the above test is positive.

Result 3. In [RS] it has been shown (see also section 1) that all noncircular AG are pure k-visit for some k > 0. Therefore testing whether an AG is pure multi-visit (Problem 1) is trivial if we know it to be noncircular, otherwise it is sufficient to test whether G is noncircular, which takes exponential time [JOR].

3. Formal power

In this section we investigate the relative formal power of all considered classes of AG (see also [Ri, RS]). Two concepts are at the basis of such a comparison: (i) that of translation defined by an AG, by means of which we can measure the formal power of a class of AG; (ii) that of semantic domain of an AG, which we use in order to compare only AG with the same operations in the semantic rules. This is needed because, with unrestricted operations in their semantic rules, even AG with only synthesized attributes are as powerful as Turing Machines. The precise definitions are as follows.

A semantic domain D is a collection of sets together with a collection of operations defined on these sets. We say that an AG G is over D if its sets of attribute values are taken from D, and its semantic rules use operations from D only.

For a (noncircular) AG G and a derivation tree t of G, $m_G(t)$ denotes the value of a designated attribute of the root of t. The translation T_G defined by G is $\{(w,v) \mid$ w is the yield of t and $v = m_G(t)$, for some derivation tree t of G$\}$. For a class Y of AG and a semantic domain D, $T(Y,D) = \{T_G \mid G \in Y, G$ is over D$\}$ is the class of Y-translations over D.
For classes Y and Z of AG, Y has less or equal power than Z (denoted by $Y \leq Z$) iff for every semantic domain D, $T(Y,D) \subseteq T(Z,D)$. Similarly, Y and Z have the same power if their classes of translations are equal over every semantic domain.

Our first result is that corresponding pure and simple classes have the same power.

Theorem 3.1. Let $k > 0$ and $Y \in \{visit, sweep, alter.pass, pass\}$. The classes of pure k-Y AG and simple k-Y AG have the same power.

Proof. We have to show that, on every semantic domain D, T(pure k-Y,D) $\subseteq$ T(simple k-Y,D). This follows directly from Lemma 2.1: if G is pure k-Y, then the produced simple k-Y AG G' is such that $\Pi(dt(G')) = dt(G)$. Hence, because the semantics are the same, $T_{G'} = T_G$.

□

Our second result compares the power of visits, sweeps, alter.passes, and passes, and considers the influence on power of increasing k. Denote, for $k > 0$, by kV, kS, kA and kP the classes of (pure and simple) k-visit, k-sweep, k-alter.pass and k-pass AG, respectively; denote by mV, mS, mA and mP the corresponding "multi-classes". Note that mV has the same power as the class of all AG. (Note that then the same holds for the absolutely noncircular AG [KW]).

Theorem 3.2. The "power diagram" of Fig. 3 is correct, in the sense that an ascending line from Y to Z means that $Y \leq Z$ but not $Z \leq Y$, whereas, if Y and Z are unconnected, it means that they are incomparable with respect to $\leq$.

□

Let us sketch the proof of the theorem. The $\leq$-inclusions in Fig. 3 are obvious from the definitions. To show all incomparabilities and proper inclusions it suffices to prove that the following four statements do not hold.

(1) $(k+1)\,P \leq kV$, for any $k \geq 1$,

(2) $1A \leq mP$,

(3) $1S \leq mA$,

(4) $2V \leq mS$.

Now, to prove that not $Y \leq Z$, we define the semantic domain STR and prove that not T(Y,STR) $\subseteq$ T(Z,STR). The semantic domain STR contains one set, viz. the set of all strings over alphabet $\{0,1,\#\}$, and contains an operation f_v on this set for every $v \in \{0,1,\#\}^*$, viz. $f_v(u) = uv$, for every $u \in \{0,1,\#\}^*$, i.e., right-concatenation. Note that, for an AG G over STR, T_G translates strings into strings.

Thus we have to prove that, over the semantic domain STR,

(1) there is a (k+1)-pass translation which is not k-visit,

(2) there is a 1-alternating pass translation which is not multi-pass,

(3) there is a 1-sweep translation which is not multi-alternating pass,

(4) there is a 2-visit translation which is not multi-sweep.

Let L_0 be a nonregular, context-free language over alphabet $\{0,1\}$. The examples we use to prove (1) to (4) are the translations (1) to (4) described below (u^R denotes the reverse of u).

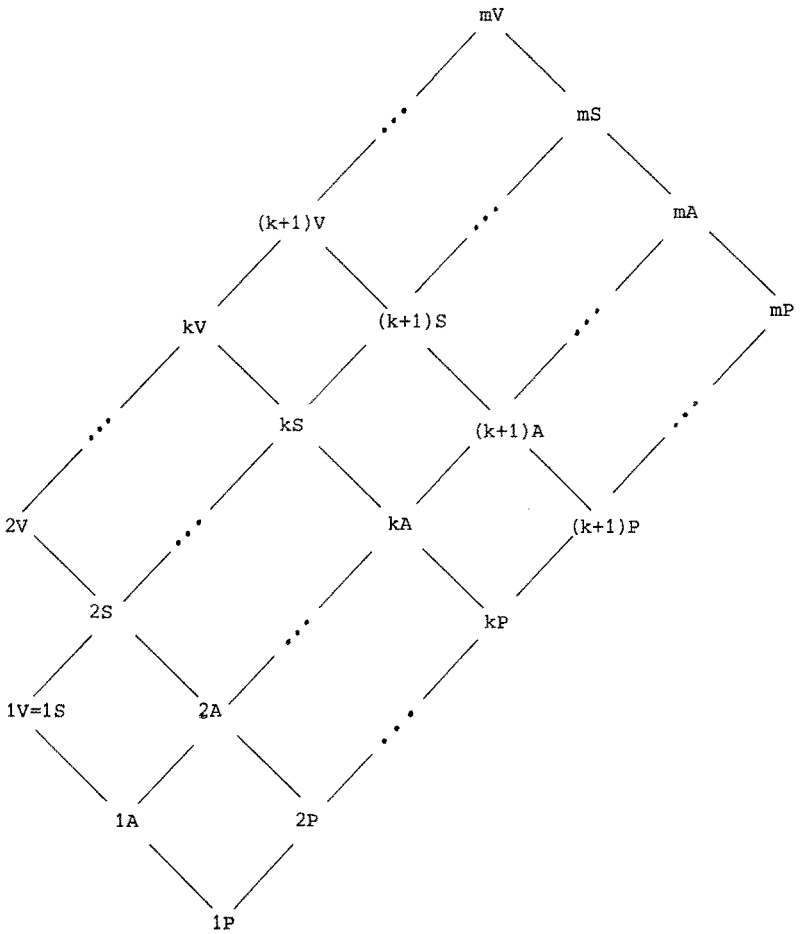

Figure 3. The formal power of visit, sweeps, and passes.

(1) $\{<\#u\#v\#,\ \#(u\#v\#)^{k+1}> \mid u,v \in L_0\}$

(2) $\{<\#u_1\#u_2\#\ldots\#u_n\#,\ \#u_n^R\#\ldots\#u_2^R\#u_1^R\#> \mid n \geq 1,\ u_i \in L_0\}$

(3) $\{<\#u_1\#v_1\#\ldots\#u_n\#v_n\#,\ \#v_1\#u_1\#\ldots\#v_n\#u_n\#> \mid n \geq 1,\ u_i, v_i \in L_0\}$

(4) $\{<\#u_1\#v_1\#\ldots\#u_n\#v_n\#,\ \#u_1\#v_1\#u_1\#\ldots\#u_n\#v_n\#u_n\#> \mid n \geq 1,\ u_i, v_i \in L_0\}$.

It should be clear that these translations can be defined by a (k+1)-pass,1-alter.pass (with a right-to-left pass),1-sweep and 2-visit AG, respectively.As an example we give a rough idea of why translation (2) is not a multi-pass translation. Consider any AG G defining translation (2). Consider a derivation tree t with the string $\#u_1\#u_2\#\ldots\#u_n\#$ as yield, where all u_i are different. Since all operators in STR are unary, the derivation tree-graph of t consists of one long oriented path Π, ending at the designated attribute of the root of t. Path Π is approximately of the form

$\Pi_n \ldots \Pi_2 \Pi_1$, where Π_i is the part of Π which adds $\# u_i^R \#$ to the output string. Let X_i be a node of t producing symbols in u_i only (we may assume that such nodes exist by the nonregularity of L_0). It should now be intuitively clear that Π_i has to visit X_i; otherwise we replace the subtree with root X_i by one with a different yield ($\in \{0,1\}^*$), and thus remove u_i from the input string, keeping it in the output string. Thus Π visits $X_n, X_{n-1}, \ldots, X_2, X_1$ in that order. Since X_{i+1} lies to the right of X_i in t, Π "goes left" at least n-1 times. Hence it takes at least n left-to-right passes to evaluate the attributes of t. Since n is unbounded, this shows that G is not multi-pass, and hence there is no multi-pass AG defining translation (2).

The other results can be proven in the same way, showing corresponding "bad" properties of the path Π.

Using results on tree transducers it can be shown (cf. [EF3]) that statements (1) and (4) even hold for output sets instead of translations (the output set of an AG G is the range of T_G). Thus, there is a (k+1)-pass output set which is not a k-visit output set, and there is a 2-visit output set which is not multi-sweep. For output sets, the nodes 1S, 1A, and 1P of Fig. 3, collapse into one node [EF1]. We conjecture that, for output sets, statements (2) and (3) hold with "1" replaced by "2", and thus no other nodes in the power diagram collapse.

References

[Al] H. Alblas; A characterization of attribute evaluation in passes; Memorandum 315, Twente University of Technology, 1980.

[Bo] G.V. Bochmann; Semantic evaluation from left to tight; CACM 19 (1976), 55-62.

[EF1] J. Engelfriet, G. Filè; Formal properties of one-visit and multi-pass attribute grammars; Proc. of the 7th ICALP at Noordwijkerhout, pp. 182-194, 1980.

[EF2] J. Engelfriet, G. Filè; Simple multi-visit attribute grammars; Memorandum 314, Twente University of Technology, 1980.

[EF3] J. Engelfriet, G. Filè; Passes and paths of attribute grammars; Memorandum 323, Twente University of Technology, 1980.

[JOR] M. Jazayeri, W.F. Ogden, W.C. Rounds; The intrinsically exponential complexity of the circularity problem for attribute grammars; CACM 18 (1975), 697-706.

[JW] M. Jazayeri, K.G. Walter; Alternating semantic evaluator; Proc. ACM 1975 Ann. Conference, pp. 230-234.

[Ka1] U. Kastens; Ordered attributed grammars; Bericht Nr. 7/78, Universität Karlsruhe, 1978.

[Ka2] U. Kastens; Ordered attributed grammars; Acta Informatica 13 (1980), 229-256.

[Kn] D.E. Knuth; Semantics of context-free languages; Math. Syst. Theory 2 (1968), 127-145. Correction: Math. Syst. Theory 5 (1971), 95-96.

[KW] K. Kennedy, S.K. Warren; Automatic generation of efficient evaluators for attribute grammars; Conf. Record of 3^d Symp. on Principles of Programming Languages, pp. 32-49, 1976.

[Ma] B.H. Mayoh; Attribute grammars and mathematical semantics; Report DAIMI PB-80, Aarhus University, 1978.

[Pa] R. Parchmann; Grammatiken mit Attributenschema und zweistufige Auswertung attributierter Grammatiken; Bericht Nr. 46, Technische Hochschule Aachen, 1978.

[Ri] H. Riis; Subclasses of attribute grammars; Report DAIMI PB-114, Aarhus University, 1980.

[RS] H. Riis, S. Skyum; k-Visit attribute grammars; Report DAIMI PB-121, Aarhus University, 1980.

[RU] K. -J. Räihä, E. Ukkonen; On the optimal assignment of attributes to passes in multi-pass attribute evaluators; Proc. of the 7th ICALP at Noordwijkerhout, pp. 500-511, 1980.

[Th] J.W. Thatcher; Tree automata: an informal survey; In "Currents in the Theory of Computing" (A.V. Aho, ed.), Prentice-Hall (1973).