Acta Informatica 29, 161–210 (1992)

Acta Informatica

Context-free hypergraph grammars have the same term-generating power as attribute grammars

Joost Engelfriet and Linda Heyker

Department of Computer Science, Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands

Abstract. Context-free hypergraph grammars and attribute grammars generate the same class of term languages. Extending the context-free hypergraph grammar with a context-free grammar and a semantic domain, a syntax-directed translation device is obtained that is equivalent to the attribute grammar.

1. Introduction

Graph grammars are of interest because they provide a formalism to express the manipulation, generation, and description of graphs. For this reason they are used in very diverse areas of computer science (see, e.g., [EhrNagRosRoz]). A simple and attractive notion of context-free graph grammar was introduced in [Fed] and re-introduced (in a more readable formulation) and studied in e.g., [BauCou; HabKre1 and 2; Cou3 and 4; MonRos; LenWan; Hab]: the context-free hypergraph grammar (cfhg), that manipulates directed hypergraphs rather than graphs.

An attractive feature of (hyper)graphs is that they can represent many other structures, such as strings. Thus, graph grammars can be used as a string-generating device. In this way one obtains a context-free way of generating non-context-free string languages (see [HabKre1; Hab; EngHey1], where the string-generating power of cfhg's is studied).

A more important use of (hyper)graphs is to represent expressions (trees, terms), with sharing allowed. Thus, a graph grammar can be used as a term-generating device (or tree grammar), generating a set of terms. Moreover, allowing an additional phase of expression evaluation, a graph grammar generates a set of elements of some semantic domain. For instance, as shown in [Cou3], cfhg's can generate the expression graphs that represent the computations of a recursive program scheme or a recursive query.

In this paper, we investigate the power of cfhg's to generate expressions. A term can be represented by a directed hypergraph in a natural way (see [Cou3; HabKre2; Hab; HabKrePlu; HofPlu]), because the order of its subterms can easily be expressed using the direction of the edges of the hypergraph: every edge has a sequence of incident nodes. We consider "jungles", i.e., hyper-

graphs that represent terms (see [HabKrePlu]), and examine cfhg's that generate jungles. In particular, we compare cfhg's to attribute grammars (AG's, [Knu; DerJouLor]). An AG can be viewed as a device that translates strings into terms (i.e., expressions, to be evaluated in the semantic domain of the AG). The range of this translation is a set of terms, generated by the AG. Our main result is that cfhg's have the same term-generating power as attribute grammars. The class of term (or tree) languages generated by attribute grammars has been studied, e.g., in [Bar; CouFra; DusParSedSpe; Eng1/2/3; EngFil; Fül]. As an example, it is known to contain the IO context-free tree languages [DusParSedSpe; EngFil] and the output languages of deterministic top-down tree transducers [CouFra; Fül], whereas it is itself contained in the class of output languages of macro tree transducers [CouFra; Eng1; EngVog1] and in the complexity class LOG(CF) [Eng3].

We even push the relationship between cfhg's and AG's a little further. By coupling an ordinary context-free grammar (cfg) to a cfhg, a syntax-directed translation device is obtained that translates the strings in the language generated by the cfg into terms, and hence into values from any semantic domain. We extend our main result by showing that such a cfhg-based translation device has the same power as the attribute grammar. In some sense, this result "explains" AG's to be syntax-directed string-to-graph translators (cf. the "DAG-evaluator" of [Mad]). Furthermore, it shows that cfhg's can be used to describe the semantics of programming languages (cf. the "push-down processor" of [AhoUll]).

The relationship between graph grammars and AG's was first explicitly observed by Ganzinger [Gan], Hoffmann [HofSch, Hof], and Courcelle [Cou1, Cou3]. It follows from [Cou1, Sect. 16.8] and [BauCou, Theorem 4.14] that the set of dependency graphs of an AG can be generated by a cfhg. A similar relationship between AG's and NLC-like graph grammars was established in [EngLeiRoz]. However, it seems to us that cfhg's are more suitable to model AG's than NLC-like grammars, because attributes can naturally be coupled to nonterminals by viewing a nonterminal as a (hyper)edge with the attributes as its incident nodes. In fact, in this paper, we use cfhg's not only to simulate the dependencies between attributes but even to simulate (formal) attribute evaluation, in a natural way. In [Hof] a more powerful (context-sensitive) type of graph grammar is used to simulate both parsing and evaluation (together with context-conditions, which we do not allow in our AG's).

The paper is organized as follows. Section 2 contains preliminaries. The reader is assumed to be familiar with AG's, but hypergraphs and cfhg's are described more comprehensively. Furthermore, some technical results concerning AG's and cfhg's are stated. In Sect. 3, jungles are defined and it is shown to be decidable whether a cfhg is term-generating, i.e., generates jungles only (Theorem 3.1). In fact, four alternative definitions of the class of term languages generated by cfhg's are given, and shown to be equivalent (Theorems 3.1 and 5.3). The proof of the main result of this paper (Theorem 5.2, which states that the term-generating power of cfhg's and AG's is the same) is divided into two parts. In Sect. 4, an attribute grammar is simulated by a cfhg, and Sect. 5 contains the simulation of a cfhg by an AG. The paper ends by introducing cfhg-based syntax-directed translations, and proving that these are exactly the translations realized by AG's (Theorem 6.3).

An extended abstract of this paper appears in [EngHey2].

2. Preliminaries

2.1. General notation and terminology

$\mathbb{N}=\{0, 1, 2, \ldots\}$. For all $n, m\in\mathbb{N}$, $[n, m]$ denotes the set $\{i\in\mathbb{N}\mid n\leqq i\leqq m\}$. In particular, if $n>m$ then $[n, m]=\emptyset$.

For a set A, A^* denotes the set of all finite sequences of elements of A, including the empty sequence ( ). In case A is an alphabet, a sequence $(a_1, a_2, \ldots, a_n)\in A^*$ will also be written as a string $a_1\, a_2 \ldots a_n$ (as λ if $n=0$).

By $\#A$ we denote the cardinality of a set A or the length of a sequence A.

A *context-free grammar* (abbreviated cfg) is a 4-tuple $G=(N, T, P, S)$ where N is the nonterminal alphabet, T is the terminal alphabet (disjoint with N), P is the finite set of productions of the form $p=X_0\to w_0 X_1 w_1 X_2 w_2 \ldots X_n w_n$, $n\geqq 0$, with $X_j\in N$ and $w_j\in T^*$ for all $j\in[0, n]$, and $S\in N$ is the initial nonterminal.

For a production $p=X_0\to w_0 X_1 w_1 X_2 w_2 \ldots X_n w_n$ in P, its left-hand side X_0 is denoted as $\mathrm{lhs}(p)$ and its right-hand side $w_0 X_1 w_1 X_2 w_2 \ldots X_n w_n$ is denoted as $\mathrm{rhs}(p)$.

A *derivation subtree* t of G is a directed ordered tree, in which each node is labeled by a production of P, defined inductively as follows. Simultaneously we define the *yield* of t, denoted $\mathrm{yield}_G(t)$ or just $\mathrm{yield}(t)$, and the *root* of t, denoted $\mathrm{root}(t)$.

(1) t is a node x labeled by a production p with $\mathrm{rhs}(p)\in T^*$. In this case $\mathrm{yield}(t)=\mathrm{rhs}(p)$, and $\mathrm{root}(t)=x$.
(2) t consists of a node x with label $p=X_0\to w_0 X_1 w_1 X_2 w_2 \ldots X_n w_n$ $(n\geqq 1)$, n derivation subtrees $t_1, t_2, \ldots, t_n$ such that $\mathrm{root}(t_j)$ is labeled by a production $p_j\in P$ with $\mathrm{lhs}(p_j)=X_j$ for all $j\in[1, n]$, and n edges from x to the roots of the derivation subtrees $t_1, t_2, \ldots, t_n$, in that order. In this case $\mathrm{yield}(t)$ is the string $w_0\, \mathrm{yield}(t_1)\, w_1\, \mathrm{yield}(t_2)\, w_2 \ldots \mathrm{yield}(t_n)\, w_n$ in T^*, and $\mathrm{root}(t)=x$.

A *derivation tree* of G is a derivation subtree of G of which the root is labeled by a production $p\in P$ with $\mathrm{lhs}(p)=S$.

The *(string) language generated by* G, denoted $L(G)$, is the set $\{\mathrm{yield}(t)\mid t$ is a derivation tree of $G\}$.

G is *reduced* if every $X\in N$ occurs in the label of some node in at least one derivation tree of G.

A *ranked alphabet* Γ is a finite set of (function) symbols together with a mapping $\mathrm{rank}_\Gamma\colon \Gamma\to\mathbb{N}$ (which specifies the number of arguments of each function symbol of Γ). We denote the ranked alphabet $\Gamma'=\{\gamma\in\Gamma\mid \mathrm{rank}_\Gamma(\gamma)\geqq 1\}$ with $\mathrm{rank}_{\Gamma'}(\gamma)=\mathrm{rank}_\Gamma(\gamma)-1$, as $\mathrm{dec}(\Gamma)$. The ranked alphabet Γ' consisting of all symbols of Γ, with $\mathrm{rank}_{\Gamma'}(\gamma)=\mathrm{rank}_\Gamma(\gamma)+1$ for all $\gamma\in\Gamma$, is denoted as $\mathrm{inc}(\Gamma)$.

The set of all *terms* written with a ranked alphabet Γ, commas, and parentheses, in prefix notation, is denoted $T(\Gamma)$. For example, if $\Gamma=\{f, g, +, *, 2, 5\}$ with $\mathrm{rank}_\Gamma(g)=3$, $\mathrm{rank}_\Gamma(+)=\mathrm{rank}_\Gamma(*)=2$, $\mathrm{rank}_\Gamma(f)=1$, and $\mathrm{rank}_\Gamma(2)=\mathrm{rank}_\Gamma(5)=0$, then 2, $f(5)$, $+(f(5), 2)$, and $g(2, +(f(5), 2), *(2, 5))$ are terms in $T(\Gamma)$. A *term language* is a subset of $T(\Gamma)$.

Let $Y=(y_1, y_2, \ldots, y_k)$ be a finite sequence of $k\geqq 0$ distinct "variables" such that $\{y_1, y_2, \ldots, y_k\}\cap\Gamma=\emptyset$. By $T(\Gamma, Y)$, we denote the set of terms $T(\Gamma')$ where $\Gamma'=\Gamma\cup\{y_1, y_2, \ldots, y_k\}$, $\mathrm{rank}_{\Gamma'}(\gamma)=\mathrm{rank}_\Gamma(\gamma)$ for all $\gamma\in\Gamma$, and $\mathrm{rank}_{\Gamma'}(y)=0$ for all

$y \in \{y_1, y_2, \ldots, y_k\}$. The terms in $T(\Gamma, Y)$ are also said to be *terms with variables*. Note that, for technical reasons, we consider a sequence, rather than the usual set, of variables.

2.2. *Attribute grammars*

We assume the reader to be familiar with attribute grammars (see, e.g., [Knu; Boc; AhoSetUll; Lor; DerJouLor]).

A *semantic domain* is a pair $D=(V, \Gamma)$ where V is a set of values, Γ is a ranked alphabet, and each $\gamma \in \Gamma$ denotes a mapping $\gamma_D: V^n \to V$ with $n = \mathrm{rank}_\Gamma(\gamma)$. Thus, D is a Γ-algebra where Γ is a one-sorted signature (see, e.g., [CouFra; ChiMar]). For a term $t \in T(\Gamma)$, we denote by t_D the value of t in V, as usual.

An *attribute grammar* G (abbreviated AG) consists of (1)–(4) as follows.

(1) A context-free grammar $G_0=(N_0, T_0, P_0, S_0)$, which is called the *underlying grammar* of G.
(2) A semantic domain $D=(V, \Gamma)$.
(3) An *attribute description* (A, Syn, Inh, Att) where A is a finite set of attributes, and Syn, Inh, and Att are mappings from N_0 to 2^A. For each nonterminal $X \in N_0$, $\mathrm{Syn}(X)$ is the set of its *synthesized attributes*, and $\mathrm{Inh}(X)$ is the set of its *inherited attributes*, $\mathrm{Syn}(X) \cap \mathrm{Inh}(X) = \emptyset$. $\mathrm{Att}(X)$ is the set of all its attributes, $\mathrm{Att}(X) = \mathrm{Syn}(X) \cup \mathrm{Inh}(X)$. The set $\mathrm{Inh}(S_0)$ is empty, and $\#\mathrm{Syn}(S_0) = 1$. The only (synthesized) attribute of S_0 is called the *designated attribute* of G and is denoted α_d.
(4) For each production $p = X_0 \to w_0 X_1 w_1 X_2 w_2 \ldots X_n w_n$ a set of *semantic rules* r_p. For each $\langle \alpha, j \rangle \in \mathrm{ins}(p)$, r_p contains one semantic rule of the form $\langle \alpha, j \rangle = t$ with $t \in T(\Gamma, \mathrm{outs}(p))$, where $\mathrm{ins}(p) = \{\langle \beta, i \rangle \mid (\beta \in \mathrm{Inh}(X_i)$ and $i \in [1, n])$ or $(\beta \in \mathrm{Syn}(X_i)$ and $i = 0)\}$ is the set of *inside attributes* of p and $\mathrm{outs}(p) = \{\langle \beta, i \rangle \mid (\beta \in \mathrm{Syn}(X_i)$ and $i \in [1, n])$ or $(\beta \in \mathrm{Inh}(X_i)$ and $i = 0)\}$ is the set of *outside attributes* of p. The set $\mathrm{outs}(p)$ is given some fixed but arbitrary order, so that it can be considered as a sequence, whenever this is convenient (in particular in $T(\Gamma, \mathrm{outs}(p))$).

A semantic rule of the form $\langle \alpha, j \rangle = \langle \beta, i \rangle$ is called a *copy rule*.

Note that we only consider attribute grammars which are in Bochmann normal form, see [Boc]. Note also that we do not allow semantic (context-) conditions.

If we want to emphasize the semantic domain D of an AG G, we say that G is an AG *over* D.

If $\langle \alpha, j \rangle = t$ is a semantic rule in r_p, for some production p of the underlying grammar $G_0=(N_0, T_0, P_0, S_0)$ of an AG G, and $\langle \beta, i \rangle$ occurs in t, then we say that $\langle \alpha, j \rangle$ *depends on* $\langle \beta, i \rangle$. The *dependency graph* of p, denoted DG_p, is the directed graph with nodes $\mathrm{ins}(p) \cup \mathrm{outs}(p)$, and with an edge from $\langle \beta, i \rangle$ to $\langle \alpha, j \rangle$ iff $\langle \alpha, j \rangle$ depends on $\langle \beta, i \rangle$. The dependency graph of a derivation subtree t of G_0, DG_t, is obtained by gluing together the dependency graphs of the productions that are labels in t (in agreement with t). More precisely, if a node x of t is labeled p and its j-th son y is labeled p', then the nodes $\langle \alpha, j \rangle$ of DG_p and $\langle \alpha, 0 \rangle$ of $DG_{p'}$ are glued together in DG_t. We shall refer to this node in DG_t as $\langle \alpha, y \rangle$. In this way, we view every attribute $\alpha \in \mathrm{Att}(\mathrm{lhs}(p'))$ as

an attribute $\langle \alpha, y\rangle$ of y. Hence, in the semantic rules of r_p, $\langle \alpha, 0\rangle$ refers to attribute $\langle \alpha, x\rangle$ of node x, and $\langle \alpha, j\rangle$, with $j \geqq 1$, refers to attribute $\langle \alpha, x_j\rangle$ of the j-th son x_j of x.

An AG G is *reduced* if for every derivation tree t of its underlying grammar G_0 there is a directed path in DG_t from every node $\langle \alpha, x\rangle$ to the node $\langle \alpha_d, \mathrm{root}(t)\rangle$ (see [Fil]). The AG G is *circular* if there exists a derivation tree t of G_0 such that DG_t contains a directed cycle, otherwise G is *non-circular*. In this paper, we shall consider non-circular AG's only.

The evaluation of the attributes $\langle \alpha, x\rangle$ of the nodes of a derivation tree t of the underlying grammar G_0 of a non-circular AG G over $D=(V, \Gamma)$, is as usual. A semantic rule $\langle \alpha, j\rangle = t$ where $t=\gamma(t_1, t_2, \ldots, t_n)$, expresses that the value of the attribute $\langle \alpha, x\rangle$ referred to by $\langle \alpha, j\rangle$ is computed by applying the function γ_D to the values of $t_1, t_2, \ldots, t_n$. This is called the evaluation of the semantic rule $\langle \alpha, j\rangle = t$ (for $\langle \alpha, x\rangle$). Thus, each attribute $\langle \alpha, x\rangle$ of a node x in t gets a unique value in V, denoted $\mathrm{val}_G(\langle \alpha, x\rangle, t)$ or just $\mathrm{val}(\langle \alpha, x\rangle, t)$. Notice that in case G is a reduced non-circular AG, every attribute $\langle \alpha, x\rangle$ of t is used in the computation of the value of $\langle \alpha_d, \mathrm{root}(t)\rangle$.

The *translation realized by* a non-circular AG G with underlying grammar G_0, is $\tau(G)=\{(\mathrm{yield}(t), \mathrm{val}_G(\langle \alpha_d, \mathrm{root}(t)\rangle, t)) \mid t \text{ is a derivation tree of } G_0\}$. By $\tau(\mathrm{AG}, D)$ we denote the class of all translations realized by AG's over semantic domain D. The *output language realized by* G, denoted $\mathrm{OUT}(G)$, is the range of $\tau(G)$, i.e., $\mathrm{OUT}(G)=\{\mathrm{val}_G(\langle \alpha_d, \mathrm{root}(t)\rangle, t) \mid t \text{ is a derivation tree of } G_0\}$.

Turning G into a reduced AG can be accomplished without changing the translation realized by G (Theorem 4.1 of [Fil]). Remark that in that case the output language realized by G does not change either.

Proposition 2.1 *For every non-circular* AG G *over the semantic domain* D *there is a reduced non-circular* AG G' *over* D *such that* $\tau(G')=\tau(G)$. □

An AG G over a semantic domain $D=(V, \Gamma)$ is *term-generating* if D is the free Γ-algebra, i.e., $V=T(\Gamma)$ and $\gamma_D(t_1, t_2, \ldots, t_n)=\gamma(t_1, t_2, \ldots, t_n)$ for all $\gamma \in \Gamma$ (see, e.g., [CouFra; EngFil; Fül]). Thus, for such an AG, $\mathrm{OUT}(G) \subseteq T(\Gamma)$ is a term language. The class of all output languages realized by term-generating AG's is denoted as OUT(AG, TERMS).

Notice that, in our approach, every AG G is term-generating, in the sense that G determines a term-generating AG G_{term}: just change the semantic domain $D=(V, \Gamma)$ into $D_{\mathrm{term}}=(T(\Gamma), \Gamma)$. It should be clear that the following "Mezei-Wright-like" result holds (see [MezWri]): for every derivation tree t of the underlying grammar G_0 of G and G_{term}, $\mathrm{val}_G(\langle \alpha_d, \mathrm{root}(t)\rangle, t) = (\mathrm{val}_{G_{\mathrm{term}}}(\langle \alpha_d, \mathrm{root}(t)\rangle, t))_D$. Thus, to compute the value of $\langle \alpha_d, \mathrm{root}(t)\rangle$ in V, one may first evaluate the attributes formally, i.e., as terms in $T(\Gamma)$, and then evaluate the term-value of $\langle \alpha_d, \mathrm{root}(t)\rangle$ in D.

Example 2.2 Consider the AG G_{bin} defined in (1.5) in [Knu], which assigns a "meaning", i.e., a rational number, to every binary number. In our notation this AG is defined as follows.

(1) $G_0=(N_0, T_0, P_0, S_0)$ where $N_0=\{N, L, B\}$, $T_0=\{0, 1, \cdot\}$, $S_0=N$, and $P_0=\{p_1, p_2, \ldots, p_6\}$ with $p_1=N \to L \cdot L$, $p_2=N \to L$, $p_3=L \to LB$, $p_4=L \to B$, $p_5=B \to 1$, and $p_6=B \to 0$.

(2) $D=(V, \Gamma)$ where V is the set of rational numbers and $\Gamma=\{+, 2\uparrow, -, 0, 1\}$ with $\mathrm{rank}_\Gamma(+)=2$, $\mathrm{rank}_\Gamma(2\uparrow)=\mathrm{rank}_\Gamma(-)=1$, and $\mathrm{rank}_\Gamma(0)=\mathrm{rank}_\Gamma(1)=0$. For every $\gamma \in \Gamma$, the function γ_D is as expected.

(3) $A=(s, \ell, v\}$, where v represents the "value" of the nonterminals N, L, and B, ℓ represents the "length" of a list of bits L, and s represents the "scale" of a bit B, or the last bit in a list L. $\mathrm{Inh}(N)=\emptyset$, $\mathrm{Syn}(N)=\{v\}$, $\mathrm{Inh}(L)=\{s\}$, $\mathrm{Syn}(L)=\{\ell, v\}$, $\mathrm{Inh}(B)=\{s\}$, and $\mathrm{Syn}(B)=\{v\}$. The designated attribute of G_{bin} is $\alpha_d=v$.
(4) For the production $p_1=N\to L\cdot L$, $\mathrm{ins}(p_1)=\{\langle v,0\rangle, \langle s,1\rangle, \langle s,2\rangle\}$, $\mathrm{outs}(p_1)=\{\langle \ell,1\rangle, \langle v,1\rangle, \langle \ell,2\rangle, \langle v,2\rangle\}$, and its semantic rules are $\langle v,0\rangle = +(\langle v,1\rangle, \langle v,2\rangle)$, $\langle s,1\rangle=0$, and $\langle s,2\rangle = -(\langle \ell,2\rangle)$. The other productions have the following semantic rules.

$p_2=N\to L$: $\langle v,0\rangle=\langle v,1\rangle$, $\langle s,1\rangle=0$,
$p_3=L\to LB$: $\langle \ell,0\rangle = +(\langle \ell,1\rangle, 1)$, $\langle s,1\rangle = +(\langle s,0\rangle, 1)$,
$\langle v,0\rangle = +(\langle v,1\rangle, \langle v,2\rangle)$, $\langle s,2\rangle=\langle s,0\rangle$,
$p_4=L\to B$: $\langle \ell,0\rangle=1$, $\langle s,1\rangle=\langle s,0\rangle$,
$\langle v,0\rangle=\langle v,1\rangle$,
$p_5=B\to 1$: $\langle v,0\rangle=2\uparrow(\langle s,0\rangle)$, and
$p_6=B\to 0$: $\langle v,0\rangle=0$. □

2.3. *Context-free hypergraph grammars*

A directed hypergraph consists of a set of nodes and a set of (hyper)edges, just as an ordinary graph except that an edge may be incident with any number of nodes rather than two. The edges are directed in the sense that with every edge a sequence of nodes is associated (possibly with repetitions). Moreover, in our hypergraphs, each edge is labeled with a symbol from a ranked alphabet, in such a way that the rank of its label equals the length of its sequence of incident nodes. Finally, we assume that every hypergraph has a sequence of designated nodes, called external nodes. Formally (cf. [BauCou; MonRos]), a hypergraph is defined as follows.

Definition. Let Σ be a ranked alphabet. A *hypergraph over* Σ is a tuple $H=(V, E, \mathrm{nod}, \mathrm{lab}, \mathrm{ext})$, where V is the finite set of nodes, E is the finite set of hyperedges (or edges), $\mathrm{nod}: E\to V^*$ is the incidence function, $\mathrm{lab}: E\to\Sigma$ is the labeling function, and $\mathrm{ext}\in V^*$ is the sequence of external nodes. It is required that for every $e\in E$, $\mathrm{rank}_\Sigma(\mathrm{lab}(e))=\#\mathrm{nod}(e)$. □

If $\mathrm{nod}(e)=(v_1, v_2, \ldots, v_r)$, $r\in\mathbb{N}$, then r is said to be the *rank* of e, denoted $\mathrm{rank}(e)$, and e is called an *r-hyperedge* (thus, $\mathrm{rank}(e)=\mathrm{rank}_\Sigma(\mathrm{lab}(e))$). The node v_i is also denoted by $\mathrm{nod}(e, i)$, and we say that e and v_i are *incident*. Similarly, if $\mathrm{ext}=(v_1, v_2, \ldots, v_m)$, $m\in\mathbb{N}$, then v_i is denoted by $\mathrm{ext}(i)$. Moreover, $m=\#\mathrm{ext}$ is the *rank* of H, denoted $\mathrm{rank}(H)$, and H will be called an *m-hypergraph*. A non-external node of H is also called an *internal node*.

If the alphabet is irrelevant or clear from the context, a hypergraph over a ranked alphabet Σ will shortly be called a hypergraph.

For a given hypergraph H, its components are denoted by V_H, E_H, nod_H, lab_H, and ext_H, respectively.

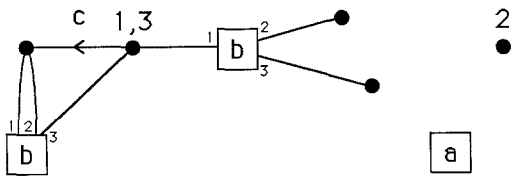

Fig. 1

For a ranked alphabet Σ, the set of all hypergraphs over Σ is denoted by $\mathrm{HGR}(\Sigma)$, and the set of all m-hypergraphs, $m \in \mathbb{N}$, over Σ is denoted by $\mathrm{HGR}_m(\Sigma)$. A *hypergraph language* is a subset of $\mathrm{HGR}(\Sigma)$, for some ranked alphabet Σ.

For hypergraphs $H, K \in \mathrm{HGR}(\Sigma)$, K is a *subgraph of* H if $V_K \subseteq V_H$, $E_K \subseteq E_H$, and nod_K and lab_K are the restrictions of nod_H and lab_H to E_K. Note that, as in [Hab], but in contrast to the subhypergraph defined in [HabKre1], we do not require that $\mathrm{ext}_K = \mathrm{ext}_H$.

We assume the reader to be experienced in the problem of concrete vs. abstract graphs (where an abstract graph is a class of isomorphic concrete graphs). As usual in the theory of graph grammars we consider hypergraph languages to consist of abstract hypergraphs, but in all our constructions we deal with concrete hypergraphs (taking an isomorphic copy when necessary). The notion of isomorphism is the obvious one, preserving the incidence structure, the edge labels, and the sequence of external nodes.

Example. Consider the ranked alphabet $\Sigma = \{a, b, c\}$ with $\mathrm{rank}_\Sigma(a) = 0$, $\mathrm{rank}_\Sigma(b) = 3$, and $\mathrm{rank}_\Sigma(c) = 2$. Figure 1 contains a picture of a 3-hypergraph $H = (V, E, \mathrm{nod}, \mathrm{lab}, \mathrm{ext})$ over Σ. A node of H is indicated by a fat dot, as usual, and an edge of H is indicated by a box containing $\mathrm{lab}(e)$, with a line between e and $\mathrm{nod}(e, i)$ labeled by i. These lines (or the corresponding integers) are called the "tentacles" of the hyperedge e (see [HabKre1]). A 2-hyperedge e is also drawn as a directed line from $\mathrm{nod}(e, 1)$ to $\mathrm{nod}(e, 2)$, with label $\mathrm{lab}(e)$, as usual in ordinary graphs. The external node $\mathrm{ext}(i)$ is indicated by a label i. In Fig. 1, H has nodes v_1, v_2, v_3, v_4, v_5, and edges e_1, e_2, e_3, e_4 (both enumerated from left to right), and it satisfies $\mathrm{nod}(e_1) = (v_1, v_1, v_2)$, $\mathrm{nod}(e_2) = (v_2, v_1)$, $\mathrm{nod}(e_3) = (v_2, v_3, v_4)$, $\mathrm{nod}(e_4) = ( )$, $\mathrm{lab}(e_1) = b$, $\mathrm{lab}(e_2) = c$, $\mathrm{lab}(e_3) = b$, $\mathrm{lab}(e_4) = a$, and $\mathrm{ext} = (v_2, v_5, v_2)$. Notice that Fig. 1 is also a picture of all hypergraphs isomorphic with H (i.e., it is a picture of an abstract hypergraph). □

We now turn to context-free hypergraph grammars (cfhg's), see, e.g., [Fed; BauCou; Cou3 and 4; HabKre1 and 2; Hab; Lau; EngHey1; EngRoz; MonRos; LenWan]. A cfhg is similar to an ordinary context-free grammar, but (labeled) edges of hypergraphs are rewritten rather than symbols of strings. Thus, a production of a cfhg is of the form (X, H) where X is a (nonterminal) edge label and H is a hypergraph of the same rank as X. The application of this production to a hyperedge e (with label X) of a sentential form K of the grammar consists of substituting H for e in K, identifying $\mathrm{nod}_K(e, i)$ with $\mathrm{ext}_H(i)$ for all $i \in [1, \mathrm{rank}(X)]$. To define this formally, it is convenient to use the following two operations on hypergraphs: identification and substitution.

(1) Identification of nodes. Let $H \in \mathrm{HGR}(\Sigma)$ and $R \subseteq V_H \times V_H$. Intuitively, we want to identify nodes v and v', for every pair $(v, v') \in R$. Let $\equiv_R$ denote the

smallest equivalence relation on V_H containing R. For $v \in V_H$, let $[v]_R$ denote the equivalence class of v with respect to $\equiv_R$, and let $V_H/\equiv_R = \{[v]_R \mid v \in V_H\}$. Then we define the hypergraph $H/R = (V_H/\equiv_R, E_H, \text{nod}, \text{lab}_H, \text{ext})$ such that if $\text{ext}_H = (v_1, v_2, \ldots, v_m)$, then $\text{ext} = ([v_1]_R, [v_2]_R, \ldots, [v_m]_R)$, and, for every edge $e \in E_H$, if $\text{nod}_H(e) = (v_1, v_2, \ldots, v_r)$, then $\text{nod}(e) = ([v_1]_R, [v_2]_R, \ldots, [v_r]_R)$.

(2) Substitution. Let $K, H \in \text{HGR}(\Sigma)$, and let $e \in E_K$ such that $\text{rank}(e) = \text{rank}(H)$. Then the hypergraph $K[e/H]$, the result of substituting H for e in K, is defined as follows. We assume that $V_K \cap V_H = \emptyset$ (otherwise an isomorphic copy of H should be taken). Let K' be the result of removing e from K and adding H (disjointly), i.e., $K' = (V, E, \text{nod}, \text{lab}, \text{ext})$ where $V = V_K \cup V_H$, $E = (E_K - \{e\}) \cup E_H$, $\text{nod} = \text{nod}_K \cup \text{nod}_H$, restricted to E, $\text{lab} = \text{lab}_K \cup \text{lab}_H$, restricted to E, and $\text{ext} = \text{ext}_K$. Then $K[e/H] = K'/R$, where $R = \{(\text{nod}_K(e, i), \text{ext}_H(i)) \mid i \in [1, \text{rank}(H)]\}$.

Finally, to define the start of a derivation of a cfhg, we also need a notation for a hypergraph consisting of a single edge, with the appropriate number of nodes. For $\sigma \in \Sigma$, with $\text{rank}_\Sigma(\sigma) = m$, the hypergraph $([1, m], \{e\}, \text{nod}, \text{lab}, \text{ext})$ with $\text{nod}(e) = \text{ext} = (1, 2, \ldots, m)$, and $\text{lab}(e) = \sigma$ will be denoted as σ. It will be clear from the context whether the label σ or the hypergraph σ is meant by σ.

We are now prepared for the definition of context-free hypergraph grammars (cf. [BauCou]).

Definition. A *context-free hypergraph grammar* (abbreviated cfhg) is a tuple $G = (\Sigma, \Delta, P, S)$, where Σ is a ranked alphabet, $\Delta \subseteq \Sigma$ is the terminal alphabet (and $\Sigma - \Delta$ is the nonterminal alphabet), P is the finite set of productions, and $S \in \Sigma - \Delta$ is the initial nonterminal. Every production in P is of the form (X, H) with $X \in \Sigma - \Delta$, $H \in \text{HGR}(\Sigma)$, and $\text{rank}_\Sigma(X) = \text{rank}(H)$. □

For a production $\pi = (X, H)$, X is the left-hand side and H is the right-hand side of π, denoted $\text{lhs}(\pi)$ and $\text{rhs}(\pi)$, respectively.

For a hyperedge e of a hypergraph $H \in \text{HGR}(\Sigma)$, e is called a *terminal edge* if $\text{lab}_H(e) \in \Delta$, and a *nonterminal edge* otherwise. H is said to be terminal if all its edges are terminal. We denote the set of all terminal (nonterminal) edges of H by $\text{tedg}(H)$ ($\text{nedg}(H)$, respectively). Thus, E_H is partitioned into $\text{tedg}(H)$ and $\text{nedg}(H)$. Whenever technically convenient, we assume that $\text{tedg}(H)$ and $\text{nedg}(H)$ are given some fixed but arbitrary order, of which the j-th element is denoted $\text{tedg}(H, j)$ and $\text{nedg}(H, j)$, respectively. The *terminal part of* H is the hypergraph $(V_H, \text{tedg}(H), \text{nod}, \text{lab}, \text{ext}_H)$, where nod and lab are nod_H and lab_H restricted to $\text{tedg}(H)$. Note that the terminal part of H is a subgraph of H.

Let $G = (\Sigma, \Delta, P, S)$ be cfhg. Formally, application of a production $\pi = (X, H)$ of G is defined as follows. Let $K \in \text{HGR}(\Sigma)$, and let $e \in \text{nedg}(K)$. Then π is applicable to e if $\text{lab}_K(e) = X$, and the result of the application is the hypergraph $K[e/H]$. We write $K \Rightarrow_{(e,\pi)} K'$, or just $K \Rightarrow K'$, if K' is the result of applying π to e of K, i.e., if K' is (isomorphic to) $K[e/H]$. As usual, $\Rightarrow^*$ denotes the transitive reflexive closure of $\Rightarrow$.

Definition. Let $G = (\Sigma, \Delta, P, S)$ be a cfhg. The *(hypergraph) language generated by* G, denoted $L(G)$, is the set $\{H \in \text{HGR}(\Delta) \mid S \Rightarrow^* H\}$. A hypergraph $H \in \text{HGR}(\Sigma)$ is a *sentential form of* G if $S \Rightarrow^* H$. □

Notice that $\text{rank}(H) = \text{rank}_\Sigma(S)$ for every sentential form of G.

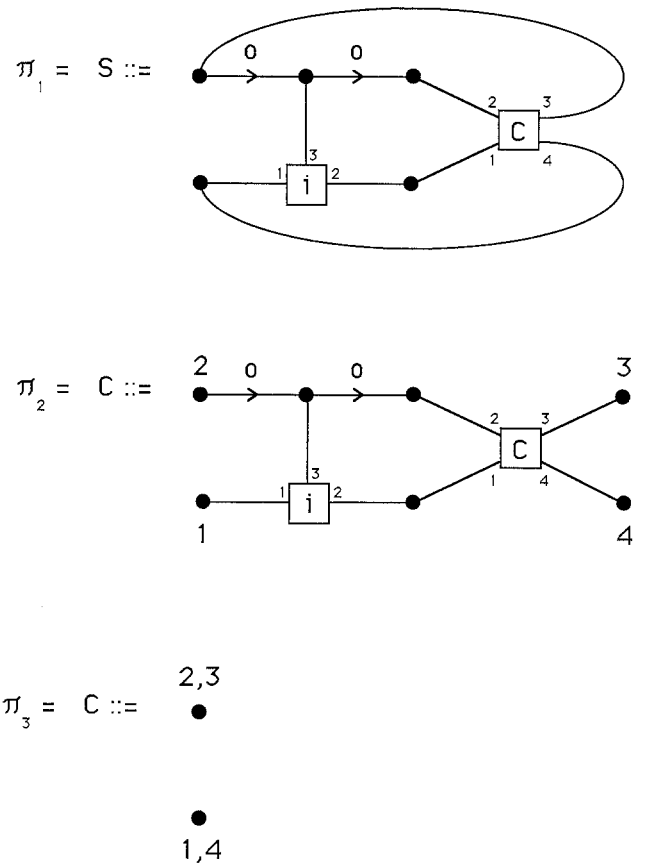

Fig. 2

A cfhg $G=(\Sigma, \Delta, P, S)$ is *reduced* if every $X \in \Sigma - \Delta$ occurs in at least one derivation $S \Rightarrow^* F$ with $F \in L(G)$. It should be clear that any cfhg G (with $L(G) \neq \emptyset$) can be turned into an equivalent reduced cfhg G' by dropping its useless nonterminals (and productions).

Example. Consider the cfhg $G=(\Sigma, \Delta, P, S)$ where $\Sigma=\{S, C, i, o\}$ with $\text{rank}_\Sigma(S)=0$, $\text{rank}_\Sigma(C)=4$, $\text{rank}_\Sigma(i)=3$, and $\text{rank}_\Sigma(o)=2$, $\Delta=\{i, o\}$, and $P=\{\pi_1, \pi_2, \pi_3\}$ as given in Fig. 2, with each production (X, H) written as $X::=H$. This cfhg generates all "double circles" of the form given in Fig. 3. Notice that $L(G) \subseteq \text{HGR}_0(\Sigma)$, because $\text{rank}_\Sigma(S)=0$. □

Derivation trees of cfhg's and the terminal hypergraphs they yield are defined as follows (see, e.g., [Lau]). The definition depends on the (fixed, but arbitrary) order on $\text{nedg}(\text{rhs}(\pi))$, for every production π.

Definition. Let $G=(\Sigma, \Delta, P, S)$ be a cfhg. A *derivation subtree* of G is a directed ordered tree t, in which each node is labeled by a production of P, defined inductively as follows. Simultaneously we define the *yield* of t, denoted $\text{yield}_G(t)$ or just $\text{yield}(t)$, and the *root* of t, denoted $\text{root}(t)$.

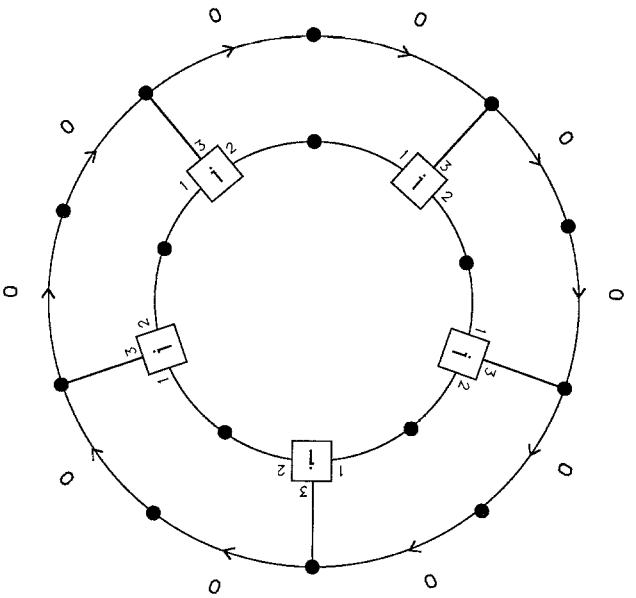

Fig. 3

(1) t is a node x labeled by a production π with $\#\text{nedg}(\text{rhs}(\pi))=0$. In this case $\text{yield}(t)$ is the terminal hypergraph $\text{rhs}(\pi)$, and $\text{root}(t)=x$.
(2) t consists of a node x with label $\pi=(X, H)$, $m=\#\text{nedg}(H)$ derivation subtrees $t_1, t_2, \ldots, t_m$ $(m\geqq 1)$ such that $\text{root}(t_j)$ is labeled by a production $\pi_j\in P$ with $\text{lhs}(\pi_j)=\text{lab}_H(\text{nedg}(H, j))$ for all $j\in[1, m]$, and m edges from x to the roots of the derivation subtrees $t_1, t_2, \ldots, t_m$, in that order. In this case $\text{yield}(t)$ is the terminal hypergraph

$$H[\text{nedg}(H, 1)/\text{yield}(t_1)][\text{nedg}(H, 2)/\text{yield}(t_2)] \ldots [\text{nedg}(H, m)/\text{yield}(t_m)],$$

and $\text{root}(t)=x$.

A *derivation tree* of G is a derivation subtree of G of which the root is labeled by a production $\pi\in P$ with $\text{lhs}(\pi)=S$. □

It can be shown in a straightforward way that, as in the case of ordinary context-free grammars, a hypergraph can be generated by a cfhg G if and only if it is the yield of a derivation tree of G (cf. Theorems 4.5 and 4.8 of [Kre], and Example 2.20 of [Cou2]). More strongly, for every $X\in\Sigma-\Delta$, $\{H\in\text{HGR}(\Delta)\mid X\Rightarrow^* H\}=\{\text{yield}(t)\mid t \text{ is a derivation subtree of } G, \text{ and } \text{root}(t)$ is labeled with some production $\pi\in P$ such that $\text{lhs}(\pi)=X\}$ (cf. Theorem II.3.6 of [Hab]).

Application of a production (X, H) of a cfhg to a sentential form K may result in the identification of some of the nodes of K. This is due to the fact that the sequence of external nodes of H may contain repetitions (cf. Fig. 2). This feature of cfhg's (which is not present in [HabKre1 and 2; Hab]) is useful, e.g., when simulating copy rules in an AG, as we shall see in Sect. 4. A cfhg in which such an identification is not allowed is called identification-free. Thus, for an identification-free cfhg, the terminal part of a sentential form K is always

a subgraph of a terminal hypergraph F generated from K. In other words, the part of F that has already been generated in K, will not change by the remainder of the derivation. This property simplifies visualizing the derivations of a cfhg.

Definition. An m-hypergraph H is *identification-free* if $\mathrm{ext}_H(i) \neq \mathrm{ext}_H(j)$ for all $i, j \in [1, m]$ with $i \neq j$. A cfhg G is *identification-free* if $\mathrm{rhs}(\pi)$ is identification-free for every production π of G. □

Notice that an identification-free cfhg generates identification-free hypergraphs only. In [EngHey1, Lemma 3.2] it is shown that every cfhg can be turned into an identification-free cfhg that generates the same identification-free hypergraphs. This result is similar to the removal of λ-productions from an ordinary context-free grammar.

Proposition 2.3 *For every* cfhg G *there is an identification-free* cfhg G' *such that* $L(G') = \{H \in L(G) \mid H \text{ is identification-free}\}$. □

A consequence of this proposition is that, for every cfhg G of which the initial nonterminal has rank 0 or 1, there exists an identification-free cfhg G' such that $L(G') = L(G)$.

As observed above, in a sentential form K of a cfhg that is not identification-free, application of a production (X, H) to a nonterminal hyperedge e may result in the identification of nodes in $\mathrm{nod}_K(e)$ as a consequence of repetitions in ext_H. The "reverse" may also occur. External nodes of H can be identified as a consequence of repetitions in $\mathrm{nod}_K(e)$, called "loops".

Definition. Let $G = (\Sigma, \Delta, P, S)$ be a cfhg. A hypergraph H over Σ is *loop-free* if for every $e \in \mathrm{nedg}(H)$, $\mathrm{nod}(e, i) \neq \mathrm{nod}(e, j)$ for all i, j in $[1, \mathrm{rank}(e)]$ with $i \neq j$. G is *loop-free* if $\mathrm{rhs}(\pi)$ is loop-free for every production $\pi \in P$. □

A sentential form of a loop-free cfhg is not necessarily loop-free. To see this, consider two nonterminal hyperedges e and e' in a loop-free sentential form K with $\mathrm{nod}_K(e, i) = \mathrm{nod}_K(e', i')$ and $\mathrm{nod}_K(e, j) = \mathrm{nod}_K(e', j')$, for some i, i', j, j'. Then, if $\mathrm{nod}_K(e, i)$ and $\mathrm{nod}_K(e, j)$ are identified as a consequence of applying a loop-free production (X, H) to e (i.e., $\mathrm{ext}_H(i) = \mathrm{ext}_H(j)$), e' has a loop in the sentential form $K[e/H]$. But it is easy to see that, for a loop-free and identification-free cfhg, all sentential forms are loop-free (and identification-free). This implies that, after application of a production π, $\mathrm{rhs}(\pi)$ is a subgraph of the sentential form (and hence the terminal part of $\mathrm{rhs}(\pi)$ will be a subgraph of the generated terminal hypergraph). Thus, every derivation subtree of a derivation tree t yields a subgraph of $\mathrm{yield}(t)$. This simplifies proofs that use induction on the structure of derivation subtrees.

For that reason, we shall use the Well-Formedness Theorem of [Hab, Theorem I.4.6], where well-formed means loop-free. Because in [Hab] the sequence of external nodes of the right-hand side of a cfhg production does not contain repetitions (by definition), this theorem states the existence of a loop-free and identification-free cfhg generating the same hypergraph language as a given identification-free cfhg.

Proposition 2.4 *For every identification-free cfhg* G *there is a loop-free and identification-free* cfhg G' *such that* $L(G') = L(G)$. □

In the proof of Theorem 3.1, we shall use a monadic second order logic (abbreviated MSOL), defined in [Cou4], to express properties of hypergraphs.

A language of MSOL, $\mathscr{L}_{\Delta,m}$ for some ranked alphabet Δ and some $m \in \mathbb{N}$, consists of formulas that express properties of m-hypergraphs over Δ. Such a formula defines a hypergraph language in $\mathrm{HGR}_m(\Delta)$, consisting of all hypergraphs that satisfy the formula.

Formulas are built by using node constants $\mathrm{ext}(i)$, $1 \leqq i \leqq m$, node variables v, w, v', v_1, v_2, ..., edge variables e, e', e_1, e_2, ..., node set variables V, V', ..., and edge set variables E, E', ... For a given hypergraph H, $\mathrm{ext}(i)$ denotes $\mathrm{ext}_H(i)$, the node and edge variables range over all elements of V_H and E_H, respectively, and the node set and edge set variables range over all subsets of V_H and E_H, respectively.

$\mathscr{L}_{\Delta,m}$ contains the atomic formulas (1) $x = x'$, for either node variables (or constants) x and x', or edge variables x and x', (2) $x \in X$, for either node variable (or constant) x and node set variable X, or edge variable x and edge set variable X, and (3) $\mathrm{edg}_a(e, v_1, v_2, \ldots, v_r)$, for $a \in \Delta$, edge variable e and node variables (or constants) $v_1, v_2, \ldots, v_r$, where r abbreviates $\mathrm{rank}_\Delta(a)$.

Intuitively, these formulas express that two nodes (edges) are equal, that a node (edge) is an element of a set of nodes (edges), and that, for hyperedge e, $\mathrm{nod}(e) = (v_1, v_2, \ldots, v_r)$ and $\mathrm{lab}(e) = a$.

The formulas of the language $\mathscr{L}_{\Delta,m}$ of MSOL are constructed from the above atomic formulas through the propositional connectives $\wedge$, $\vee$, $\neg$, $\rightarrow$, and the quantifiers $\forall$ and $\exists$, as usual. A formula is closed if it has no free variables. If φ is a closed formula in $\mathscr{L}_{\Delta,m}$, then an m-hypergraph H over Δ either satisfies the property defined by φ or it does not. We write $H \models \varphi$ if H satisfies φ.

Furthermore, we shall use the abbreviations $\bigwedge$ and $\bigvee$ to denote the conjunction and disjunction of a (finite) set of formulas.

Example. Consider the ranked alphabet $\Delta = \{a, b, c\}$ with $\mathrm{rank}_\Delta(a) = 0$, $\mathrm{rank}_\Delta(b) = 3$, and $\mathrm{rank}_\Delta(c) = 2$. The 3-hypergraph H over Δ, given in Fig. 1, does not satisfy the closed formula φ in $\mathscr{L}_{\Delta,3}$, where φ expresses that no external node is isolated (i.e., for every external node $v = \mathrm{ext}(i)$, $1 \leqq i \leqq 3$, there is a hyperedge e, with an arbitrary label d, that is incident with v). The formula ϕ is defined as follows:

$$\varphi \Leftrightarrow \forall v \bigwedge_{i=1}^{3} : v = \mathrm{ext}(i) \rightarrow \bigvee_{d \in \Delta} \Big(\exists e \, \exists v_1 \, \exists v_2 \ldots \exists v_r : \mathrm{edg}_d(e, v_1, v_2, \ldots, v_r) \wedge \bigvee_{j=1}^{r} (v_j = v) \Big),$$

where r abbreviates $\mathrm{rank}_\Delta(d)$. □

In many constructions concerning cfhg's, the following proposition (Corollary 4.8 of [Cou4]) is useful.

Proposition 2.5 *Let Δ be a ranked alphabet, and $m \in \mathbb{N}$. Let G be a* cfhg *with $L(G) \subseteq \mathrm{HGR}_m(\Delta)$. Let φ be a closed formula in $\mathscr{L}_{\Delta,m}$.*

(1) One can decide whether for all H in $L(G)$ $H \models \varphi$.
(2) One can construct a cfhg *generating $\{H \in L(G) \mid H \models \varphi\}$.*

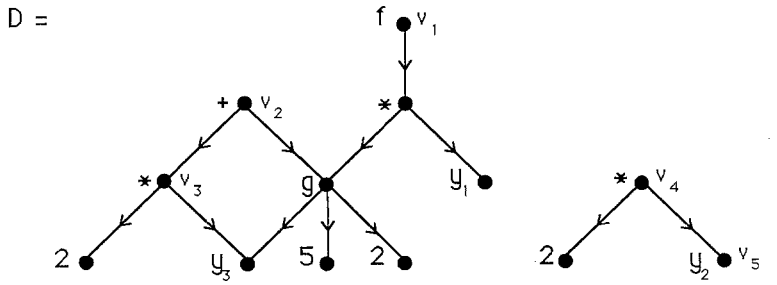

Fig. 4

3. Term generation by context-free hypergraph grammars

A term is often represented by a node in a finite, node-labeled, directed, ordered, acyclic graph (DOAG). As opposed to a representation by trees, this allows sharing of subterms. In this paper we analogously, but more conveniently, represent a term by a node in a hypergraph, as in [Cou3; HabKrePlu]. Although we are interested in the generation of terms without variables (cf. Sect. 2.2), we will also need to represent terms with variables (in particular because the right-hand side of a semantic rule of an AG is a term with variables). Recall from Sect. 2.1 that, for technical reasons, we always consider a sequence rather than a set of variables.

Let Γ be a ranked alphabet and Y a sequence of (distinct) variables. In a DOAG D where a node v represents a term $t=\gamma(t_1, t_2, \ldots, t_n)$ in $T(\Gamma, Y)$, v is labeled γ. Further, D contains n (i.e., $\mathrm{rank}_\Gamma(\gamma)$) ordered edges directed from v to the nodes representing the terms $t_1, t_2, \ldots, t_n$ in $T(\Gamma, Y)$, respectively. In a hypergraph H where a node v represents t, γ is represented by a hyperedge e labeled γ (unless $\gamma \in Y$). The tentacles of e, ordered in the sequence $\mathrm{nod}(e)$, are used to connect the nodes representing $t_1, t_2, \ldots, t_n$ with v, respectively. This means that e is a $(\mathrm{rank}_\Gamma(\gamma)+1)$-hyperedge of H. Variables from Y are represented by external nodes rather than hyperedges (the i-th variable by the i-th external node).

Example. Consider the ranked alphabet $\Gamma=\{f, g, +, *, 2, 5\}$ with $\mathrm{rank}_\Gamma(g)=3$, $\mathrm{rank}_\Gamma(+)=\mathrm{rank}_\Gamma(*)=2$, $\mathrm{rank}_\Gamma(f)=1$, and $\mathrm{rank}_\Gamma(2)=\mathrm{rank}_\Gamma(5)=0$.

In the DOAG D given in Fig. 4, the order of the outgoing edges is from left to right at each node, and the node labels are displayed to the left of the nodes. A corresponding hypergraph H is given in Fig. 5. In both graphs, the nodes represent terms in $T(\Gamma, Y)$, where $Y=(y_1, y_2, y_3)$. For example, the nodes v_1, v_2, v_3, v_4, and v_5 represent $f(*(g(y_3, 5, 2), y_1))$, $+(*(2, y_3), g(y_3, 5, 2))$, $*(2, y_3)$, $*(2, y_2)$, and y_2, respectively. Note that y_1, y_2, and y_3 are represented by $\mathrm{ext}_H(1)$, $\mathrm{ext}_H(2)$, and $\mathrm{ext}_H(3)$; $\mathrm{ext}_H(4)$ and $\mathrm{ext}_H(5)$ are used to indicate distinguished nodes of H (a possibility not present in D). □

We only wish to consider hypergraphs of which every node represents a term (as the one in Fig. 5). To define such "jungles" [HabKrePlu] we need some more terminology.

First we need a notion of direction of a hyperedge e, in addition to the one given by the order of $\mathrm{nod}(e)$. This new notion of direction of an r-hyperedge

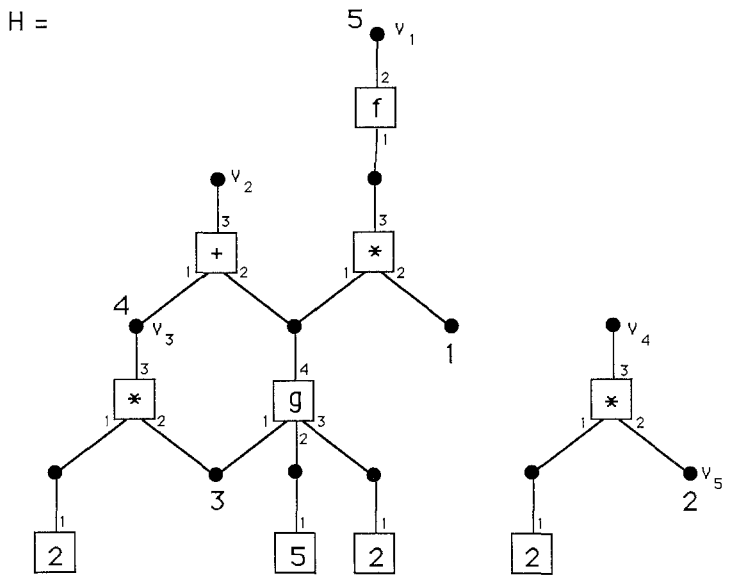

Fig. 5

e with $\mathrm{nod}(e)=(v_1, v_2, \ldots, v_r)$ is as follows (cf. [Cou3]). The nodes $v_1, v_2, \ldots, v_{r-1}$ are the "source" nodes of e, and we say that e is an *outgoing* edge of these nodes. The node v_r is the "target" node of e, and e is called an *incoming* edge of v_r. In this sense, according to [HabKre1 and 2; Hab; HabKrePlu], e is a hyperedge of type $(r-1, 1)$. For 2-hyperedges (i.e., ordinary edges) the two notions of direction coincide. Note that in the representation of terms by hypergraphs the direction is taken opposite to the one in DOAG's (cf. Figs. 4 and 5).

With this new notion of direction, a path in a hypergraph H from a node v to a node w is a sequence $(v_0, v_1, \ldots, v_n) \in V_H^{n+1}$ with $n \geqq 0$, $v_0 = v$, and $v_n = w$, such that for every $i \in [1, n]$ there exist $e \in E_H$ and $j \in [1, \mathrm{rank}(e)-1]$ with $\mathrm{nod}_H(e, j) = v_{i-1}$ and $\mathrm{nod}_H(e, \mathrm{rank}(e)) = v_i$. We call such a sequence $(v_0, v_1, \ldots, v_n)$ a *hyperpath* (of length n) from v to w. A *cycle* is a hyperpath of positive length from a node v to itself. A hypergraph is *acyclic* if it does not contain any cycles. Obviously, a hypergraph of which every node represents a term should be acyclic.

To be able to determine uniquely which term in $T(\Gamma, Y)$, where $Y = (y_1, y_2, \ldots, y_k)$ is a sequence of distinct variables, is represented by a node v in a hypergraph H with $\mathrm{rank}(H) \geqq k$, we demand firstly that the external nodes representing the k variables are distinct (because $\mathrm{ext}_H(i)$ represents y_i).

Definition. Let H be a hypergraph, and $k \in [0, \mathrm{rank}(H)]$. H is *k-identification-free* if $\mathrm{ext}_H(i) \neq \mathrm{ext}_H(j)$ for all $i, j \in [1, k]$ with $i \neq j$. □

Secondly, we demand that each node v, except the first k external nodes corresponding to variables, has precisely one incoming hyperedge.

Definition. Let H be a hypergraph. H is *one-incoming for k*, where $k \in [0, \mathrm{rank}(H)]$, if

(1) each $v \in V_H - \{\text{ext}_H(i) \mid 1 \leqq i \leqq k\}$ has precisely one incoming hyperedge, and
(2) no $v \in \{\text{ext}_H(i) \mid 1 \leqq i \leqq k\}$ has incoming hyperedges.

H is *one-incoming* if it is one-incoming for 0, i.e., each node of H has precisely one incoming hyperedge. □

Since a 0-hyperedge in a hypergraph is not incident with any node, it does not contribute to any term represented by a node of that hypergraph. Thus, for technical convenience, we omit 0-hyperedges. We are now ready to define the notion of jungle. It is the same as the one defined in [HabKrePlu; HofPlu], apart from a few technicalities (such as the presence of external nodes, cf. [Cou3]).

Definition. Let H be a hypergraph, and $k \in [0, \text{rank}(H)]$. H is a *jungle with k variables* if H is acyclic, k-identification-free, one-incoming for k, and has no 0-hyperedges. □

After having defined the hypergraphs of which the nodes represent terms, we now define which terms are represented by the nodes of such a hypergraph. For that purpose we use an unfolding operation that duplicates shared subterms, as with the unfolding of DOAG's.

Recall from Sect. 2.1 that, for a ranked alphabet Σ, $\text{dec}(\Sigma)$ is obtained by decreasing the rank of each symbol by 1.

Definition. Let H be a jungle with k variables over a ranked alphabet Σ, and let $v \in V_H$. Let $Y = (y_1, y_2, \ldots, y_k)$ be a sequence of k distinct variables. The *term associated with v in H over Y*, denoted $\text{term}(v, H, Y)$, is the term in $T(\text{dec}(\Sigma), Y)$ defined as

$$term(v, H, Y) = \begin{cases} y_i & \text{if } v \text{ is } \text{ext}_H(i) \text{ for some } i \in [1, k] \\ \text{lab}_H(e) & \text{if } e \text{ is the incoming edge of } v, \text{ and } e \text{ has rank 1} \\ \text{lab}_H(e)(\text{term}(\text{nod}_H(e, 1), H, Y), & \\ \quad \text{term}(\text{nod}_H(e, 2), H, Y) & \\ \quad \vdots & \\ \quad \text{term}(\text{nod}_H(e, r-1), H, Y)) & \\ \quad \text{if } e \text{ is the incoming edge of } v, & \\ \quad \text{and } e \text{ has rank } r \geqq 2. \quad \square & \end{cases}$$

Notice that in the above definition the recursion always ends, because H is acyclic (and finite).

Since, in H, the external nodes $\text{ext}_H(k+1)$, $\text{ext}_H(k+2)$, ..., $\text{ext}_H(\text{rank}(H))$ are distinguished nodes, H represents $\text{rank}(H) - k$ (not necessarily distinct) terms in particular. For example, if $Y = (y_1, y_2, y_3)$, then the jungle H of rank 5 with 3 variables, given in Fig. 5, represents 2 terms, viz.

$$\text{term}(\text{ext}_H(4), H, Y) = *(2, y_3) \quad \text{and} \quad \text{term}(\text{ext}_H(5), H, Y) = f(*(g(y_3, 5, 2), y_1)).$$

Thus, in case $\text{rank}(H) = k+1$, H represents one term in particular. Such jungles will be used to represent right-hand sides of semantic rules of an AG.

Definition. Let H be a jungle with $\text{rank}(H)-1$ variables, $\text{rank}(H)\geqq 1$. Let Y be a sequence of $\text{rank}(H)-1$ distinct variables. The *term associated with H over Y*, denoted $\text{term}(H, Y)$, is $\text{term}(\text{ext}_H(\text{rank}(H)), H, Y)$. □

In this paper we are mainly interested in jungles without variables, i.e., with 0 variables, in order to compare their associated terms (without variables) with those generated by AG's. That is why we focus attention on cfhg's generating jungles without variables with which only one term is associated, in the rest of this paper. From now on, by a *jungle* a jungle of rank 1 without variables is meant, unless it is explicitly mentioned that a jungle of rank m with k variables, for some k ($0\leqq k\leqq m$), is considered. This is equivalent to the following definition.

Definition. A *jungle* H is an acyclic one-incoming 1-hypergraph without 0-hyperedges. The *term associated with H*, denoted $\text{term}(H)$, is $\text{term}(\text{ext}_H(1), H, ( ))$. □

Definition. A cfhg G is *term-generating* if every hypergraph in $L(G)$ is a jungle. □

Thus the hypergraph language of a term-generating cfhg G consists of a set of jungles. The set of terms associated with these jungles is called the term language of G.

Definition. Let G be a term-generating cfhg. The *term language generated by G* is $\text{TERM}(G)=\{\text{term}(H)\mid H\in L(G)\}$. □

Notice that for a term-generating cfhg $G=(\Sigma, \Delta, P, S)$, $\text{TERM}(G)\subseteq T(\text{dec}(\Delta))$, the set of terms over $\text{dec}(\Delta)$.

The class of all term languages generated by cfhg's will be denoted as $\text{TERM}(\text{CFHG})$, i.e., $\text{TERM}(\text{CFHG})=\{\text{TERM}(G)\mid G \text{ is a term-generating cfhg}\}$.

Example. Consider the term-generating cfhg $G=(\Sigma, \Delta, P, S)$ with $\Sigma=\{S, A, f, g, h, a\}$, $\text{rank}_\Sigma(S)=\text{rank}_\Sigma(a)=1$, $\text{rank}_\Sigma(A)=\text{rank}_\Sigma(g)=\text{rank}_\Sigma(h)=2$, and $\text{rank}_\Sigma(f)=3$, $\Delta=\{f, g, h, a\}$, and $P=\{\pi_1, \pi_2, \pi_3\}$ as given in Fig. 6. For the jungle $H\in L(G)$ of Fig. 7, the associated term in $T(\text{dec}(\Delta))$ is

$$\text{term}(H)=f(g(f(g(f(a, a)), h(f(a, a)))),\\ h(f(g(f(a, a)), h(f(a, a))))).$$

Notice that application of production π_2 "causes" sharing. In the terms represented by the nodes of H, the terms $f(a, a)$ and $f(g(f(a, a)), h(f(a, a)))$ are shared. The term language generated by G can be described inductively by

(1) $f(a, a)\in\text{TERM}(G)$,
(2) if $t\in\text{TERM}(G)$ then $f(g(t), h(t))\in\text{TERM}(G)$. □

The right-hand sides of the productions of a term-generating cfhg G do not have to be jungles themselves. In fact, they even do not have to be jungles with variables (consider production π_1 of the cfhg defined in the above example). But (assuming that G is identification-free, loop-free, and reduced), the yield of every derivation subtree t is a subgraph of a jungle in $L(G)$; using this, it is not difficult to see that $\text{yield}(t)$ is a jungle with k variables for some $k\in[0, \text{rank}(\text{yield}(t))]$, provided we permute the sequence of external nodes of $\text{yield}(t)$ in such a way that the first k external nodes have no incoming hyperedge in $\text{yield}(t)$ and the remaining ones have an incoming hyperedge in $\text{yield}(t)$. This

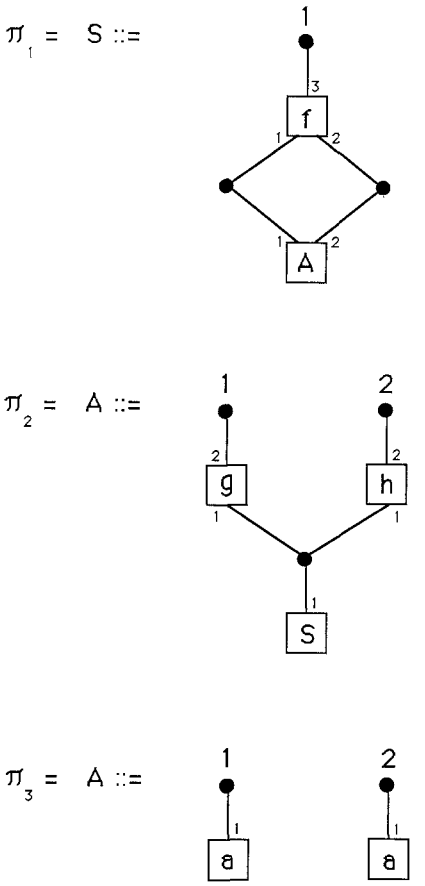

Fig. 6

also explains our interest in jungles (with variables) of arbitrary rank. In the example above, A generates terminal jungles of rank 2 (without variables).

We now consider a more complicated example to provide the reader already with some notion of the correspondence between cfhg's and AG's.

Example. Consider the cfhg $G=(\Sigma, \Delta, P, S)$ with

$$\Sigma=\{N, L, B, +, 2\uparrow, -, 0, 1\},$$

$$\text{rank}_\Sigma(N)=\text{rank}_\Sigma(0)=\text{rank}_\Sigma(1)=1,$$

$$\text{rank}_\Sigma(B)=\text{rank}_\Sigma(2\uparrow)=\text{rank}_\Sigma(-)=2,$$

$$\text{rank}_\Sigma(L)=\text{rank}_\Sigma(+)=3,$$

$\Delta=\{+, 2\uparrow, -, 0, 1\}$, $P=\{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6\}$, as given in Fig. 8, and $S=N$.

This term-generating cfhg generates the same term language as the AG G_{bin} of Example 2.2, viewed as a term-generating AG (i.e., to be precise, $L(G)$

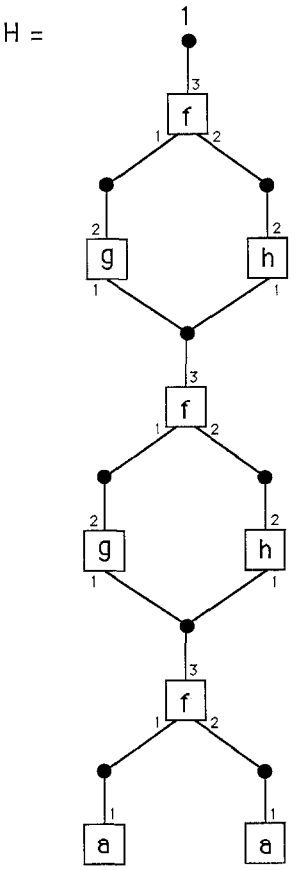

Fig. 7

$= \mathrm{OUT}((G_{\mathrm{bin}})_{\mathrm{term}})$). Thus, each jungle generated by G represents the "meaning" of a binary number.

The nonterminals of G are the nonterminals of the underlying context-free grammar G_0 of the AG G_{bin}. Their ranks in Σ are equal to the number of attributes they have. The terminals of G are the function symbols that occur in the semantic rules of the productions of P_0, and their ranks in Σ equal the number of arguments they have plus one. Thus $\Delta = \mathrm{inc}(\Gamma)$, where Γ is the ranked alphabet of the semantic domain of G_{bin}.

The productions of P correspond to those of P_0, including their semantic rules, with π_i corresponding to p_i. The order of the tentacles of the nonterminals and of the external nodes corresponds to an (arbitrary but fixed) order of the sets of attributes of these nonterminals. In this example $\mathrm{Att}(N) = \{v\}$, $\mathrm{Att}(L) = \{s, \ell, v\}$ and $\mathrm{Att}(B) = \{s, v\}$, in those orders. The order of the nedg-set of the right-hand side of a production of P is determined by the sequence of nonterminals in the right-hand side of the corresponding production of P_0.

Compare the derivation tree t shown in Fig. 9 with the one of (1.2) in [Knu], which is the derivation tree t' of the binary number 1101.01 (obtained by changing π_i into p_i in t). This derivation tree t yields the jungle H of Fig. 10, which

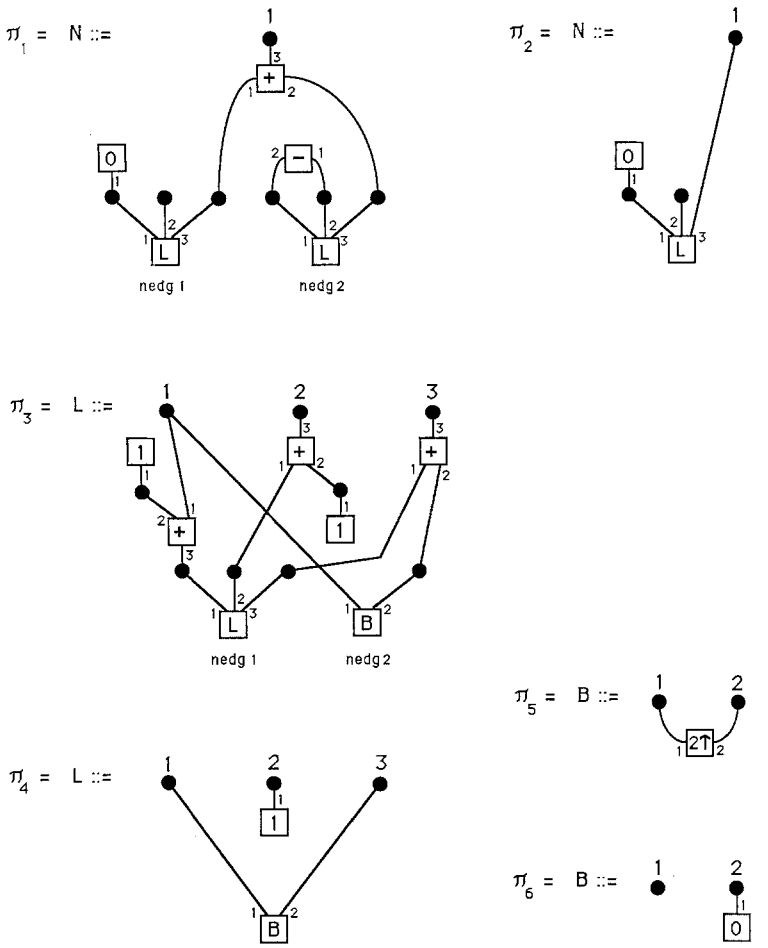

Fig. 8

represents the value of the designated attribute in t'. In fact, the term in $T(\mathrm{dec}(\Delta))$ associated with H is

$$\begin{aligned}\mathrm{term}(H) = +(+(+(+(2\uparrow(+(+(+(0,1),1),1)),\\ &\quad 2\uparrow(+(+(0,1),1))),\\ &\quad 0),\\ &\quad 2\uparrow(0)),\\ &\quad +(0,2\uparrow(-(+(1,1))))).\end{aligned}$$

This term is also the value of v of N in t', when G_{bin} is viewed as a term-generating AG. Evaluation of this term in the semantic domain $D=(V,\Gamma)$ of G_{bin} gives $(\mathrm{term}(H))_D = 13.25$, the meaning of 1101.01.

Notice that the subterms $+(+(0,1),1)$, 0, and $-(+(1,1))$ are being shared. This is "caused" by the applications of production π_3, and corresponds to

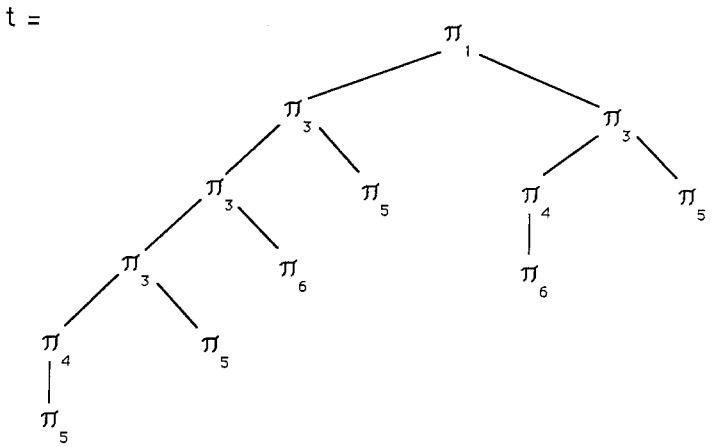

Fig. 9

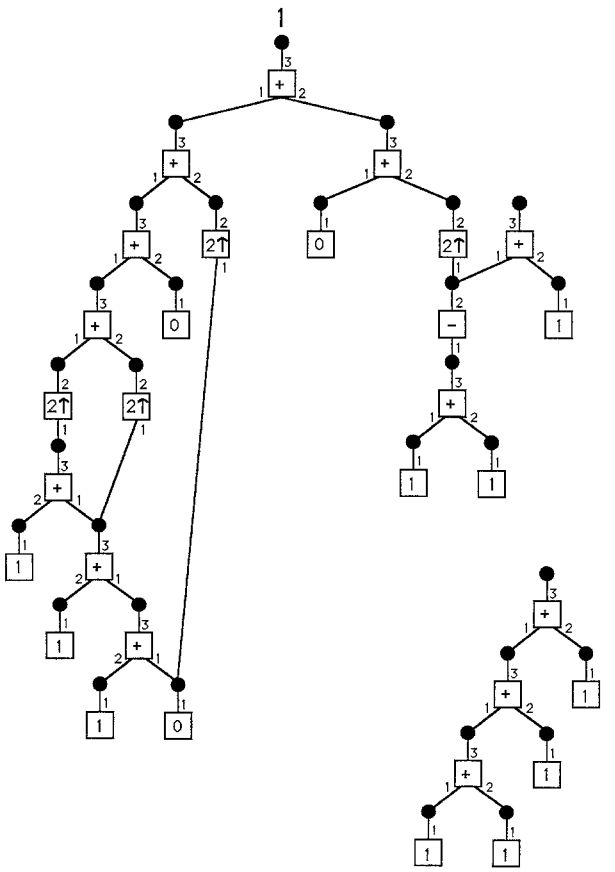

Fig. 10

the multiple use of the outside attribute $\langle s, 0\rangle$ in the semantic rules of the production p_3 of G_0.

Notice finally (cf. the discussion preceding this example) that B generates terminal jungles of rank 2 with 1 variable, and L generates terminal jungles of rank 3 with 1 variable (where no permutation of the external nodes is needed). □

The reader should be aware of the fact that the definition of TERM(CFHG) given in this section is quite flexible. The present definition was chosen to be suitable for the comparison of cfhg's with AG's as far as terms are concerned (in particular for the simulation of AG's by cfhg's, in the next section). In the remainder of this section we discuss some alternative ways of defining TERM(CFHG).

The above example (in particular Fig. 10) shows that not always all the hyperedges and nodes of a jungle are used in the computation of the term associated with that jungle. This is a consequence of the fact that the attribute ℓ of the first L in the production $N \to L.L$ and the attribute s of B in the (rightmost occurrence of the) production $B \to 0$ are not used by the AG G_{bin} in the computation of the value of the designated attribute of the root of the derivation tree t' of G_0, that corresponds to the derivation tree t of the cfhg G, given in Fig. 9. This is illustrated in Fig. 11 that shows the dependency graph $DG_{t'}$, see also (3.1) in [Knu]. In Fig. 11 a dashed edge from an attribute β to an attribute α indicates that the dependence of α on β is determined by a copy rule. A short incoming arrow indicates a semantic rule with a constant right-hand side (i.e., in $T(\Gamma)$).

We call the hyperedges and nodes of a jungle H that are not used in the computation of term(H), *garbage*. A jungle without garbage is said to be *clean*. A more formal way to define this, is the following.

Definition. Let H be a jungle with k variables. H is *clean* if for every $v \in V_H$ there exists a hyperpath from v to $\text{ext}_H(i)$, for some $i \in [k+1, \text{rank}(H)]$. □

A clean jungle of rank 1 without variables is shortly called a clean jungle. Compared to the representation of a term by a DOAG, a clean jungle can be viewed as a tree with sharing, where the external node of the jungle corresponds to the root of the tree.

Definition. A cfhg G is *clean term-generating* if every hypergraph in $L(G)$ is a clean jungle. □

Clearly, garbage can be removed from a jungle without changing the associated term. Thus, one may consider it to be more natural to represent terms by clean jungles only, and to restrict attention to clean term-generating cfhg's. In the "garbage theorem" (Theorem 5.3) it will be shown that this is possible without changing the term-generating power of cfhg's. At this moment we just mention this alternative way of defining TERM(CFHG) as $\{\text{TERM}(G) \mid G$ is a clean term-generating cfhg$\}$; we postpone the "garbage theorem" to Sect. 5.

As another example of the flexibility of the definition of the class TERM(CFHG), we do not have to restrict ourselves to term-generating cfhg's. We could just as well allow all cfhg's that generate 1-hypergraphs, and consider the terms associated with the (clean) jungles they generate. This is stated in the following theorem. At the same time we reassure the reader who is worried about the decidability of "(clean) term-generating".

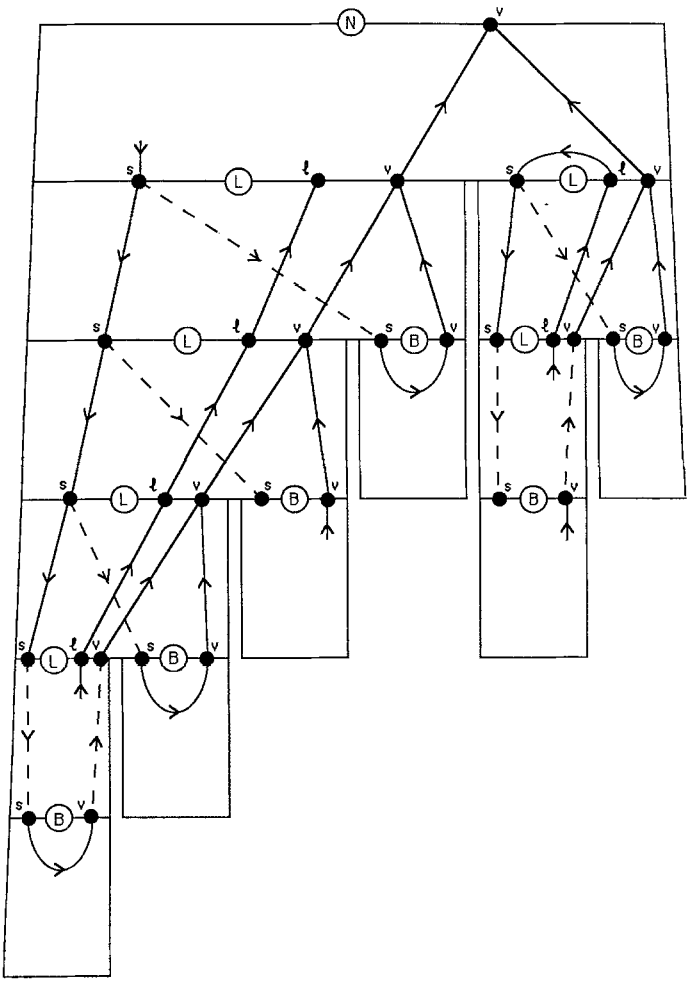

Fig. 11

Theorem 3.1 *Let G be a* cfhg *generating 1-hypergraphs.*

(1) It is decidable whether G is (clean) term-generating.
(2) $\{\text{term}(H) \mid H \in L(G), H \text{ is a (clean) jungle}\} \in \text{TERM(CFHG)}$.

Proof. Let $G=(\Sigma, \Delta, P, S)$ be a cfhg with $\text{rank}_\Sigma(S)=1$.

By definition, a 1-hypergraph is a jungle if it is acyclic, one-incoming, and has no 0-hyperedges. And it is a clean jungle if, moreover, there is a hyperpath from each of its nodes to ext(1). One can easily see that these three (four) properties for a 1-hypergraph to be a (clean) jungle can be expressed in an MSOL formula φ of $\mathscr{L}_{\Delta,1}$, as follows (for MSOL, see Sect. 2.3).

To define this φ, we first need MSOL formulas $\vartheta(v, w)$, $\psi(v, w)$, and $\zeta(e, v)$ to express that there is a hyperedge from a node v to a node w, that there is a hyperpath from v to w, and that a hyperedge e is incoming for a node v, respectively. There is a hyperpath from v to w iff for all sets of nodes V

with $v \in V$, if V is "closed" then $w \in V$. A set of nodes V is closed, expressed as $\theta(V)$ in $\mathscr{L}_{\Delta,1}$, iff all outgoing edges of nodes of V are incoming for nodes of V only.

(hyperedge) $\vartheta(v, w) \Leftrightarrow \bigvee_{a \in \Delta} \exists e \exists v_1 \exists v_2 \ldots \exists v_{r-1}$:

$$\left(\mathrm{edg}_a(e, v_1, v_2, \ldots, v_{r-1}, w) \wedge \left(\bigvee_{i=1}^{r-1} (v_i = v)\right)\right)$$

where r abbreviates $\mathrm{rank}_\Delta(a)$.

(closed) $\theta(V) \Leftrightarrow \forall v \forall w$: $(v \in V \wedge \vartheta(v, w)) \rightarrow w \in V$.

(hyperpath) $\psi(v, w) \Leftrightarrow \forall V$: $(v \in V \wedge \theta(V)) \rightarrow w \in V$.

(incoming) $\zeta(e, v) \Leftrightarrow \bigvee_{a \in \Delta} (\exists v_1 \exists v_2 \ldots \exists v_{r-1}$: $\mathrm{edg}_a(e, v_1, v_2, \ldots, v_{r-1}, v))$

where r abbreviates $\mathrm{rank}_\Delta(a)$.

Further we need the following auxiliary formulas.

(acyclic) $\varphi_1 \Leftrightarrow \neg(\exists v \exists w$: $\vartheta(v, w) \wedge \psi(w, v))$.

(one-incoming) $\varphi_2 \Leftrightarrow \forall v$: $((\exists e$: $\zeta(e, v))$

$$\wedge ((\forall e_1 \forall e_2 \colon \zeta(e_1, v) \wedge \zeta(e_2, v)) \rightarrow (e_1 = e_2))).$$

(no 0-hyperedges) $\varphi_3 \Leftrightarrow \bigwedge_{a \in \Delta_0} (\neg \exists e$: $\mathrm{edg}_a(e))$

where $\Delta_0 = \{a \in \Delta \mid \mathrm{rank}_\Delta(a) = 0\}$.

(clean) $\varphi_4 \Leftrightarrow \forall v$: $\psi(v, \mathrm{ext}(1))$.

Then the formula φ expressing that a 1-hypergraph is a (clean) jungle is defined as $\varphi \Leftrightarrow \varphi_1 \wedge \varphi_2 \wedge \varphi_3 (\wedge \varphi_4)$.

(1) By Proposition 2.5(1), it is decidable whether $H \models \varphi$ for all $H \in L(G)$. This means that it is decidable whether G is (clean) term-generating.
(2) By Proposition 2.5(2), one can construct a cfhg G' generating the hypergraph language $L(G') = \{H \in L(G) \mid H \models \varphi\}$. G' is (clean) term-generating, because all $H \in L(G')$ satisfy φ. Thus the term language generated by G' is defined. In fact $\mathrm{TERM}(G') = \{\mathrm{term}(H) \mid H \in L(G')\} = \{\mathrm{term}(H) \mid H \in L(G)$ and $H \models \varphi\}$, i.e., $\{\mathrm{term}(H) \mid H \in L(G), H$ is a (clean) jungle$\} \in \mathrm{TERM}(\mathrm{CFHG})$. □

4. Simulation of attribute grammars by context-free hypergraph grammars

In this paper we compare the term-generating power of cfhg's with that of AG's. In particular, we prove that $\mathrm{TERM}(\mathrm{CFHG}) = \mathrm{OUT}(\mathrm{AG}, \mathrm{TERMS})$. In this section we show that $\mathrm{OUT}(\mathrm{AG}, \mathrm{TERMS}) \subseteq \mathrm{TERM}(\mathrm{CFHG})$, i.e., that term-generating cfhg's can simulate all term-generating AG's.

The output language of an AG G is obtained by considering all the derivation trees of its underlying grammar G_0, and determining the values of the designated attributes of their roots. The computation of such a value for a derivation tree t of G_0 depends on the semantic rules of the productions that are applied in t.

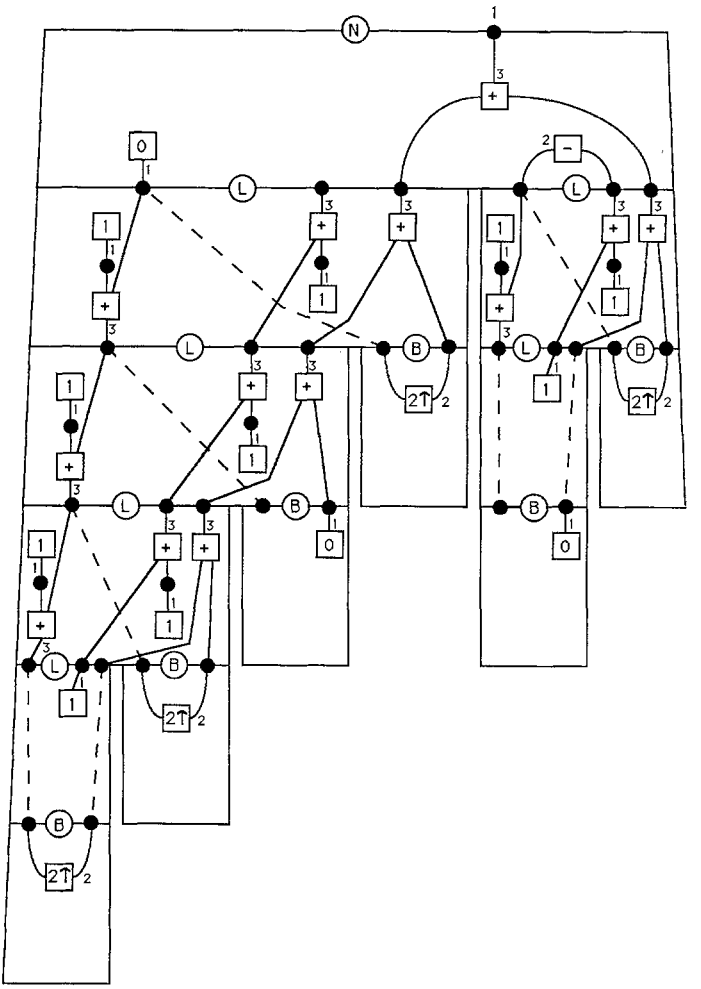

Fig. 12

Thus a cfhg has to simulate both the underlying cfg and the evaluation of the semantic rules of the AG. The simulation of the cfg is straightforward, because in fact a cfhg may be viewed as a cfg in which graphs are added to the productions. In particular, we shall represent the attributes of the nonterminals in a cfg production p by nodes in the right-hand side of the corresponding cfhg production π, as in the dependency graph DG_p. The nonterminals in $\mathrm{rhs}(p)$ will be represented in $\mathrm{rhs}(\pi)$ by nonterminal hyperedges with tentacles to the nodes representing their attributes. The attributes of $\mathrm{lhs}(p)$ form the external nodes of $\mathrm{rhs}(\pi)$. The way in which a cfhg can simulate the evaluation of a semantic rule is explained in the following example.

Example. Consider again the term-generating AG G_{bin} of Example 2.2. If we "substitute" the corresponding cfhg productions, given in Fig. 8, in the dependency graph of Fig. 11, we get the jungle H of Fig. 12. A dashed line between

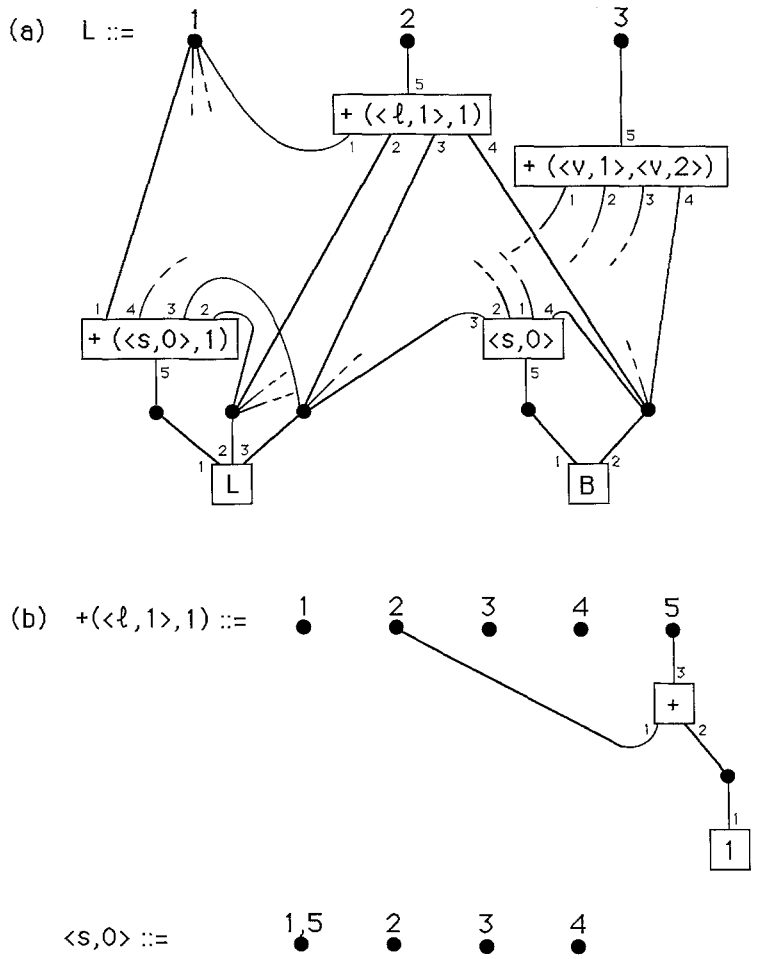

Fig. 13

two nodes indicates that these nodes are identified in H. To keep the figure surveyable, for every hyperedge e, we only numbered the tentacle rank(e) indicating the direction of e. The same jungle is also shown in Fig. 10. From Fig. 12 it should be clear that term(H) is the value of the designated attribute v of root(t), where t is the derivation tree given in Fig. 9.

For technical reasons we shall define separate cfhg productions for the simulation of the productions of the underlying grammar G_0, including their semantic rules, and for the simulation of the evaluation of these semantic rules (in contrast to the cfhg productions of Fig. 8).

For example, for production $p_3 = L \to LB$ of P_0, the corresponding cfhg production, that simulates p_3 including its semantic rules, is shown in Fig. 13a. The attributes of p_3 are represented by nodes. The semantic rules of p_3 are represented by nonterminal hyperedges labeled with the right-hand sides of these semantic rules. Since, in principle, a semantic rule defines an inside attribute

of p_3 in terms of all the outside attributes of p_3, a hyperedge e representing $(\langle \alpha, j\rangle = t)$ has tentacles to all nodes representing the outside attributes, and to the node representing $\langle \alpha, j\rangle$. In particular, $\mathrm{nod}(e, i)$ represents the i-th attribute in the sequence $\mathrm{outs}(p_3)$, $1 \leqq i \leqq 4 = \#\mathrm{outs}(p_3)$, and $\mathrm{nod}(e, \#\mathrm{outs}(p_3)+1)$ represents $\langle \alpha, j\rangle$. Notice that the direction of e corresponds to the direction of the original dependencies determined by $\langle \alpha, j\rangle = t$. For clearness' sake, some tentacles are not drawn completely in Fig. 13a, but the nodes with which they are incident should be clear (assuming that $\mathrm{outs}(p_3)$ is the sequence $(\langle s, 0\rangle, \langle \ell, 1\rangle, \langle v, 1\rangle, \langle v, 2\rangle)$).

The other cfhg productions corresponding to p_3 are used to rewrite the nonterminal hyperedges that represent the semantic rules of p_3, into corresponding jungles of rank $\#\mathrm{outs}(p_3)+1$ with $\#\mathrm{outs}(p_3)$ variables. Thus, they simulate the evaluation of the semantic rules of p_3. In Fig. 13b two of these productions are given. Notice that the application of all these productions to Fig. 13a would produce production π_3 of Fig. 8. □

To translate the right-hand side $t \in T(\Gamma, \mathrm{outs}(p))$ of a semantic rule of a production p of G_0 into a jungle H (with variables) such that $\mathrm{term}(H, \mathrm{outs}(p)) = t$, we associate with t and $\mathrm{outs}(p)$ a jungle of rank $\#\mathrm{outs}(p)+1$ with $\#\mathrm{outs}(p)$ variables, denoted $\mathrm{jung}(t, \mathrm{outs}(p))$. This hypergraph resembles the usual tree corresponding to t, except that the variables are shared. In [HabKrePlu] it is called the "variable-collapsed tree" corresponding to t.

Definition. Let Γ be a ranked alphabet, let $Y = (y_1, y_2, \ldots, y_k)$ be a sequence of k distinct variables, and let $t \in T(\Gamma, Y)$. The *jungle associated with t and Y*, denoted $\mathrm{jung}(t, Y)$, is the jungle of rank $k+1$ with k variables over $\mathrm{inc}(\Gamma)$ defined inductively as follows. In the definition, we take $\mathrm{ext}(i)$ equal to y_i, for all $1 \leqq i \leqq k$, in all hypergraphs.

(1) If $t = y_i$, $1 \leqq i \leqq k$, then

$$\mathrm{jung}(t, Y) = (\{y_1, y_2, \ldots, y_k\}, \emptyset, \emptyset, \emptyset, (y_1, y_2, \ldots, y_k, y_i)).$$

(2) Let $t = \gamma(t_1, t_2, \ldots, t_n)$ with $\gamma \in \Gamma$, $\mathrm{rank}_\Gamma(\gamma) = n$, $n \geqq 0$, and $t_i \in T(\Gamma, Y)$ for all $i \in [1, n]$. Denote $\mathrm{jung}(t_i, Y)$ as H_i. Take isomorphic copies of $H_1, H_2, \ldots, H_n$, such that for all $i, j \in [1, n]$ with $i \neq j$, $V_{H_i} \cap V_{H_j} = \{y_1, y_2, \ldots, y_k\}$ and $E_{H_i} \cap E_{H_j} = \emptyset$. Then $\mathrm{jung}(t, Y) = (V, E, \mathrm{nod}, \mathrm{lab}, \mathrm{ext})$ with

$$V = \{v, y_1, y_2, \ldots, y_k\} \cup \bigcup_{i=1}^{n} V_{H_i}, \text{ where } v \text{ is a "new" node,}$$

$$E = \{\varepsilon\} \cup \bigcup_{i=1}^{n} E_{H_i}, \text{ where } \varepsilon \text{ is a "new" hyperedge,}$$

$$\mathrm{nod}(e) = \begin{cases} (\mathrm{ext}_{H_1}(k+1), \mathrm{ext}_{H_2}(k+1), \ldots, \mathrm{ext}_{H_n}(k+1), v) & \text{if } e = \varepsilon \\ \mathrm{nod}_{H_i}(e) & \text{if } e \in E_{H_i},\ 1 \leqq i \leqq n, \end{cases}$$

$$\mathrm{lab}(e) = \begin{cases} \gamma & \text{if } e = \varepsilon \\ \mathrm{lab}_{H_i}(e) & \text{if } e \in E_{H_i},\ 1 \leqq i \leqq n, \text{ and} \end{cases}$$

$$\mathrm{ext} = (y_1, y_2, \ldots, y_k, v). \quad \square$$

Example. Consider the ranked alphabet $\Gamma = \{f, g, +, *, 2, 5\}$ with $\mathrm{rank}_\Gamma(g) = 3$, $\mathrm{rank}_\Gamma(+) = \mathrm{rank}_\Gamma(*) = 2$, $\mathrm{rank}_\Gamma(f) = 1$, and $\mathrm{rank}_\Gamma(2) = \mathrm{rank}_\Gamma(5) = 0$. Let Y

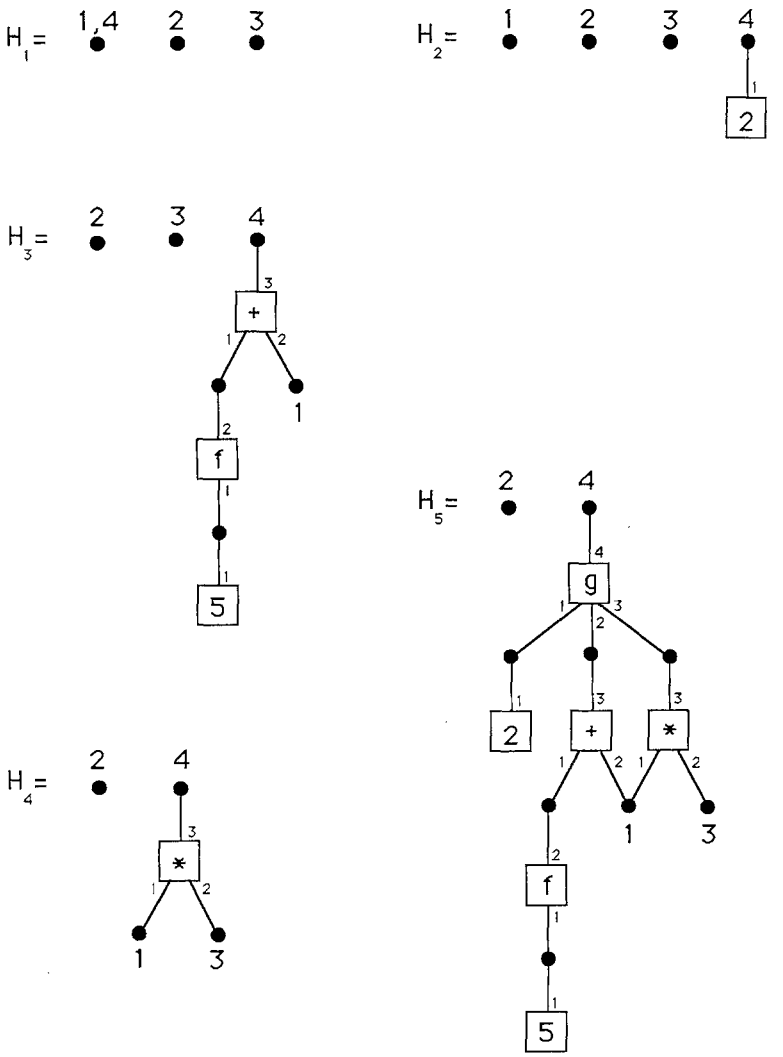

Fig. 14

$=(y_1, y_2, y_3)$, and consider the following terms in $T(\Gamma, Y)$: $t_1=y_1$, $t_2=2$, $t_3=+(f(5), y_1)$, $t_4=*(y_1, y_3)$, and $t_5=g(t_2, t_3, t_4)=g(2, +(f(5), y_1), *(y_1, y_3))$. The jungles with 3 variables over $\mathrm{inc}(\Gamma)$ associated with these terms and Y are given in Fig. 14, i.e., $H_i=\mathrm{jung}(t_i, Y)$, for $i\in[1, 5]$. □

As a consequence of the above definition and the definition of $\mathrm{term}(H, Y)$, $\mathrm{term}(\mathrm{jung}(t, Y), Y)=t$ for all terms $t\in T(\Gamma, Y)$. This can be proved by induction on the structure of t. The formal proof is left to the reader. In fact, this is the only property of $\mathrm{jung}(t, Y)$ that we need. In the above example

$$\begin{aligned}&\mathrm{term}(\mathrm{jung}(t_5, Y), Y)\\&\quad=\mathrm{term}(H_5, Y)=g(\mathrm{term}(H_2, Y), \mathrm{term}(H_3, Y), \mathrm{term}(H_4, Y))\\&\quad=g(\mathrm{term}(\mathrm{jung}(t_2, Y), Y), \mathrm{term}(\mathrm{jung}(t_3, Y), Y), \mathrm{term}(\mathrm{jung}(t_4, Y), Y))\\&\quad=g(2, +(f(5), y_1), *(y_1, y_3))=t_5.\end{aligned}$$

To prove that OUT(AG, TERMS)$\subseteq$TERM(CFHG), we shall now give a detailed description of the siulation of a term-generating AG by a cfhg, using the above definition to simulate the evaluation of the semantic rules by jungles (cf. Fig. 13b that shows the jungles jung$(+(\langle \ell, 1\rangle, 1), Y)$ and jung$(\langle s, 0\rangle, Y)$ associated with the right-hand sides of the semantic rules of the production p_3 of Example 2.2 and $Y=(\langle s, 0\rangle, \langle \ell, 1\rangle, \langle v, 1\rangle, \langle v, 2\rangle)$).

Lemma 4.1 *For every term-generating* AG G *there exists a term-generating* cfhg G' *such that* TERM(G')=OUT(G). *Moreover, if* G *is reduced then* G' *is clean term-generating.*

Proof. Let G be a non-circular attribute grammar over $(T(\Gamma), \Gamma)$ for some ranked alphabet Γ, and let $G_0=(N_0, T_0, P_0, S_0)$ be its underlying context-free grammar. This implies that the terminal alphabet of G' must be inc(Γ).

Every term $t\in T(\Gamma)$ in OUT(G) is the value of the designated attribute of the root of some derivation tree t of G_0. We shall construct a cfhg G' such that if t' is a derivation tree of G' that corresponds to t, then term(yield$(t'))=t$, see Fig. 12. For that purpose, each attribute α of a node x of t will be represented by a node v in yield(t') such that term$(v,$ yield$(t'), (\,))$ equals the value of $\langle \alpha, x\rangle$. Due to copy rules, distinct attributes may be represented by the same node.

Every $X\in N_0$ is a nonterminal of G' of rank $\#\mathrm{Att}(X)$; in particular, S_0 is the initial nonterminal of G' (with rank 1). These nonterminals will be used to simulate the derivation trees of G_0. The terminals of G_0 need not be simulated by G', because they have no attributes and consequently do not influence OUT(G).

Every production $p=X_0\to w_0 X_1 w_1 X_2 w_2 \ldots X_n w_n$ of G_0 is translated into exactly one production $p'=(X_0, H)$ of G', which simulates both p and its semantic rules. The nodes of H are the attributes of p, i.e., ins$(p)\cup$outs(p). The attributes of X_0 are the external nodes of H. Besides, H has $n+\#\mathrm{ins}(p)$ nonterminal hyperedges, as follows. Each nonterminal X_i, $1\leq i\leq n$, corresponds to a hyperedge labeled X_i of which the tentacles are connected to the nodes representing its attributes. Each semantic rule $(\langle \alpha, j\rangle = t)\in r_p$, for some inside attribute $\langle \alpha, j\rangle$, is represented by a hyperedge with tentacles to all nodes representing outside attributes of p, and to the node representing $\langle \alpha, j\rangle$. To simulate the evaluation of the semantic rule it represents, such a hyperedge will be rewritten to jung $(t,$ outs$(p))$. To do this, we use a set N_{rs} of nonterminals of G' that consists of all pairs $\langle t, p\rangle$ representing the right-hand side t of a semantic rule of a production $p\in P_0$. These nonterminals $\langle t, p\rangle$ have rank $\#\mathrm{outs}(p)+1$. Notice that p must be added to t to determine its set of variables outs(p). In fact, t can occur in semantic rules of different productions. For example, the term $+(\langle v, 1\rangle, \langle v, 2\rangle)$ is the right-hand side of a semantic rule of both productions p_1 and p_3 of Example 2.2 and outs$(p_1)\neq$outs(p_3).

The formal definition of the cfhg G' is as follows. Let

$$N_{rs}=\{\langle t, p\rangle \mid p\in P_0,\ t\in T(\Gamma, \mathrm{outs}(p)),\ r_p \text{ contains } (\langle \alpha, j\rangle = t) \text{ for some } \langle \alpha, j\rangle\in \mathrm{ins}(p)\}.$$

We assume for the sake of convenience that $\mathrm{Att}(X)=[1, \#\mathrm{Att}(X)]$ for all $X\in N_0$.

Then G' is the cfhg $(\Sigma, \Delta, P', S_0)$ where $\Sigma = N_0 \cup N_{rs} \cup \Gamma$ with

$$\mathrm{rank}_\Sigma(\sigma) = \begin{cases} \#\mathrm{Att}(\sigma) & \text{if } \sigma \in N_0 \\ \#\mathrm{outs}(p)+1 & \text{if } \sigma = \langle t, p\rangle \in N_{rs} \\ \mathrm{rank}_\Gamma(\sigma)+1 & \text{if } \sigma \in \Gamma \end{cases}$$

$\Delta = \mathrm{inc}(\Gamma)$, and P' is constructed as follows.

Let $p = X_0 \to w_0 X_1 w_1 X_2 w_2 \ldots X_n w_n \in P_0$. Then P' contains the production $p' = (X_0, H)$ with $H = (V, E, \mathrm{nod}, \mathrm{lab}, \mathrm{ext})$ where

$$V = \mathrm{ins}(p) \cup \mathrm{outs}(p),$$

$$E = \{e_1, e_2, \ldots, e_n\} \cup \mathrm{ins}(p),$$

$$\mathrm{ext} = (\langle 1, 0\rangle, \langle 2, 0\rangle, \ldots, \langle \#\mathrm{Att}(X_0), 0\rangle),$$

and for all $i \in [1, n]$:

$$\mathrm{lab}(e_i) = X_i,$$

$$\mathrm{nod}(e_i) = (\langle 1, i\rangle, \langle 2, i\rangle, \ldots, \langle \#\mathrm{Att}(X_i), i\rangle),$$

and for all $\langle \alpha, j\rangle \in \mathrm{ins}(p)$ with $(\langle \alpha, j\rangle = t) \in r_p$:

$$\mathrm{lab}(\langle \alpha, j\rangle) = \langle t, p\rangle,$$

$$\mathrm{nod}(\langle \alpha, j\rangle) = (\mathrm{outs}(p, 1), \mathrm{outs}(p, 2), \ldots, \mathrm{outs}(p, \#\mathrm{outs}(p)), \langle \alpha, j\rangle),$$

where $\mathrm{outs}(p, j)$ denotes the j-th element of $\mathrm{outs}(p)$.

Furthermore P' contains the productions $(\langle t, p\rangle, \mathrm{jung}(t, \mathrm{outs}(p)))$ for all $\langle t, p\rangle \in N_{rs}$.

This ends the definition of G'. Note that G' is loop-free, but not necessarily identification-free (cf. Fig. 13b).

Now we shall argue that G' generates jungles, i.e., acyclic (i), one-incoming (ii), hypergraphs of rank 1 (iii) without 0-hyperedges (iv), that $\mathrm{TERM}(G') = \mathrm{OUT}(G)$ (v), and that G' generates clean jungles if G is reduced (vi).

Since $\mathrm{rank}_\Sigma(S_0) = 1$, G' generates 1-hypergraphs. These hypergraphs do not have any 0-hyperedges because Δ contains no symbols of rank 0 (recall that $\Delta = \mathrm{inc}(\Gamma)$). Thus, properties (iii) and (iv) are satisfied.

To every derivation tree of G_0 corresponds a unique derivation tree of G'. In fact, for each production $p \in P_0$ we have constructed one production p' of P', and every $\langle t, p\rangle \in N_{rs}$ is the left-hand side of one production of P'.

Because of the non-circularity of G, the value of every attribute $\langle \alpha, x\rangle$ of a node x in a derivation subtree ξ of G_0, is uniquely defined in terms of the inherited attributes of the root of ξ. This means that if $\mathrm{root}(\xi)$ is labeled by a production p with $\mathrm{lhs}(p) = X_0$, then this value can be viewed as a term in $T(\Gamma, Y)$, where Y is a sequence obtained by giving the set $\{\langle i, \mathrm{root}(\xi)\rangle \mid i \in \mathrm{Inh}(X_0)\}$ some order. A formal definition of this term will not be given, because it should be intuitively clear. It will be denoted by $\mathrm{val}_G(\langle \alpha, x\rangle, \xi)$, in agreement with the case that ξ is a derivation tree.

Since in a jungle with k variables the first k external nodes represent the variables, we assume (without loss of generality) that $\mathrm{Att}(X) = [1, \#\mathrm{Att}(X)]$

is such that $\mathrm{Inh}(X)=[1, \#\mathrm{Inh}(X)]$ and $\mathrm{Syn}(X)=[\#\mathrm{Inh}(X)+1, \#\mathrm{Att}(X)]$, for every $X\in N_0$.

The following claim is used to show properties (i), (ii), (v), and (vi).

Claim. Let t be a derivation subtree of G_0 of which the root x is labeled $p=X_0\to w_0X_1w_1X_2w_2\ldots X_nw_n$. Let t' be the corresponding derivation subtree of G', with $\mathrm{yield}(t')=F$. Then F is a jungle of rank $\#\mathrm{Att}(X_0)$ with $\#\mathrm{Inh}(X_0)$ variables such that, for every $s\in\mathrm{Syn}(X_0)$, $\mathrm{term}(\mathrm{ext}_F(s), F, Y)=\mathrm{val}_G(\langle s, x\rangle, t)$, where $Y=(\langle 1, x\rangle, \langle 2, x\rangle, \ldots, \langle \#\mathrm{Inh}(X_0), x\rangle)$. Moreover, F is clean if G is reduced.

This claim can be proved in a straightforward way with induction on the structure of the derivation subtrees of G_0. The formal proofs are omitted, but the following four informal remarks on the properties (i), (ii), (v), and (vi), respectively, should help to convince the reader of the correctness of the claim.

Remark 1. For every semantic rule $(\langle\alpha, j\rangle=t)\in r_q$ of a production $q\in P_0$ that is used in t, there is a hyperpath in $\mathrm{jung}(t, \mathrm{outs}(q))$ from every $\langle\beta, i\rangle$ that occurs in t to $\langle\alpha, j\rangle$ (or, more precisely, from $\mathrm{ext}(k)$ to $\mathrm{ext}(\#\mathrm{outs}(q)+1)$, if $\langle\beta, i\rangle$ is the k-th element of $\mathrm{outs}(q)$). In particular, if $t=\langle\beta, i\rangle$ then this hyperpath has length 0, i.e., $\langle\alpha, j\rangle$ and $\langle\beta, i\rangle$ are identified in $\mathrm{jung}(t, \mathrm{outs}(q))$. Moreover, there is no such path if $\langle\beta, i\rangle$ does not occur in t. Thus, for $\langle\alpha, j\rangle\in\mathrm{ins}(q)$ and $\langle\beta, i\rangle\in\mathrm{outs}(q)$, there is a hyperpath in the part of F corresponding to q, from the node representing $\langle\beta, i\rangle$ to the node representing $\langle\alpha, j\rangle$ if and only if $\langle\alpha, j\rangle$ depends on $\langle\beta, i\rangle$. Hence, since all $\mathrm{jung}(t, \mathrm{outs}(q))$ are acyclic by definition, F is acyclic iff $\overline{DG}_t$ is acyclic, where $\overline{DG}_t$ is the graph obtained from DG_t by contracting all edges that correspond to copy rules (i.e., all dashed lines in Fig. 11). Since G is non-circular, DG_t is acyclic. From this it easily follows that $\overline{DG}_t$ is acyclic too (because, in DG_t, there is only one path from the start node of an edge that will be contracted to its end node, viz. the one consisting of the contracted edge).

Remark 2. Every node v in F not representing an attribute has one incoming hyperedge in F, because it corresponds to an internal node of $\mathrm{jung}(t, \mathrm{outs}(q))$ for some t and q, which is one-incoming for $\#\mathrm{outs}(q)$ by definition. For all nodes of F that represent attributes, their incoming edges are "caused" by the semantic rules that define these attributes. Since every attribute of t (except those in Y) is defined by a unique semantic rule, the node representing the attribute has one incoming hyperedge. Note however that one node may represent several attributes. In that case, these attributes are defined by copy rules and one semantic rule that is either not a copy rule (hence the node has one incoming hyperedge) or a copy rule of which the right-hand side is an element of Y. Thus, F is one-incoming for $\#\mathrm{Inh}(X_0)$.

Remark 3. Let $v_{\alpha, j}$ be the node in F that represents the attribute $\langle\alpha, j\rangle\in\mathrm{ins}(p)\cup\mathrm{outs}(p)$. For a semantic rule $(\langle\alpha, j\rangle=t)\in r_p$, it should be clear that $\mathrm{term}(v_{\alpha, j}, F, Y)$ equals $\mathrm{term}(\mathrm{jung}(t, \mathrm{outs}(p)), \mathrm{outs}(p))$ in which every $\langle\beta, i\rangle\in\mathrm{outs}(p)$ is replaced by $\mathrm{term}(v_{\beta, i}, F, Y)$. Since $\mathrm{term}(\mathrm{jung}(t, Y), Y)=t$, for every $t\in T(\Gamma, Y)$, this implies that $\mathrm{term}(v_{\alpha, j}, F, Y)$ equals the term t, with the same replacement. This can be used in an inductive proof of the fact that $\mathrm{term}(v_{\alpha, j}, F, Y)=\mathrm{val}_G(\langle\alpha, x_j\rangle, t)$ for all $\langle\alpha, j\rangle$ (where x_0 denotes x and x_j the j-th son of x). The induction is on any linear order of the $\langle\alpha, x_j\rangle$ obtained from a topological order of DG_t.

Remark 4. Lemma 4.1 of [Fil] says that if the non-circular AG G is reduced then, for each production $q \in P_0$, every outside attribute of q occurs in at least one right-hand side of a semantic rule in r_q. Hence, if G is reduced then from each node in DG_t there is a path in DG_t to at least one of the nodes $\langle s, \text{root}(t)\rangle$, where $s \in \text{Syn}(X_0)$. As observed in Remark 1, for every $\langle \alpha, j\rangle \in \text{ins}(q)$ and $\langle \beta, i\rangle \in \text{outs}(q)$ of a production $q \in P_0$, there is a hyperpath (possibly of length 0) in F from the node representing $\langle \beta, i\rangle$ to the node representing $\langle \alpha, j\rangle$ if there is an edge in DG_q from $\langle \beta, i\rangle$ to $\langle \alpha, j\rangle$. Thus, if G is reduced then, for all $v \in V_F$, there is a hyperpath in F from v to one of the external nodes $\text{ext}_F(s)$, where $s \in \text{Syn}(X_0)$, i.e., F is clean.

This ends our remarks on the claim. Now let t be a derivation tree of G_0. Let t' be the corresponding derivation tree of G'. From the claim we can conclude that $\text{yield}_{G'}(t')$ is a jungle such that $\text{term}(\text{yield}_{G'}(t')) = \text{val}_G(\langle \alpha_d, \text{root}(t)\rangle, t)$, and that $\text{yield}_{G'}(t')$ is clean if G is reduced. This shows that G' is term-generating, that $\text{TERM}(G') = \text{OUT}(G)$, and that G' is clean term-generating in case G is reduced. □

By firstly applying Proposition 2.1 to the AG G, we can always construct a clean term-generating cfhg of which the term language equals the output language of G. Note however that, as shown in [Fil], reduction of an AG takes exponential time in general, whereas our construction of a cfhg clearly takes polynomial time.

We end this section with a discussion concerning the translations realized by AG's.

In the proof of Lemma 4.1, the set of productions of the cfhg G' consists of two parts: a production p' associated with every production p of the underlying grammar G_0 of a term-generating AG G over a semantic domain $(T(\Gamma), \Gamma)$, and a production for every $\langle t, p\rangle \in N_{rs}$. Using the context-freeness of cfhg's (see [Cou2]), it should be clear that an equivalent cfhg G'' is obtained by applying all $\langle t, p\rangle$-productions to the right-hand sides of all productions p'. Thus, G'' contains one production p'' for every $p \in P_0$, and no other productions. (By the way, this G'' is in general neither loop-free nor identification-free.) As an example, for $G = G_{\text{bin}}$, G'' is the cfhg of Fig. 8 (where p_i'' is denoted π_i). In particular, as mentioned before, the production of Fig. 13a turns into production π_3 of Fig. 8. For every derivation tree t of G_0 we have, as in the proof of Lemma 4.1, that $\text{val}_G(\langle \alpha_d, \text{root}(t)\rangle, t) = \text{term}(\text{yield}_{G''}(\varphi(t)))$, where $\varphi(t)$ is obtained from t by relabeling every p with p''.

Thus, for an AG G with an arbitrary semantic domain D, the translation realized by G is $\tau(G) = \{(\text{yield}(t), (\text{term}(\text{yield}_{G''_{\text{term}}}(\varphi(t))))_D) \mid t \text{ is a derivation tree of } G_0\}$. (Recall from Sect. 2.2 that every AG G determines a term-generating AG G_{term}). This suggests the following attribute evaluation method for G. Given a derivation tree t of G_0, turn it into a derivation tree $\varphi(t)$ of G''_{term}. Compute the jungle $H = \text{yield}_{G''_{\text{term}}}(\varphi(t))$, in a bottom-up fashion. Compute the value of H in D, i.e., $(\text{term}(H))_D$. An evaluation method of this type, called the DAG-evaluator, is described in [Mad] and implemented in the NEATS System (see also [DerJouLor]). The advantage of the method is that copy rules do not have to be executed, because the corresponding nodes are already identified in the right-hand sides of the productions of G''_{term}. The disadvantage of the method is that the jungle H may take a lot of space (which can be improved by evaluating "sub-jungles" as soon as possible).

We finally note that, of course, any term-generating cfhg G, together with a semantic domain $D=(V, \Gamma)$, can be viewed as a syntax-directed translation device, realizing the translation $\{(\ell, (\text{term}(\text{yield}_G(\ell)))_D) \mid \ell$ is a derivation tree of $G\}$ (cf. the "pushdown processor" of [AhoUll, Sect. 9.2], that translates strings into graphs). Above, we have shown that the translation of every AG can be viewed as a parsing phase followed by such a translation realized by a cfhg with the same semantic domain. In Sect. 6 we shall define a cfhg-based translation device that translates strings (rather than trees) into values of the semantic domain, and show that this device has the same translation power as the AG.

5. Simulation of context-free hypergraph grammars by attribute grammars

In this section we examine the (more surprising) converse of Lemma 4.1. Given a term-generating cfhg, it is not very hard to construct an underlying context-free grammar of an AG such that there is a 1–1 correspondence between their derivation trees. The simulation of the jungles, that are the yields of the derivation trees of the cfhg, by the attribute description and the semantic rules of an AG is less straightforward. We shall add attributes to the nonterminals of the cfg which can be distinguished in inherited and synthesized attributes. This distinction and the definition of the semantic rules is based on the fact that jungles are one-incoming.

Example. Consider the clean term-generating cfhg $G=(\Sigma, \Delta, P, S)$ with $\Sigma=\{S, B, F, +, 1\}$,

$$\text{rank}_\Sigma(F)=4, \quad \text{rank}_\Sigma(+)=3, \quad \text{rank}_\Sigma(B)=2,$$

$$\text{rank}_\Sigma(S)=\text{rank}_\Sigma(1)=1,$$

$$\Delta=\{+, 1\},$$

and $P=\{\pi_1, \pi_2, \pi_3, \pi_4\}$ as given in Fig. 15.

This cfhg generates clean jungles of the form given in Fig. 16 (see also Fig. 17, where nodes connected by a dashed line should be identified). Each jungle generated by G represents the "construction" of a Fibonacci number $\varphi(n)$, for some $n \geqq 4$, where $\varphi(1)=1$, $\varphi(2)=1$, and $\varphi(n)=\varphi(n-1)+\varphi(n-2)$ for all $n \geqq 3$. For example, the jungle H of Fig. 16 represents the construction of $\varphi(7)=13$, with

$$\text{term}(H)=+(+(+(+(+(1, 1), 1), +(1, 1)), +(+(1, 1), 1)), +(+(+(1, 1), 1), +(1, 1))).$$

Note that, for the purpose of illustration, we do not consider the most natural cfhg to generate these jungles.

Now we shall define a term-generating AG G' with $\text{OUT}(G')=\text{TERM}(G)$.

(1) The underlying grammar G_0 of G' is (N_0, T_0, P_0, S_0) with $N_0=\Sigma-\Delta=\{S, B, F\}$, $T_0=P$, $S_0=S$, and $P_0=\{p_1=S \to \pi_1 FB,\ p_2=F \to \pi_2 F,\ p_3=F \to \pi_3,\ p_4=B \to \pi_4\}$. Notice that there is a 1–1 correspondence between the derivation trees of G_0 and those of G.

(2) The semantic domain of G' is $(T(\text{dec}(\Delta)), \text{dec}(\Delta))$.

(3) The attributes of a nonterminal $X \in N_0$ are $1, 2, \ldots, \text{rank}_\Sigma(X)$, one for each tentacle of a hyperedge labeled X. The initial nonterminal has no inherited

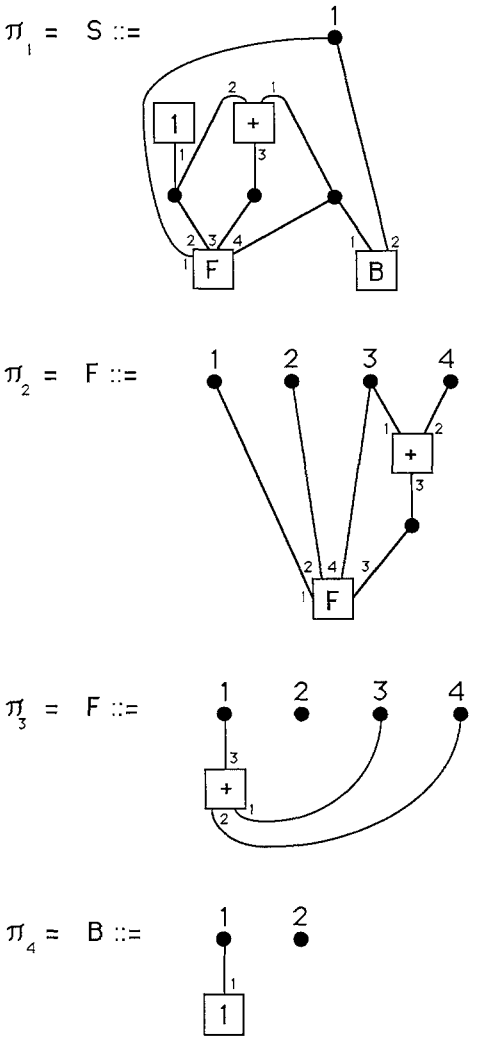

Fig. 15

attributes and the designated attribute α_d of G' is 1. B has inherited attribute 2, and the inherited attributes of F are 2, 3, and 4. Notice that the choice of the attributes is quite obvious, but not their division in inherited and synthesized; this will be explained later.

(4) The semantic rules are as follows.

For $p_1 = S \to \pi_1 F B$, r_{p_1} contains the rules $\langle 1, 0\rangle = \langle 1, 1\rangle$, $\langle 2, 1\rangle = 1$, $\langle 3, 1\rangle = +(\langle 1, 2\rangle, 1)$, $\langle 4, 1\rangle = \langle 1, 2\rangle$, and $\langle 2, 2\rangle = \langle 1, 1\rangle$.

For $p_2 = F \to \pi_2 F$, r_{p_2} contains the rules $\langle 1, 0\rangle = \langle 1, 1\rangle$, $\langle 2, 1\rangle = \langle 2, 0\rangle$, $\langle 3, 1\rangle = +(\langle 3, 0\rangle, \langle 4, 0\rangle)$, and $\langle 4, 1\rangle = \langle 3, 0\rangle$.

For $p_3 = F \to \pi_3$, r_{p_3} contains the rule $\langle 1, 0\rangle = +(\langle 3, 0\rangle, \langle 4, 0\rangle)$.

For $p_4 = B \to \pi_4$, r_{p_4} contains the rule $\langle 1, 0\rangle = 1$.

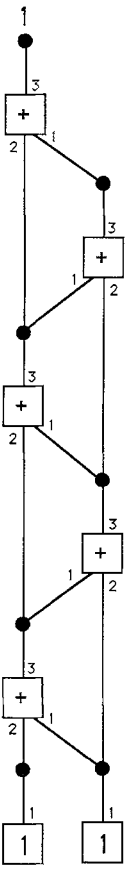

Fig. 16

The way in which the derivation tree of G, that yields the jungle H of Fig. 16, is drawn in Fig. 17, together with its yield H, emphasizes the correspondence between the AG G' and the cfhg G (dashed lines correspond to copy rules of G').

For every production p_i of G', the attribute $\langle \alpha, j \rangle$ (with $j \geqq 1$) represents $\mathrm{nod}(\mathrm{nedg}(\mathrm{rhs}(\pi_i), j), \alpha)$, and the attribute $\langle \alpha, 0 \rangle$ represents $\mathrm{ext}(\alpha)$ of $\mathrm{rhs}(\pi_i)$. Notice that one node in $\mathrm{rhs}(\pi_i)$ can be represented by more than one attribute of p_i. For example, for p_1, $\langle 1, 0 \rangle$ as well as $\langle 1, 1 \rangle$ and $\langle 2, 2 \rangle$ represent $\mathrm{ext}(1)$ of $\mathrm{rhs}(\pi_1)$. We defined the semantic rules of G' in such a way that the value of an attribute $\langle \alpha, x \rangle$ of a derivation tree t' of G_0 equals the term associated with the node that is represented by $\langle \alpha, x \rangle$ in the yield of the corresponding derivation tree t of the cfhg G. In particular, $\mathrm{val}_{G'}(\langle \alpha_d, \mathrm{root}(t') \rangle, t') = \mathrm{term}(\mathrm{yield}_G(t))$. With this intuition it is easy to "read off" these semantic rules from the right-hand sides of the productions of G (resulting in Fig. 17). So, a dashed line between two attributes (corresponding to a copy rule of G') means that they represent the same node in the right-hand side of a production of G.

Figure 17 suggests that it would be more natural to define an AG G' not in Bochmann normal form (consider $\mathrm{nod}_H(e, 2)$ and $\mathrm{nod}_H(e, 3)$ of π_1, where

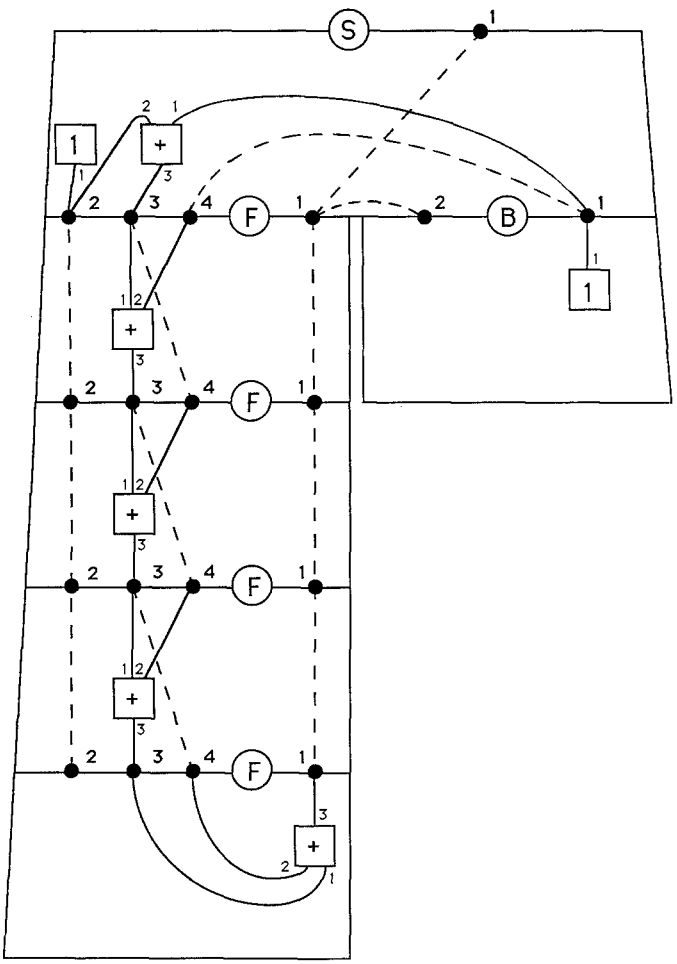

Fig. 17

$H = \mathrm{rhs}(\pi_1)$ and e is the nonterminal hyperedge with $\mathrm{lab}_H(e) = F$). Nevertheless, by the definition of the semantic rules of G', G' indeed is in Bochmann normal form. This is a consequence of the way we "read off" semantic rules from the cfhg productions, which depends on the fact that jungles are one-incoming (we shall return to this later). □

As opposed to Sect. 4, a cfhg G and an AG G' that generate the same term language, can have different "sharings", because an attribute grammar can share (values of) attributes only, whereas a cfhg can share the terms represented by any node in the right-hand side of a production, even by nodes that are incident with terminal hyperedges only. Comparatively, in Sect. 4, attributes are represented by external nodes and nodes incident with nonterminal hyperedges. In the above example, the term 1 in the right-hand sides of the semantic rules $\langle 2, 1\rangle = 1$ and $\langle 3, 1\rangle = +(\langle 1, 2\rangle, 1)$ of p_1 is shared in $\mathrm{rhs}(\pi_1)$. Similarly, G and G' may not have the same "cleanness". In fact, even a term-generating AG

simulating a clean term-generating cfhg does not have to be "clean", where clean for an AG means reduced. For instance, in the above example, the attributes 2 of F and 2 of B are not needed for the computation of the value of the attribute 1 of S.

Despite these differences $\mathrm{TERM}(\mathrm{CFHG}) \subseteq \mathrm{OUT}(\mathrm{AG}, \mathrm{TERMS})$.

Lemma 5.1 *For every term-generating* cfhg G *there exists a term-generating* AG G' *such that* $\mathrm{OUT}(G') = \mathrm{TERM}(G)$.

Proof. Let $G = (\Sigma, \Delta, P, S)$ be a reduced term-generating cfhg. We may assume that G is identification-free by Proposition 2.3 ($\mathrm{rank}_\Sigma(S) = 1$). Consequently, we may assume that G is identification-free and loop-free by Proposition 2.4.

It is easy to choose the underlying grammar G_0 of the non-circular AG G' in such a way that there is a 1–1 correspondence between the derivation trees of G_0 and those of G. We let the nonterminals of G_0 be those of G. The terminals of G_0 are the productions of G, and if $\pi = (X, H)$ is a production of G then $X \to \pi X_1 X_2 \ldots X_m$ is a production of G_0, where $m = \#\mathrm{nedg}(H)$ and $X_j = \mathrm{lab}_H(\mathrm{nedg}(H, j))$ for all $j \in [1, m]$. We add π to the right-hand sides of the productions of G_0 to get a 1–1 correspondence between the productions of G_0 and those of G.

It is obvious that since $\mathrm{TERM}(G) \subseteq T(\mathrm{dec}(\Delta))$, G' must have semantic domain $(T(\mathrm{dec}(\Delta)), \mathrm{dec}(\Delta))$.

One could imagine that a terminal r-hyperedge e in a jungle H "passes" $\mathrm{term}(\mathrm{nod}_H(e, 1), H, ( ))$, $\mathrm{term}(\mathrm{nod}_H(e, 2), H, ( ))$, ..., and $\mathrm{term}(\mathrm{nod}_H(e, r-1), H, ( ))$ to $\mathrm{nod}_H(e, r)$, which "uses" these values (together with $\mathrm{lab}_H(e)$) to construct $\mathrm{term}(\mathrm{nod}_H(e, r), H, ( ))$. The "passing" of values in an AG is determined by the semantic rules. For that reason we let attributes correspond to nodes in a jungle and semantic rules to "pieces of jungle" consisting of terminal hyperedges.

We give each nonterminal $X \in \Sigma - \Delta$ the attributes $1, 2, \ldots, \mathrm{rank}_\Sigma(X)$, one for each tentacle of a hyperedge labeled X. Now we have to determine which are its inherited and which are its synthesized attributes.

Consider a derivation subtree t' of a derivation tree t of G, that is associated with the derivation $X \Rightarrow^* F \in \mathrm{HGR}(\Delta)$. Since G is identification-free, loop-free, and reduced, F is (isomorphic with) an identification-free subgraph of the hypergraph $\mathrm{yield}(t)$ in $L(G)$. Since G is term-generating, $\mathrm{yield}(t)$ is a jungle. Hence, it can easily be seen that F is a jungle with k variables, for some $k \in [0, \mathrm{rank}_\Sigma(X)]$, provided we permute the sequence of external nodes of F in such a way that the first k external nodes have no incoming hyperedge in F and the remaining ones have an incoming hyperedge in F (as already observed in Sect. 3). It is natural to view these first k external nodes as inherited attributes of X, because the terms associated with their corresponding nodes in $\mathrm{yield}(t)$ are initially determined by the context of t', as in AG's. The remaining external nodes have one incoming hyperedge in F, i.e., the associated terms of their corresponding nodes in $\mathrm{yield}(t)$ are initially determined by t'. Thus it is natural to view them as synthesized attributes of X.

Hence, we define the set of inherited attributes of a nonterminal X as $\mathrm{Inh}(X) = \{i \in [1, \mathrm{rank}_\Sigma(X)] \mid \mathrm{ext}_F(i)$ has no incoming hyperedge in $F\}$, where $F \in \mathrm{HGR}(\Delta)$ is such that $X \Rightarrow^* F$ in G (and thus $\mathrm{Syn}(X) = \{i \in [1, \mathrm{rank}_\Sigma(X)] \mid i \notin \mathrm{Inh}(X)\}$). This set is (independently of the choice of F) uniquely defined for every nonterminal X. In fact, consider derivation trees t and t' of G, where t' is obtained from t by replacing a derivation subtree

that corresponds to $X \Rightarrow^* F$ by a derivation subtree that corresponds to $X \Rightarrow^* F'$. By the context-freeness of cfhg's, there exists a hypergraph $K \in \mathrm{HGR}(\Sigma)$ with exactly one nonterminal hyperedge e (labeled X), such that $\mathrm{yield}(\ell) = K[e/F]$ and $\mathrm{yield}(\ell') = K[e/F']$. Note that K is loop-free, and F, F' are identification-free. Since G is term-generating, every node in $\mathrm{yield}(\ell)$ and $\mathrm{yield}(\ell')$ has one incoming hyperedge. Thus, an external node $\mathrm{ext}_F(i)$ has no incoming hyperedge in F iff $\mathrm{ext}_{F'}(i)$ has no incoming hyperedge in F' (iff $\mathrm{nod}_K(e, i)$ has an incoming hyperedge in K). As an example, the reader may now see from Fig. 17 that $\mathrm{Inh}(F) = \{2, 3, 4\}$ and $\mathrm{Inh}(B) = \{2\}$. As another example, for the nonterminal A in Fig. 6, $\mathrm{Inh}(A) = \emptyset$, i.e., A has two synthesized attributes.

For technical reasons, we would indeed like to assume that the yield of every derivation subtree of G is a jungle with variables. Notice that the tentacles of a nonterminal hyperedge e labeled X in the right-hand side H of a production (X, H) of G can easily be permuted such that $\mathrm{nod}_H(e, 1), \mathrm{nod}_H(e, 2), \ldots, \mathrm{nod}_H(e, \#\mathrm{Inh}(X))$ are represented by the inherited attributes and $\mathrm{nod}_H(e, \#\mathrm{Inh}(X)+1), \ldots, \mathrm{nod}_H(e, \mathrm{rank}(e))$ are represented by the synthesized attributes of X. The external nodes of the right-hand sides of the productions that can be applied to e (i.e., productions π with $\mathrm{lhs}(\pi) = X$) can, of course, be permuted in the same way. So, from now on we assume that $\mathrm{Inh}(X) = [1, \#\mathrm{Inh}(X)]$ and $\mathrm{Syn}(X) = [\#\mathrm{Inh}(X)+1, \mathrm{rank}_\Sigma(X)]$, for every $X \in \Sigma - \Delta$. For example, if the tentacles of the nonterminal hyperedge labeled F in $\mathrm{rhs}(\pi_1)$ of Fig. 15 would be permuted in such a way, then $\mathrm{Inh}(F)$ would be $\{1, 2, 3\}$ instead of $\{2, 3, 4\}$.

To define the semantic rules of the productions in G_0, we use a kind of unfolding operation. This operation is based on the fact that jungles are one-incoming.

Consider a derivation tree t of G. Let x be a node in t with label $\pi = (X, H)$. Let $\mathrm{lab}_H(\mathrm{nedg}(H, j)) = X_j$ for all $j \in [1, m]$, where $m = \#\mathrm{nedg}(H)$. Each node $v \in V_H$ has one incoming hyperedge in $\mathrm{yield}(\ell)$. This incoming hyperedge e is "related to" H according to one of the following three mutually exclusive, exhaustive cases.

(i) e corresponds to a (terminal) hyperedge of H.
(ii) $v = \mathrm{nod}_H(\mathrm{nedg}(H, j), i)$ and e corresponds to an incoming hyperedge of $\mathrm{ext}(i)$ in $\mathrm{yield}(\ell_j)$, where ℓ_j is the derivation subtree of ℓ such that $\mathrm{root}(\ell_j)$ is the j-th son of x, for some $j \in [1, m]$. Note that this j and i are unique.
(iii) For e, neither case (i) nor case (ii) holds. In this case $v = \mathrm{ext}_H(i)$ for some unique i.

Notice that cases (ii) and (iii) correspond to the facts that $i \in \mathrm{Syn}(X_j)$ and $i \in \mathrm{Inh}(X)$, respectively. Thus, $\mathrm{incoming}(v) = 1$, for all $v \in V_H$, where $\mathrm{incoming}(v)$ is defined as $\#\{e \in \mathrm{tedg}(H) \mid \mathrm{nod}_H(e, \mathrm{rank}(e)) = v\} + \#\{(j, i) \mid 1 \leqq j \leqq m,\ 1 \leqq i \leqq \mathrm{rank}(X_j),\ i \in \mathrm{Syn}(X_j),\ \mathrm{nod}_H(\mathrm{nedg}(H, j), i) = v\} + \#\{i \in \mathrm{Inh}(X) \mid \mathrm{ext}_H(i) = v\}$. This will be used in the definition of the unfolding operation that determines the set of semantic rules of the production $p = X \to \pi X_1 X_2 \ldots X_m$.

The formal definition of the AG G' is now as follows.

(1) The underlying cfg G_0 is (N_0, T_0, P_0, S_0) where $N_0 = \Sigma - \Delta$, $T_0 = P$, $S_0 = S$, and

$$P_0 = \{X \to \pi X_1 X_2 \ldots X_m \mid \exists H: \pi = (X, H) \in P, m = \#\mathrm{nedg}(H), X_j = \mathrm{lab}_H(\mathrm{nedg}(H, j)) \text{ for all } j \in [1, m]\}.$$

(2) The semantic domain is $(T(\mathrm{dec}(\Delta)), \mathrm{dec}(\Delta))$.

(3) For every $X \in N_0$, $\mathrm{Inh}(X) = \{i \in [1, \mathrm{rank}_\Sigma(X)] \mid \mathrm{ext}_F(i)$ has no incoming hyperedge in $F\}$, where $F \in \mathrm{HGR}(\Delta)$ is such that $X \Rightarrow^* F$ in G (i.e., $\mathrm{Inh}(X) = [1, \#\mathrm{Inh}(X)]$, because we assume that every derivation subtree of G yields a jungle with variables), and $\mathrm{Syn}(X) = [\#\mathrm{Inh}(X)+1, \mathrm{rank}_\Sigma(X)]$. The designated attribute α_d is 1.
(4) For every $p = X \to \pi X_1 X_2 \ldots X_m \in P_0$ with $\pi = (X, H)$, r_p contains the semantic rules $\langle s, 0\rangle = \mathrm{rule}(\mathrm{ext}_H(s), H)$ for all $s \in \mathrm{Syn}(X)$, and $\langle i, j\rangle = \mathrm{rule}(\mathrm{nod}_H(\mathrm{nedg}(H, j), i), H)$ for all $i \in \mathrm{Inh}(X_j)$ and $j \in [1, m]$, where $\mathrm{rule}(v, H)$ is recursively defined for $v \in V_H$ as follows (it is the unfolding operation meant above).

If $v = \mathrm{ext}_H(i)$ for some $i \in \mathrm{Inh}(X)$, then $\mathrm{rule}(v, H) = \langle i, 0\rangle$.
If $v = \mathrm{nod}_H(e, s)$ for some $e = \mathrm{nedg}(H, j)$ $(1 \leqq j \leqq m)$ and $s \in \mathrm{Syn}(\mathrm{lab}_H(e))$, then $\mathrm{rule}(v, H) = \langle s, j\rangle$.
If $v = \mathrm{nod}_H(e, 1)$ for some 1-hyperedge $e \in \mathrm{tedg}(H)$, then $\mathrm{rule}(v, H) = \mathrm{lab}_H(e)$.
If $v = \mathrm{nod}_H(e, r)$ for some r-hyperedge $e \in \mathrm{tedg}(H)$ with $r \geqq 2$, then

$$\begin{aligned}\mathrm{rule}(v, H) = \mathrm{lab}_H(e)(&\mathrm{rule}(\mathrm{nod}_H(e, 1), H),\\ &\mathrm{rule}(\mathrm{nod}_H(e, 2), H),\\ &\vdots\\ &\mathrm{rule}(\mathrm{nod}_H(e, r-1), H)).\end{aligned}$$

Notice that this definition of "rule" corresponds to the way we "read off" the semantic rules in the example at the beginning of this section.

The terminal part of H is a subgraph of a generated jungle, because G is identification-free, loop-free, and reduced. Thus, the terminal part of H is acyclic. Since besides, $\mathrm{incoming}(v) = 1$ for all $v \in V_H$ and H is finite, $\mathrm{rule}(v, H)$ is determined uniquely for each $v \in V_H$. Furthermore, since $\mathrm{rule}(v, H) \in T(\mathrm{dec}(\Delta), \mathrm{outs}(p))$, G' is in Bochmann normal form (as required).

To show the correctness of the above construction, we shall discuss that G' is a non-circular AG and that $\mathrm{OUT}(G') = \mathrm{TERM}(G)$.

Consider the dependency graph of a derivation tree t' of G_0. Every node $\langle \alpha, x\rangle$ of $DG_{t'}$ represents a node $v_{\alpha,x}$ of $\mathrm{yield}_G(t)$, where t is the derivation tree of G that corresponds to t'. Besides, according to the definition of "rule", an edge from $\langle \beta, y\rangle$ to $\langle \alpha, x\rangle$ in $DG_{t'}$ represents the existence of a hyperpath (possibly of length 0) from $v_{\beta,y}$ to $v_{\alpha,x}$ in $\mathrm{yield}_G(t)$ (in fact, in the terminal part of the right-hand side of the cfhg production that corresponds to the production of G_0 of which a semantic rule "caused" the edge in $DG_{t'}$). Thus, a cycle in $DG_{t'}$ either represents a cycle in $\mathrm{yield}_G(t)$ or it represents just one node in $\mathrm{yield}_G(t)$ (i.e., all nodes of the cycle represent the same node in $\mathrm{yield}_G(t)$). The first case cannot occur, because jungles are acyclic. In the latter case, the whole cycle in $DG_{t'}$ must "consist of" copy rules. But, since G is identification-free, there can be no semantic rule of the form $\langle s, 0\rangle = \langle i, 0\rangle$ in r_p, for any production p of G_0, where $s \in \mathrm{Syn}(\mathrm{lhs}(p))$ and $i \in \mathrm{Inh}(\mathrm{lhs}(p))$. However, such a copy rule must always be present in such a cycle. Hence, G' is a non-circular AG.

To show that $\mathrm{OUT}(G') = \mathrm{TERM}(G)$, we use the following claim, where, as in the proof of Lemma 4.1, the value of an attribute α of a node x in a derivation subtree t of G_0 is denoted as $\mathrm{val}_{G'}(\langle \alpha, x\rangle, t)$.

Claim. Let t be a derivation subtree of G of which the root is labeled $\pi=(X, H)$. Let t' be the corresponding derivation subtree of G_0 with root x'. For every $s\in\mathrm{Syn}(X)$,

$$\mathrm{val}_{G'}(\langle s, x'\rangle, t')=\mathrm{term}(\mathrm{ext}(s), \mathrm{yield}_G(t), Y),$$

where $Y=(\langle 1, x'\rangle, \langle 2, x'\rangle, \ldots, \langle \#\mathrm{Inh}(X), x'\rangle)$.

As in Sect. 4, we omit the formal proof of this claim. It can be proved with induction on the structure of t (or, equivalently, t'). Let us just make some informal remarks on the proof.

Since Y is a sequence of $\#\mathrm{Inh}(X)$ variables and $\mathrm{yield}_G(t)=F$ is a jungle with $\#\mathrm{Inh}(X)$ variables, $\mathrm{term}(v, F, Y)\in T(\mathrm{dec}(\Delta), Y)$ for each $v\in V_F$.

$\mathrm{Root}(t')$ has label $p=X\to\pi X_1 X_2 \ldots X_m$, where $m=\#\mathrm{nedg}(H)$ and $X_j=\mathrm{lab}_H(\mathrm{nedg}(H, j))$, for all $j\in[1, m]$. The value of an attribute $\langle\alpha, j\rangle\in\mathrm{ins}(p)$ is determined with the help of the operation "rule". Thus, for $j=0$, $\mathrm{val}_{G'}(\langle\alpha, x'\rangle, t')$ equals $\mathrm{rule}(\mathrm{ext}_H(\alpha), H)$, in which every outside attribute $\langle\beta, i\rangle$ is replaced by $\mathrm{val}_{G'}(\langle\beta, x'\rangle, t')=\langle\beta, x'\rangle$, if $i=0$, and by $\mathrm{val}_{G'}(\langle\beta, x'_i\rangle, t')$, where x'_i denotes the i-th son of x', if $i>0$. Similarly, for $j>0$, $\mathrm{val}_{G'}(\langle\alpha, x'_j\rangle, t')$, where x'_j denotes the j-th son of x', equals $\mathrm{rule}(\mathrm{nod}_H(\mathrm{nedg}(H, j), \alpha), H)$, with the same replacement. Using the induction hypothesis (for the subtrees of t' with roots x'_i), it can be shown that, for every $v\in V_H$, $\mathrm{rule}(v, H)$, in which the outside attributes are replaced by the values they have in t' as indicated above, equals $\mathrm{term}(w, F, Y)$, where w is the node corresponding to v in F, because "rule" is the "term-operation" for right-hand sides of productions of G. Thus, $\mathrm{val}_{G'}(\langle\alpha, x'\rangle, t')$ equals $\mathrm{term}(\mathrm{ext}_F(\alpha), F, Y)$.

This ends our discussion of the proof of the claim. Now, let t be a derivation tree of G. Let t' be the corresponding derivation tree of G_0. Then, according to the claim,

$$\mathrm{val}_{G'}(\langle\alpha_d, \mathrm{root}(t')\rangle, t')=\mathrm{term}(\mathrm{ext}(1), \mathrm{yield}_G(t), ( ))=\mathrm{term}(\mathrm{yield}_G(t)).$$

Thus, because of the 1–1 correspondence between the derivation trees of G and G_0, $\mathrm{OUT}(G')=\mathrm{TERM}(G)$. Note that the construction in this proof takes exponential time in general, due to the unfolding in "rule". □

From this result (Lemma 5.1) and Lemma 4.1 we conclude that cfhg's and AG's have the same term-generating power.

Theorem 5.2 $\mathrm{TERM}(\mathrm{CFHG})=\mathrm{OUT}(\mathrm{AG}, \mathrm{TERMS})$.

As observed in the Introduction, this positions TERM(CFHG) in a family of known classes of term (or tree) languages.

Furthermore, Lemma 5.1 and Lemma 4.1 are now used to prove the "garbage theorem" that provides an alternative way of defining TERM(CFHG) (as observed in Sect. 3).

Theorem 5.3. *For every term-generating cfhg G there exists a clean term-generating* cfhg G' *such that* $\mathrm{TERM}(G')=\mathrm{TERM}(G)$.

Proof. Let G be a term-generating cfhg. Then by Lemma 5.1 there exists a term-generating AG G_1 such that $\mathrm{OUT}(G_1)=\mathrm{TERM}(G)$. By Proposition 2.1 there exists a reduced AG G_2 over the same semantic domain as G_1 such that

$\mathrm{OUT}(G_2)=\mathrm{OUT}(G_1)$. Hence G_2 is term-generating. Thus, by Lemma 4.1 there exists a clean term-generating cfhg G' such that $\mathrm{TERM}(G')=\mathrm{OUT}(G_2)=\mathrm{OUT}(G_1)=\mathrm{TERM}(G)$. □

6. The translation power of context-free hypergraph grammars

Recall from the end of Sect. 4 that a term-generating cfhg G, together with a semantic domain D, can be viewed as a syntax-directed translation device realizing the translation $\{(\xi, (\mathrm{term}(\mathrm{yield}_G(\xi)))_D) \mid \xi \text{ is a derivation tree of } G\}$. To compare the translation power of cfhg's with that of AG's, we introduce "cfhg-based" syntax-directed translation schemes, that translate strings (rather than trees) to values of the semantic domain. They are similar to the syntax-directed translation schemes of [AhoUll] and the string-to-graph translators (the "pair grammars") of [Pra].

Definition. A cfhg-*based syntax-directed translation scheme* (abbreviated cts) is a 4-tuple $\mathbf{T}=(D, GL, GR, \varphi)$, where $D=(V, \Gamma)$ is a semantic domain, $GL=(N, T, P_{GL}, S)$ is a cfg, the left grammar, $GR=(\Sigma, \mathrm{inc}(\Gamma), P_{GR}, S)$ is a term-generating cfhg, the right grammar, with $\Sigma-\mathrm{inc}(\Gamma)=N$, and φ is a mapping from P_{GL} to P_{GR} such that if $p=X_0 \to w_0 X_1 w_1 X_2 w_2 \ldots X_m w_m \in P_{GL}$ and $\varphi(p)=(X, H)\in P_{GR}$, then $X_0=X$, $m=\#\mathrm{nedg}(H)$, and $X_j=\mathrm{lab}_H(\mathrm{nedg}(H, j))$, for all $j\in[1, m]$. □

Notice that this kind of syntax-directed translation device is "simple" in the sense of [AhoUll], i.e., nonterminals of p cannot be deleted or duplicated in $\varphi(p)$.

For a production $p\in P_{GL}$, its corresponding production in P_{GR} is $\varphi(p)$. Similarly, for a derivation tree ξ of GL, its corresponding derivation tree of GR, denoted $\varphi(\xi)$, is obtained by relabeling the nodes of ξ with their corresponding productions. Thus, φ is extended to derivation trees.

$\mathbf{T}$ realizes a translation from strings in $L(GL)$ to values in D. For $w\in L(GL)$, a derivation tree ξ with $\mathrm{yield}_{GL}(\xi)=w$ is considered and the value (in D) of the term associated with the jungle that is the yield of the corresponding derivation tree $\varphi(\xi)$ of GR, is assigned to w.

Definition. Let $\mathbf{T}=(D, GL, GR, \varphi)$ be a cts. The *translation realized by* $\mathbf{T}$, denoted $\tau(\mathbf{T})$, is the relation $\{(\mathrm{yield}_{GL}(\xi), (\mathrm{term}(\mathrm{yield}_{GR}(\varphi(\xi))))_D) \mid \xi \text{ is a derivation tree of } GL\}$. □

The set of all translations realized by cts's with semantic domain D is denoted $\tau(\mathrm{CTS}, D)$.

Dropping the useless nonterminals (and productions) of both the right and the left grammar of a cts does not influence the realized translation, because they do not occur in derivation trees. Thus, we may assume both grammars to be reduced.

Example. Consider the cts $\mathbf{T}=(D, GL, GR, \varphi)$, where $D=(V, \Gamma)$ is the semantic domain of the AG G_{bin} of Example 2.2, $GL=(N_0, T_0, P_0, S_0)$ is the underlying grammar G_0 of G_{bin}, $GR=(N_0\cup\mathrm{inc}(\Gamma), \mathrm{inc}(\Gamma), P, S_0)$ is the term-generating cfhg of which the productions are given in Fig. 8, and $\varphi: P_0 \to P$ is defined as $\varphi(p_i)=\pi_i$, $1\leqq i\leqq 6$, as in the discussion at the end of Sect. 4, where GR is called G''.

The translation realized by $\mathbf{T}$ is $\tau(\mathbf{T}) = \{(\text{yield}_{G_0}(t), (\text{term}(\text{yield}_{G''}(\varphi(t))))_D) \mid t$ is a derivation tree of $G_0\}$. Thus, to each binary number in $L(GL)$ its rational value (in V) is assigned by $\mathbf{T}$, just like the translation realized by the AG G_{bin} (see also Sect. 4). Hence $\tau(\mathbf{T}) = \tau(G_{\text{bin}})$. □

We shall prove that for every cts there is an AG that realizes the same translation, and vice versa. More precisely, we show that $\tau(\text{CTS}, D) = \tau(\text{AG}, D)$ for every semantic domain D (Theorem 6.5). First we show that $\tau(\text{AG}, D) \subseteq \tau(\text{CTS}, D)$ for every semantic domain D.

Lemma 6.1 *For every* AG G *over a semantic domain* D, *there exists a* cts $\mathbf{T}$ *with semantic domain* D *such that* $\tau(\mathbf{T}) = \tau(G)$. *Moreover, if* G *is reduced, then the right grammar of* $\mathbf{T}$ *is clean term-generating.*

Proof. Let G be an AG over a semantic domain D with underlying grammar G_0. Recall from Sect. 2.2 that, to compute the value of the designated attribute α_d of the root of a derivation tree t of G_0, one may first evaluate the attributes as terms in $T(I)$ (by G_{term}), and then evaluate the term-value of $\langle \alpha_d, \text{root}(t) \rangle$ in D. This resembles the way in which the value of a jungle H generated by the right grammar of a cts with semantic domain D is computed; first determine the term associated with H, and then evaluate this term in D.

By Lemma 4.1 there exists a term-generating cfhg G'_{term} such that $\text{TERM}(G'_{\text{term}}) = \text{OUT}(G_{\text{term}})$ (and such that G'_{term} is clean term-generating if G is reduced). In the discussion at the end of Sect. 4, we showed that the translation $\tau(G)$ realized by the AG G equals $\{(\text{yield}(t), (\text{term}(\text{yield}_{G''_{\text{term}}}(\varphi(t))))_D) \mid t$ is a derivation tree of $G_0\}$, where G''_{term} is a term-generating cfhg that is equivalent to G'_{term} (hence G''_{term} is clean term-generating if G is reduced), and φ is a mapping from the productions of G_0 to those of G''_{term}. Since φ preserves nonterminals, $\mathbf{T} = (D, G_0, G''_{\text{term}}, \varphi)$ satisfies the definition of cts. Thus, $\tau(\mathbf{T}) = \tau(G)$. □

To show the converse of Lemma 6.1, we need two auxiliary lemma's concerning the translations realized by cts's. These lemma's allow the generalization of results for cfhg's to similar results for cts's.

The first lemma says that, maintaining the realized translation, we may change the right-hand sides of the productions of the right grammar of a cts arbitrarily, as long as the sequences of the labels of the nonterminal hyperedges in these right-hand sides are preserved. Thus, even nodes and (terminal) hyperedges may be added to or deleted from the right-hand sides of the productions. Additionally, however, the terms associated with the yields of the derivation trees should be preserved.

Lemma 6.2 *Let* $\mathbf{T} = (D, GL, GR, \varphi)$ *be a* cts *with* $GR = (\Sigma, \Delta, P_{GR}, S)$. *Let* $GR' = (\Sigma, \Delta, P'_{GR}, S)$ *be a term-generating* cfhg. *If there exists a mapping* $\chi\colon P_{GR} \to P'_{GR}$ *such that*

(1) for every production $\pi = (X, H) \in P_{GR}$, $\text{lhs}(\chi(\pi)) = X$, $\#\text{nedg}(\text{rhs}(\chi(\pi))) = \#\text{nedg}(H) = m$ *and* $\text{lab}_{\text{rhs}(\chi(\pi))}(\text{nedg}(\text{rhs}(\chi(\pi)), j)) = \text{lab}_H(\text{nedg}(H, j))$ *for all* $j \in [1, m]$, *and*

(2) for every derivation tree t *of* GR, $\text{term}(\text{yield}_{GR'}(\chi(t))) = \text{term}(\text{yield}_{GR}(t))$, *where* χ *is extended to derivation trees in the obvious way,*

then there exists a cts $\mathbf{T}'$ *with semantic domain* D *and right grammar* GR' *such that* $\tau(\mathbf{T}') = \tau(\mathbf{T})$.

Proof. Let $GL=(\Sigma-\Delta, T, P_{GL}, S)$. Since χ is a mapping from P_{GR} to P'_{GR} that preserves the sequences of nonterminal labels, i.e., χ has property (1), and since φ is a mapping from P_{GL} to P_{GR} that also preserves these sequences, $\mathbf{T}'=(D, GL, GR', \chi\circ\varphi)$ is a cts. Furthermore, $\tau(\mathbf{T}')=\{(\text{yield}_{GL}(t), (\text{term}(\text{yield}_{GR'}(\chi(\varphi(t)))))_D)\mid t \text{ is a derivation tree of } GL\}$. Thus, because of (2), $\tau(\mathbf{T}')=\{(\text{yield}_{GL}(t), (\text{term}(\text{yield}_{GR}(\varphi(t))))_D)\mid t \text{ is a derivation tree of } GL\}=\tau(\mathbf{T})$. □

The second lemma states that we can add information to the nonterminal labels of the right grammar of a cts without affecting the translation it realizes (with an appropriate change of the left grammar).

Lemma 6.3 *Let* $\mathbf{T}=(D, GL, GR, \varphi)$ *be a* cts *with* $GR=(\Sigma, \Delta, P_{GR}, S)$. *Let* $GR'=(\Sigma, \Delta, P'_{GR}, S')$ *be a term-generating* cfhg. *If there exists a mapping* $\psi: \Sigma'-\Delta\to\Sigma-\Delta$ *such that*

(1) for every $\pi'=(Y, K)\in P'_{GR}$, *there exists a production* $\pi\in P_{GR}$ *with* $\psi(\pi')=\pi$, *i.e.,* $\text{lhs}(\pi)=\psi(Y)$ *and* $\text{rhs}(\pi)=(V_K, E_K, \text{nod}_K, \text{lab}, \text{ext}_K)$ *where*

$$\text{lab}(e)=\begin{cases}\text{lab}_K(e) & \text{for all } e\in\text{tedg}(K)\\ \psi(\text{lab}_K(e)) & \text{for all } e\in\text{nedg}(K), \text{ and}\end{cases}$$

(2) for every derivation tree t *of* GR, *there exists a derivation tree* t' *of* GR' *such that* $\psi(t')=t$, *where* ψ *is extended to (productions and) derivation trees in the obvious way,*

then there exists a cts $\mathbf{T}'$ *with semantic domain* D *and right grammar* GR' *such that* $\tau(\mathbf{T}')=\tau(\mathbf{T})$.

Proof. Let $GL=(\Sigma-\Delta, T, P_{GL}, S)$. We shall construct a cts $\mathbf{T}'=(D, GL', GR', \eta)$ such that $\tau(\mathbf{T}')=\tau(\mathbf{T})$. For that purpose, $L(GL')$ must be $L(GL)$. Thus, GL' must have the same terminals as GL. Besides, GL' must have the same nonterminals and initial nonterminal as GR'. Thus, $GL'=(\Sigma'-\Delta, T, P'_{GL}, S')$ for some P'_{GL}.

The idea for the construction of P'_{GL} is obvious. If a production π of GR is turned into a production π' of GR' by a relabeling of the nonterminals, then every production p of GL with $\varphi(p)=\pi$ is turned into a production p' of GL' by the same relabeling. Hence, we can extend ψ also to a mapping from the productions (and derivation trees) of GL' to those of GL. This can be illustrated by

$$\begin{array}{ccc} p'\in P'_{GL} & \xrightarrow{\eta} & \pi'\in P'_{GR}\\ \downarrow\psi & & \downarrow\psi\\ p\in P_{GL} & \xrightarrow{\varphi} & \pi\in P_{GR}. \end{array}$$

Formally, P'_{GL} and η are constructed as follows.

Let $\pi'=(X', H')$ be a production of GR', with $\#\text{nedg}(H')=m$, and $X'_j=\text{lab}_{H'}(\text{nedg}(H', j))$ for all $j\in[1, m]$. Let $p=X_0\to w_0 X_1 w_1 X_2 w_2 \dots X_m w_m\in P_{GL}$ be a production in $\varphi^{-1}(\psi(\pi'))$. Thus, $X_0=\text{lhs}(\psi(\pi'))=\psi(X')$ and $X_j=\text{lab}_{\text{rhs}(\psi(\pi'))}(\text{nedg}(\text{rhs}(\psi(\pi')), j))=\psi(X'_j)$.

Then P'_{GL} contains the production $p' = X' \to w_0 X'_1 w_1 X'_2 w_2 \dots X'_m w_m$ and $\eta(p') = \pi'$.

It is easy to see that η is a mapping from P'_{GL} to P'_{GR} such that, for every $p' \in P'_{GL}$, $\mathrm{lhs}(p') = \mathrm{lhs}(\eta(p'))$ and the sequences of nonterminals of $\mathrm{rhs}(p')$ and $\mathrm{rhs}(\eta(p'))$ are the same. Hence, $\mathbf{T}'$ indeed is a cts.

The translation realized by $\mathbf{T}'$ is $\tau(\mathbf{T}') = \{(\mathrm{yield}_{GL'}(\delta), (\mathrm{term}(\mathrm{yield}_{GR'}(\eta(\delta)))_D) \mid \delta$ is a derivation tree of $GL'\}$. Since $\mathrm{yield}_{GL'}(\delta) = \mathrm{yield}_{GL}(\psi(\delta))$, for every derivation tree δ of GL', and $\mathrm{yield}_{GR'}(\delta) = \mathrm{yield}_{GR}(\psi(\delta))$, for every derivation tree δ of GR', $\tau(\mathbf{T}') = \{(\mathrm{yield}_{GL}(\psi(\delta)), (\mathrm{term}(\mathrm{yield}_{GR}(\psi(\eta(\delta))))_D) \mid \delta$ is a derivation tree of $GL'\}$. Now, since η and GL' are constructed such that $\psi(\eta(\delta)) = \varphi(\psi(\delta))$, for every derivation tree δ of GL', the latter relation equals $\{(\mathrm{yield}_{GL}(\psi(\delta)), (\mathrm{term}(\mathrm{yield}_{GR}(\varphi(\psi(\delta))))_D) \mid \delta$ is a derivation tree of $GL'\}$. From (2) it follows that, for every derivation tree δ of GL, there exists a derivation tree δ' of GR' such that $\delta \in \varphi^{-1}(\psi(\delta'))$. Hence, by the construction of P'_{GL}, for every derivation tree δ of GL there exists a derivation tree δ'' of GL' such that $\delta = \psi(\delta'')$. Thus, $\tau(\mathbf{T}') = \{(\mathrm{yield}_{GL}(\delta), (\mathrm{term}(\mathrm{yield}_{GR}(\varphi(\delta)))_D) \mid \delta$ is a derivation tree of $GL\} = \tau(\mathbf{T})$. □

Often, in applications of Lemma 6.3, the nonterminal alphabet of GR' is an extension of the nonterminal alphabet of GR, i.e., $\Sigma' - \Delta$ is of the form $(\Sigma - \Delta) \times I$, for some set I, and ψ is the projection on $\Sigma - \Delta$.

Example. For every cts $\mathbf{T} = (D, GL, GR, \varphi)$, where GR is identification-free, there exists a cts $\mathbf{T}'$ with semantic domain D such that (i) the right grammar of $\mathbf{T}'$ is identification-free and loop-free, and (ii) $\tau(\mathbf{T}') = \tau(\mathbf{T})$.

This is based on the construction in the proof of Proposition 2.4 [Hab, Theorem I.4.6] applied to GR. In fact, that construction may be viewed as consisting of two consecutive transformations, viz. a relabeling of the nonterminal hyperedges such that the labels contain information about which tentacles point to the same node (in a sentential form), and the joining of the tentacles and identification of the external nodes in the right-hand sides of the productions in a unique way, according to the just mentioned relabeling. In fact, these two transformations are combined into one (with three steps) in the proof in [Hab]. Lemma's 6.3 and 6.2, respectively, can be used to show the above for cts's.

To be more precise, let GR be the identification-free term-generating cfhg (Σ, Δ, P, S) and consider the cfhg $GR_1 = (\Sigma_1, \Delta, P_1, S_1)$ obtained from GR by relabeling the nonterminals in the following way. The nonterminal alphabet Σ_N of GR_1 is $\{\langle X, \mathrm{REL}\rangle \mid X \in \Sigma - \Delta$, REL is an equivalence relation on $[1, \mathrm{rank}_\Sigma(X)]\}$. The total alphabet Σ_1 is $\Sigma_N \cup \Delta$, where

$$\mathrm{rank}(\sigma)_{\Sigma_1} = \begin{cases} \mathrm{rank}_\Sigma(\sigma) & \text{if } \sigma \in \Delta \\ \mathrm{rank}_\Sigma(X) & \text{if } \sigma = \langle X, \mathrm{REL}\rangle \in \Sigma_N, \text{ for some REL.} \end{cases}$$

The initial nonterminal S_1 is $\langle S, \{(1, 1)\}\rangle$. If (X, H) is a production of P and REL is an equivalence relation on $[1, \mathrm{rank}_\Sigma(X)]$, then P_1 contains the production $(\langle X, \mathrm{REL}\rangle, K)$, where $K = (V_H, E_H, \mathrm{nod}_H, \mathrm{lab}, \mathrm{ext}_H)$ with $\mathrm{lab}(e) = \mathrm{lab}_H(e)$ for all $e \in \mathrm{tedg}(H)$, and, for all $e \in \mathrm{nedg}(H)$, $\mathrm{lab}(e)$ is $\langle \mathrm{lab}_H(e), \mathrm{REL}(e)\rangle$, where $\mathrm{REL}(e)$ is defined as follows. By $\underline{H}$ we denote the hypergraph in which the external nodes of H are identified according to REL, i.e.,

$$\underline{H} = H / \{(\mathrm{ext}_H(i), \mathrm{ext}_H(j)) \mid (i, j) \in \mathrm{REL}\}.$$

Then, for all $e \in \text{nedg}(\underline{H})$,

$$\text{REL}(e) = \{(i, j) \in [1, \text{rank}(e)]^2 \mid \text{nod}_{\underline{H}}(e, i) = \text{nod}_{\underline{H}}(e, j)\}.$$

Note that GR_1 is still identification-free. It is straightforward to show that the mapping $\psi: \Sigma_N \to \Sigma - \Delta$ defined as $\psi(\langle X, \text{REL}\rangle) = X$, for all $\langle X, \text{REL}\rangle \in \Sigma_N$, satisfies the conditions (1) and (2) of Lemma 6.3 (in (2), t' can be obtained from t by a deterministic top-down relabeling of the nonterminals). From these conditions it follows that $L(GR_1) = L(GR)$, and thus GR_1 is term-generating. Hence, by Lemma 6.3, there exists a cts $\mathbf{T}_1$ with semantic domain D and right grammar GR_1 such that $\tau(\mathbf{T}_1) = \tau(\mathbf{T})$.

To join the tentacles and to identify the external nodes in the right-hand sides of the productions of G_1 in the way described in [Hab], we define for every nonterminal $\sigma = \langle X, \text{REL}\rangle \in \Sigma_N$ the sequence $\text{EQ}(\sigma)$ obtained by ordering the set $\{j \in [1, \text{rank}_{\Sigma_1}(\sigma)] \mid \forall\, i \in [1, j-1]: (i, j) \notin \text{REL}\}$ increasingly.

Now consider the cfhg $GR' = (\Sigma', \Delta, P', S_1)$, where $\Sigma' = \Sigma_1\ (= \Sigma_N \cup \Delta)$ with

$$\text{rank}_{\Sigma'}(\sigma) = \begin{cases} \text{rank}_{\Sigma_1}(\sigma) & \text{if } \sigma \in \Delta \\ \#\text{EQ}(\sigma) & \text{if } \Sigma \in \Sigma_N, \end{cases}$$

and P' is constructed from P_1 as follows.

Let $\pi_1 = (\langle X, \text{REL}\rangle, K) \in P_1$. By $\underline{K}$ we denote the hypergraph in which the external nodes of K are identified according to REL, i.e., $\underline{K} = K/\{(\text{ext}_K(i), \text{ext}_K(j)) \mid (i, j) \in \text{REL}\}$.

Then P' contains the production $\pi' = (\langle X, \text{REL}\rangle, M)$, where $M = (V_{\underline{K}}, E_{\underline{K}}, \text{nod}, \text{lab}_{\underline{K}}, \text{ext})$ with for all $e \in E_{\underline{K}}$ and $j \in [1, \text{rank}_{\Sigma'}(\text{lab}_{\underline{K}}(e))]$:

$$\text{nod}(e, j) = \begin{cases} \text{nod}_{\underline{K}}(e, j) & \text{if } e \in \text{tedg}(\underline{K}) \\ \text{nod}_{\underline{K}}(e, i) & \text{if } e \in \text{nedg}(\underline{K}), \end{cases}$$

where i is the j-th element of $\text{EQ}(\text{lab}_{\underline{K}}(e))$, and for all $j \in [1, \text{rank}_{\Sigma'}(\langle X, \text{REL}\rangle)]$: $\text{ext}(j) = \text{ext}_{\underline{K}}(i)$, where i is the j-th element of $\text{EQ}(\langle X, \text{REL}\rangle)$.

Note that GR' is identification-free and loop-free. It is not difficult to show that the mapping $\chi: P_1 \to P'$ defined as $\chi(\pi_1) = \pi'$ (with π_1 and π' as described above) satisfies the conditions (1) and (2) of Lemma 6.2. From these conditions (and the fact that $\chi(P_1) = P'$) it follows that $L(GR') = L(GR_1)$, and hence GR' is term-generating. Thus, by Lemma 6.2, there exists a cts $\mathbf{T}'$ with semantic domain D and right grammar GR' such that $\tau(\mathbf{T}') = \tau(\mathbf{T}_1) = \tau(\mathbf{T})$.

This shows our statement. Similarly, we may assume that the right grammar of a cts is identification-free (by a similar adaptation of the construction in the proof of Lemma 3.2 of [EngHey1]). □

Thus, by the above example, we may assume that the right (term-generating) grammar of a cts is identification-free and loop-free, without changing its semantic domain.

It may even be assumed for a cts that its right grammar is clean term-generating. This can easily be proved with the help of Lemma 6.4 below, Proposition 2.1, and Lemma 6.1, in that order, similar to the "garbage theorem" in Sect. 5. Consequently, we can give an alternative definition of the class of translations realized by cts's with semantic domain D, namely $\tau(\text{CTS}, D) = \{\tau(\mathbf{T}) \mid \mathbf{T}$

is a cts with semantic domain D, of which the right grammar is clean term-generating}.

As in Sect. 3 we can give two other possible definitions of $\tau(\mathrm{CTS}, D)$ by allowing all cfhg's generating 1-hypergraphs as right grammar of a cts. We define the translation of such a cts $\mathbf{T}=(D, GL, GR, \varphi)$ as $\tilde{\tau}(\mathbf{T})$ $=\{(\mathrm{yield}_{GL}(\ell), (\mathrm{term}(\mathrm{yield}_{GR}(\varphi(\ell))))_D) \mid \ell$ is a derivation tree of GL and $\mathrm{yield}_{GR}(\varphi(\ell))$ is a (clean) jungle$\}$.

Obviously, we cannot translate the proof of Theorem 3.1(2) directly to cts's. Nevertheless, we know from [Cou4] how to construct, for a given cfhg G generating m-hypergraphs over a ranked alphabet Δ and an MSOL formula ϑ in $\mathscr{L}_{\Delta,m}$, a cfhg G' such that $L(G')=\{H\in L(G) \mid H\vDash\vartheta\}$. In fact, if $GR=(\Sigma, \Delta, P_{GR}, S)$ is a cfhg generating 1-hypergraphs and ϑ expresses that a 1-hypergraph over Δ is a (clean) jungle, then a (clean) term-generating cfhg GR' can be constructed such that there exists a mapping ψ from the nonterminals of GR' to $\Sigma-\Delta$ that satisfies demand (1) of Lemma 6.3 and the following demand (2'): for every derivation tree ℓ of GR, $\mathrm{yield}(\ell)$ is a (clean) jungle if and only if there exists a derivation tree ℓ' of GR' with $\psi(\ell')=\ell$. Thus, with the construction in the proof of Lemma 6.3 an "ordinary" cts $\mathbf{T}'$ (i.e., with a (clean) term-generating right grammar, namely GR') is obtained that realizes the same translation as $\mathbf{T}$, because a derivation tree of GR that does not yield a (clean) jungle does not influence $\tilde{\tau}(\mathbf{T})$. Hence, $\tilde{\tau}(\mathbf{T})=\tau(\mathbf{T}')\in\tau(\mathrm{CTS}, D)$.

We now return to the main aim of this section, the comparison of the translation power of cts's with that of AG's, and we show that $\tau(\mathrm{CTS}, D)\subseteq\tau(\mathrm{AG}, D)$, for every semantic domain D.

Lemma 6.4 *For every* cts $\mathbf{T}$ *with semantic domain* D *there exists an* AG G *over* D *such that* $\tau(G)=\tau(\mathbf{T})$.

Proof. Let $\mathbf{T}=(D, GL, GR, \varphi)$ be a cts with semantic domain $D=(V, \Gamma)$. The proof is a variation of the one of Lemma 5.1 with G corresponding to GR.

In that proof, we assumed for technical reasons that every derivation subtree of the cfhg yields a jungle with variables. We mentioned that for every identification-free, loop-free, and reduced term-generating cfhg G, we may define a (unique) permutation of the external nodes and the tentacles of the nonterminal hyperedges of the right-hand sides of the productions of G such that every permuted derivation subtree of G yields a jungle with variables. As argued before, we may assume that GR is identification-free, loop-free, and reduced. Hence, we may also assume that every derivation subtree of GR yields a jungle with variables, because such a permutation can obviously be realized by a transformation of the type described in Lemma 6.2.

Therefore, we can construct for GR (that has terminal alphabet $\mathrm{inc}(\Gamma)$ by the definition of cts) a term-generating AG G' over $(T(\Gamma), \Gamma)$ such that $\mathrm{OUT}(G')$ $=\mathrm{TERM}(GR)$, analogous to the construction of the AG G' in the proof of Lemma 5.1.

Recall that in that construction the "π" in the right-hand sides of the productions of the underlying grammar G_0 was added to obtain a 1–1 correspondence between the productions of the cfhg and those of G_0. In this case, we may take GL instead of G_0 as underlying grammar of G', with, for every production $p=X_0\to w_0X_1w_1X_2w_2\ldots X_mw_m$ of GL, $r_p=r_{p'}$, where p' is the production $X_0\to\varphi(p)X_1X_2\ldots X_m$ of G_0. Since we may assume φ to be surjective, still $\mathrm{OUT}(G')=\mathrm{TERM}(GR)$. But as in the proof of Lemma 5.1 it can be shown

not only that $\mathrm{OUT}(G')=\mathrm{TERM}(GR)$, but even that $\mathrm{val}_{G'}(\langle \alpha_d, \mathrm{root}(\ell)\rangle, \ell) = \mathrm{term}(\mathrm{yield}_{GR}(\varphi(\ell)))$ for every derivation tree ℓ of GL. This shows that by changing the semantic domain $(T(\Gamma), \Gamma)$ of G' into (V, Γ), G' is an AG over D such that $\tau(G')=\tau(\mathbf{T})$. □

Thus by Lemma's 6.1 and 6.4 we may extend our main result of Sect. 5, Theorem 5.2, as follows.

Theorem 6.5 $\tau(\mathrm{CTS}, D)=\tau(\mathrm{AG}, D)$, *for every semantic domain* D.

Conclusion

We have shown that context-free hypergraph grammars have the same term generating power as attribute grammars, and, when provided with an additional context-free grammar as input component, they have the same translation power as attribute grammars. Thus, cfhg's provide a formalism to describe the semantics of programming languages, similar to AG's.

It is shown in [HabKrePlu; HofPlu] that hypergraph rewriting rules (that are usually not context-free) can be used to simulate term rewriting systems on jungles rather than trees. In particular they describe "folding" rules that can be used to obtain jungles from trees, or to increase the sharing of subterms in jungles. One goal one might try to achieve is to try to model as many aspects as possible of the implementation of attribute grammars in the uniform formalism of hypergraph rewriting (cf. [Hof] where another type of graph rewriting is used), i.e., to combine the context-free hypergraph grammar that simulates formal attribute evaluation, with other hypergraph rewriting rules. For instance, the actual evaluation of the jungles could be realized by term rewriting, and hence by hypergraph rewriting, in case the sets of values of the attributes are specified equationally as abstract data types (see [Hof]). Another possibility would be to use the folding rules of [HabKrePlu; HofPlu] during formal evaluation, to improve space efficiency.

Extending ideas in [Cou3], it has recently been shown in [CorRos] (see also [CorRosPar]) that there is a close relationship between jungle rewriting and logic programming. Since it is well known that attribute grammars are closely related to logic programming (see, e.g., [DerMal; CouDer]), it is no surprise that their result seems to be very similar to (but more general than) ours. To be more precise, logic programming is close to attribute grammars in which trees are used as actual attribute values (rather than representations of expressions); moreover, these trees can not only be composed (i.e., substituted in each other), but also tested (by looking at the top label) and decomposed (by taking subtrees). Thus, attribute grammars as considered in this paper are the special case that trees can be composed only. In [CorRos] it is shown that the decomposition of trees can be realized in an elegant way by considering context-free jungle grammars, in which the rewriting of jungles is defined within the category of jungles rather than the larger category of hypergraphs. Such rewriting may have a "global" effect on the rewritten jungle, as opposed to the "local" effect of hypergraph rewriting. It seems that our result corresponds precisely to the case of logic programming where one may in fact stick to the "local" hypergraph rewriting because of the absence of tree decomposition. However, the precise relationship between the two results needs more investigation.

As mentioned in the introduction, the classes of tree languages and tree translations defined by attribute grammars have been investigated in the tree literature. The class of tree translations is contained properly in the class of translations defined by macro tree transducers. In [EngVog2] the formalism of context-free hypergraph grammars is extended in a natural way, and it is shown that these extended grammars have the same power as macro tree transducers. One might also think about generalizing the cts in such a way that duplication and deletion of nonterminals would be allowed (as in the generalized syntax-directed translation schemes of [AhoUll]). This would be equivalent to the formalism in [EngVog2].

Rather than generalizing the context-free hypergraph grammar, one may try to consider restrictions that generate well-known subclasses of the class of tree languages generated by attribute grammars. One such class is the class of tree languages generated by IO context-free tree grammars (see [DusParSedSpe; EngFil]). We conjecture that this class has a nice characterization: it is generated by the context-free hypergraph grammars of which all sentential forms are jungles. Another question is which class of tree languages is generated by context-free hypergraph grammars if one forbids sharing, i.e., the generated jungles should all be trees, in the sense that from each node there is a unique hyperpath to the external node (which is the root). This class of tree languages certainly contains the tree languages generated by the grammars of [Rao].

We finally mention the area of attributed graph grammars (see, e.g., [Göt]). One possible type of attributed graph grammar would be the context-free hypergraph grammar extended with (inherited and synthesized) attributes, just as in the case of an ordinary context-free grammar. Such an attributed cfhg could be used to define a translation from graphs to values (in some semantic domain). Another way of using it (as in [Göt]) would be to allow the terminal symbols to have inherited attributes, to disregard the synthesized attributes of the initial nonterminal, and to view the attributed cfhg as a device that generates attributed hypergraphs, i.e., hypergraphs of which the edges have attribute values. One example is the generation of pictures, where the graph represents the topological aspects of the picture, whereas the attribute values represent its metrical properties. Another example is the generation of the attributed derivation trees of an ordinary attribute grammar. It should be rather obvious that our results can be generalized to such attributed cfhg's: every attributed cfhg can be simulated by an ordinary cfhg (in which the attribute values are represented as jungles, attached to the edges by additional tentacles). Thus, in a theoretical sense, one does not need attributed cfhg's; it suffices to consider cfhg's that generate graphs, parts of which can be interpreted as values in some domain.

Acknowledgements. The authors wish to thank George Leih for several useful observations. The remarks of the referees have been very helpful in writing the conclusion.

References

[AhoSetUll] Aho, A.V., Sethi, R., Ullman, J.D.: Compilers; Principles, techniques, and tools. Reading, MA: Addison-Wesley 1986

[AhoUll] Aho, A.V., Ullman, J.D.: The theory of parsing, translation, and compiling. Englewood Cliffs, NJ: Prentice-Hall 1972

[BauCou] Bauderon, M., Courcelle, B.: Graph expressions and graph rewritings. Math. Syst. Theory **20**, 83–127 (1987)

[Bar] Bartha, M.: An algebraic definition of attributed transformations. In: Gécseg, F. (ed.) Fundamentals of computation theory. (Lect. Notes Comput. Sci., vol. 117, pp. 51–60) Berlin, Heidelberg, New York: Springer 1981

[Boc] Bochmann, G.V.: Semantic evaluation from left to right. Commun. ACM **19**, 55–62 (1976)

[ChiMar] Chirica, L.M., Martin, D.F.: An order-algebraic definition of Knuthian semantics. Math. Syst. Theory **13**, 1–27 (1979)

[CorRos] Corradini, A., Rossi, F.: On the power of context-free jungle rewriting for term rewriting systems and logic programming, University of Pisa, Italy, June 1990

[CorRosPar] Corradini, A., Rossi, F., Parisi-Presicce, F.: Logic programming as hypergraph rewriting. In: Proceedings CAAP '91. (Lect. Notes Comput. Sci., vol. 493, pp. 275–295) Berlin, Heidelberg, New York: Springer 1991

[Cou1] Courcelle, B.: Equivalences and transformations of regular systems, applications to recursive program schemes and grammars. Theor. Comput. Sci. **42**, 1–122 (1986)

[Cou2] Courcelle, B.: An axiomatic definition of context-free rewriting and its application to NLC graph grammars. Theor. Comput. Sci. **55**, 141–181 (1987)

[Cou3] Courcelle, B.: On using context-free graph grammars for analyzing recursive definitions. In: Fuchi, K., Kott, L. (eds.) Programming of future generation computers II, pp. 83–122. Amsterdam: Elsevier 1988

[Cou4] Courcelle, B.: The monadic second-order logic of graphs, I: recognizable sets of finite graphs. Inf. Comput. **85**, 12–75 (1990)

[CouDer] Courcelle, B., Deransart, P.: Proofs of partial correctness for attribute grammars with applications to recursive procedures and logic programming. Inf. Comput. **78**, 1–55 (1988)

[CouFra] Courcelle, B., Franchi-Zannettacci, P.: Attribute grammars and recursive program schemes I and II. Theor. Comput. Sci. **17**, 163–191, 235–257 (1982)

[DerJouLor] Deransart, P., Jourdan, M., Lorho, B.: Attribute grammars; definitions, systems and bibliography. (Lect. Notes Comput. Sci., vol. 323) Berlin, Heidelberg, New York: Springer 1988

[DerMal] Deransart, P., Maluszynski, J.: Relating logic programs and attribute grammars. J. Logic Program. **2**, 119–155 (1985)

[DusParSedSpe] Duske, J., Parchmann, R., Sedello, M., Specht, J.: IO-macrolanguages and attributed translations. Inf. Control **35**, 87–105 (1977)

[EhrNagRosRoz] Ehrig, H., Nagl, M., Rozenberg, G., Rosenfeld, A. (eds.): Graph-grammars and their application to computer science. (Lect. Notes Comput. Sci., vol. 291) Berlin, Heidelberg, New York: Springer 1987

[Eng1] Engelfriet, J.: Some open questions and recent results on tree transducers and tree languages. In: Book, R.V. (ed.) Formal language theory: perspectives and open problems, pp. 241–286. New York: Academic Press 1980

[Eng2] Engelfriet, J.: Tree transducers and syntax-directed semantics. TW Memorandum 363, Twente University of Technology, 1981, presented at the 7th CAAP, March 1982, Lille

[Eng3] Engelfriet, J.: The complexity of languages generated by attribute grammars. SIAM J. Comput. **15**, 70–86 (1986)

[EngFil] Engelfriet, J., Filè, G.: The formal power of one-visit attribute grammars. Acta Inf. **16**, 275–302 (1981)

[EngHey1] Engelfriet, J., Heyker, L.M.: The string generating power of context-free hypergraph grammars. J. Comput. Syst. Sci. **43**, 328–360 (1991)

[EngHey2] Engelfriet, J., Heyker, L.M.: The term generating power of context-free hypergraph grammars. In: Ehrig, H., Kreowski, H.-J., Rozenberg, G. (eds.) Graph-grammars and their application to computer science. (Lect. Notes Comput. Sci., vol. 532) Berlin, Heidelberg, New York: Springer 1991, pp. 328–343

[EngLeiRoz] Engelfriet, J., Leih, G., Rozenberg, G.: Apex graph grammars and attribute grammars. Acta Inf. **25**, 537–571 (1988)

[EngRoz] Engelfriet, J., Rozenberg, G.: A comparison of boundary graph gram-

mars and context-free hypergraph grammars. Inf. Comput. **84**, 163–206 (1990)

[EngVog1] Engelfriet, J., Vogler, H.: Macro tree transducers. J. Comput. Syst. Sci. **31**, 71–146 (1985)

[EngVog2] Engelfriet, J., Vogler, H.: The translation power of top-down tree-to-graph transducers. in preparation

[Fed] Feder, J.: Plex languages. Inf. Sci. **3**, 225–241 (1971)

[Fil] Filè, G.: Interpretation and reduction of attribute grammars. Acta Inf. **19**, 115–150 (1983)

[Fül] Fülöp, Z.: On attributed tree transducers. Acta Cybern. **5**, 261–279 (1981)

[Gan] Ganzinger, H.: On storage optimization for automatically generated compilers. In: Weibrauch, K. (ed.) Theoretical computer science, 4th GI Conference. (Lect. Notes Comput. Sci., vol. 67, pp. 132–141) Berlin, Heidelberg, New York: Springer 1979

[Göt] Göttler, H.: Graph-grammars and diagram editing. In [EhrNagRosRoz], pp. 216–231

[Hab] Habel, A.: Hyperedge replacement: grammars and languages. Ph.D. Thesis, Bremen, 1989

[HabKre1] Habel, A., Kreowski, H.-J.: Some structural aspects of hypergraph languages generated by hyperedge replacement. In: Brandenburg, F.J., Vidal-Naquet, G., Wirsing, M. (eds.) STACS '87 Proceedings. (Lect. Notes Comput. Sci., vol. 247, pp. 207–219) Berlin, Heidelberg, New York: Springer 1987

[HabKre2] Habel, A., Kreowski, H.-J.: May we introduce to you: hyperedge replacement. In [EhrNagRosRoz], pp. 15–26

[HabKrePlu] Habel, A., Kreowski, H.-J., Plump, D.: Jungle evaluation. In: Sanella, D., Tarlecki, A. (eds.) Recent trends in data type specification. (Lect. Notes Comput. Sci., vol. 332, pp. 92–112) Berlin, Heidelberg, New York: Springer 1987

[Hof] Hoffmann, B.: Modelling compiler generation by graph grammars. In: Ehrig, H., Nagl, M., Rozenberg, G. (eds.) Graph-grammars and their application to computer science. (Lect. Notes Comput. Sci., vol. 153, pp. 159–171) Berlin, Heidelberg, New York: Springer 1983

[HofPlu] Hoffmann, B., Plump, D.: Jungle evaluation for efficient term rewriting. In: Grabowski, J., Lescanne, P., Wechler, W. (eds.) Algebraic and logic programming. (Lect. Notes Comput. Sci., vol. 343, pp. 191–203) Berlin, Heidelberg, New York: Springer 1988

[HofSch] Hoffmann, B., Schmiedecke, I.-R.: Multi-pass parsing for two-level grammars. In: Dembinski, P. (ed.) Mathematical Foundations of Computer Science 1980. (Lect. Notes Comput. Sci., vol. 88, pp. 275–290) Berlin, Heidelberg, New York: Springer 1980

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

[Kre] Kreowski, H.-J.: Rule trees represent derivations in edge replacement systems. In: Rozenberg, G., Salomaa, A. (eds.) The Book of *L*, pp. 217–232. Berlin, Heidelberg, New York: Springer 1986

[Lau] Lautemann, C.: Decomposition trees: structured graph representation and efficient algorithms. In: Dauchet, M., Nivat, M. (eds.) CAAP '88. Proceedings. (Lect. Notes Comput. Sci., vol. 299, pp. 28–39) Berlin, Heidelberg, New York: Springer 1988

[LenWan] Lengauer, T., Wanke, E.: Efficient analysis of graph properties on context-free graph languages (extended abstract). In: Lepistö, T., Salomaa, A. (eds.) Automata languages and programming. ICALP '88 Proceedings. (Lect. Notes Comput. Sci., vol. 317, pp. 379–393) Berlin, Heidelberg, New York: Springer 1988

[Lor] Lorho, B. (ed.): Methods and tools for compiler construction. New York: Cambridge University Press 1984

[Mad] Madsen, O.L.: On defining semantics by means of extended attribute grammars. In: Jones, N.D. (ed.) Semantics-directed compiler generation. (Lect. Notes Comput. Sci., vol. 94, pp. 259–299) Berlin, Heidelberg, New York: Springer 1980

[MezWri] Mezei, J., Wright, J.B.: Algebraic automata and context-free sets. Inf. Control

11, 3–29 (1967)

[MonRos] Montanari, U., Rossi, F.: An efficient algorithm for the solution of hierarchical networks of constraints. In: [EhrNagRosRoz], pp. 440–457

[Pra] Pratt, T.W.: Pair grammars, graph languages and string-to-graph translations. J. Comput. Syst. Sci. **5**, 560–595 (1971)

[Rao] Raoult, J.-C.: Algebraic sets of tree-vectors and rational tree-transductions. Publication Nr. 502, IRISA, Rennes, France, 1989