Acta Informatica 6, 297—316 (1976)

Grammars on Partial Graphs

H. J. Schneider and H. Ehrig

Received February 28, 1975

Summary. The concept of Chomsky-grammars is generalized to graph-grammars; the "gluing" of graphs is defined by a pushout-construction. In the present paper, we allow the left-hand and right-hand side of a production to be partial graphs, i.e. graphs in which there may be edges without a source or target node. A necessary and sufficient condition for applicability of productions is given. Furthermore, convex graph-grammars are studied.

Introduction

In trying to generalize the concept of Chomsky-grammars to multidimensional symbol-structures, the main problem is to specify the embedding and the exchange of substructures in order to get direct derivation of symbol-structures. There are several concepts to solve this problem.

The first known approach is the concept of web grammars [8]. A web is a directed graph with labelled nodes; a production is given by a pair of webs (W_l, W_r). Applying the production to a web W, a subweb of which is W_l, we must replace W_l by W_r; in addition to the case of strings, we need a rule as to how to connect the nodes of W_r with the nodes remaining in W after W_l is removed. In [8], these "embedding-rules" are given in an informal way. A formal description of these rules is due to [6].

Another approach is given in [12, 13], where "n-diagrams" are considered, i.e. relational systems with a set K of labelled nodes and n partial orderings $\varrho_1, \ldots, \varrho_n$ on K. A production of a n-grammar consists of two n-diagrams D_l and D_r describing precisely the embedding rules. A generalization of this approach is given in [7]. These approaches consider convex graph-grammars only.

In [2, 3], we proposed an algebraic approach using homomorphisms and pushouts. In contrast to the above mentioned concepts, we were able to define gluing of graphs and direct derivations of labelled graphs separately. This concept was generalized by [11] to partially labelled graphs and noninjective embeddings. In the present paper, we discuss another generalization, mentioned and used already in [14] for the description of incremental compilers: We allow the lefts hand and right-hand side of our productions to be "partial" graphs, i.e. graph-in which there may be edges without a source or target node, but we give additional conditions to make sure that the derived graphs are total, labelled graphs again.

There is a great variety of applications about which we can give only some references: pattern recognition [5, 8], translation of programming languages [10],

two-dimensional programming languages [1], incremental compilers [14], biological organisms [4].

The reader may ask why we want to allow partial graphs: Instead of a partial graph which is to be replaced, we may replace a sufficiently large graph containing the given partial graph. It is clear that we have to add, on both sides of the original production, some nodes which remain unchanged in the derivation step. These nodes may be interpreted as constant context. Because all possibilities of context must be added, the set of productions increases rapidly. Additionally, in the applications mentioned, data structures are to be stored by node and edge lists (see e.g. [15]); if we want to describe syntactic operations on such data structures by formal graph-grammars, it would be not a natural way of thinking to add unnecessary information (which must be tested in each step) to the productions and to increase at the same time the number of productions. (In the case of an infinite labelling alphabet, this process leads to an infinite set of productions; furthermore, the productions of a convex graph-grammar with total graphs must take into account all combinations of edges between nodes of the original partial graph and the additional peripheral nodes.) Therefore, it is of interest to know some criteria for applicability of productions with partial graphs.

The definitions of partial graphs, the embedding mechanism, and derivation between graphs are given in Section 1. The notion of derivation is illustrated by some examples in Section 2. Section 3 contains the main results and proofs: the definition of labelled gluing is shown to be correct, and there is a necessary condition for applicability and unique applicability of productions. The last section is concerned with convex graph grammars.

1. Definitions

First, we summarize some basic definitions, given for total graphs in [2, 3]. Since we allow partial graphs to be the left-hand and right-hand side of productions, we have to modify the definition of morphisms, especially injections, and labelled gluing.

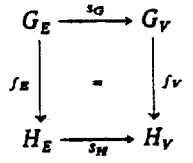
Definition 1.1. A *partial directed graph* G consists of two sets G_E and G_V together with two partial mappings $q: G_E \to G_V$ and $z: G_E \to G_V$. The graph is called *total* and *directed* if and only if $\mathrm{dom}(q) = \mathrm{dom}(z) = G_E$.

Remarks. (i) The elements of G_E are called edges, these of G_V nodes or vertices; q determines the source and z the targets of edges.

(ii) With respect to the applications we have in mind, it is not of interest to allow edges to be neither in the domain of q nor in the domain of z, but the proofs even hold true in this case. In the following we speak of "graph" as abbreviation of "partial directed graph".

Definition 1.2. Let $G = (G_E, G_V, q_G, z_G)$ and $H = (H_E, H_V, q_H, z_H)$ be graphs. A *weak graph morphism* is a pair $f = (f_E, f_V)$ of mappings $f_E: G_E \to H_E$, $f_V: G_V \to H_V$

where



commutes for $s=q$ and $s=z$ in the sense that $s_H(f_E(e))$ is defined if $s_G(e)$ is defined.

Since we require partial graphs to be subgraphs of a total one, we may not use in each case the usual graph-morphism:

Definition 1.3. A weak graph-morphism $f: G \to H$ is called *graph-morphism* if $s_H(f_E(e))$ is defined if and only if $s_G(e)$ is defined.

Remarks. (i) Partial graphs together with weak graph-morphisms constitute a category $\mathfrak{Graph}^-$; composition of morphisms and identities are defined componentwise.

(ii) If $\mathfrak{Sets}^2$ is the category of all pairs of sets (together with all pairs of mappings), the *forgetful functor* $U: \mathfrak{Graph}^- \to \mathfrak{Sets}^2$ is given by

$$UG := (E, V) \quad \text{for } G: E \overset{q}{\underset{z}{\rightrightarrows}} V,$$

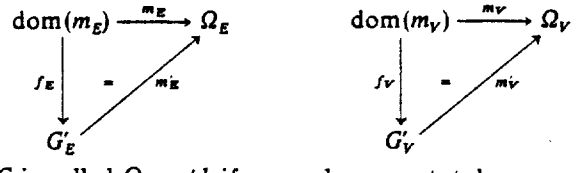
$$Uf := (f_E, f_V) \quad \text{for } f: G \to G'.$$

(iii) Let $\mathfrak{Graph}$ be the full subcategory of $\mathfrak{Graph}^-$ the objects of which are the total graphs.

(iv) If $\mathfrak{X}$ is a category, we denote with $|\mathfrak{X}|$ the set of objects and with $\mathrm{Mor}\,\mathfrak{X}$ the set of morphisms.

Most of the following results are applicable to infinite graphs, too, but only finite graphs are of interest here. From now on, we consider only graphs the nodes and edges of which are labelled:

Definition 1.4. Let $\Omega = (\Omega_E, \Omega_V) \in |\mathfrak{Sets}^2|$ be a pair of sets (*labelling alphabet*), $G: E \overset{q}{\underset{z}{\rightrightarrows}} V$ a finite (partial) graph, and $m := (m_E, m_V)$ a pair of partial mappings $m_E: E \to \Omega_E$, $m_V: V \to \Omega_V$ (*labelling mapping*). Then (G, m) with $m: G \to \Omega$ is called (partial) Ω^-*-graph*. Given Ω^-- graphs (G, m) and (G', m'), a graph morphism $f: G \to G'$ is called Ω^-*-graph-morphism* if $m' \cdot Uf = m$ holds in the sense that the following diagrams commute:



Notaton. (i) G is called Ω*-graph* if m_E and m_V are total.

(ii) Ω^--graphs together with these Ω^--graph-morphisms constitute a category $\mathfrak{Graph}^-_\Omega$. The categories $\mathfrak{Graph}^-_\Omega$, $\mathfrak{Graph}_{\Omega^-}$, and $\mathfrak{Graph}_\Omega$ are full subcategories, the objects of which are the partial Ω-graphs, total Ω^--graphs, and total Ω-graphs respectively.

Definition 1.5. A *graph-grammar* is a quadruple $Q=(\Omega, T, S, P)$ where $\Omega:=(\Omega_E, \Omega_V)$ is a labelling alphabet, $T=(T_E, T_V)\subseteq\Omega$ the *terminal alphabet*, S a single-noded discrete labelled graph (*initial graph*), and P a finite set of *productions* of the form:

$$p=(('B, 'm), 'p, K, p', (B', m'))$$

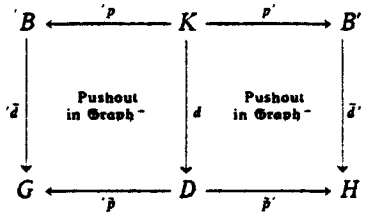
with $K\in|\mathfrak{Graph}^-|$ and $('B, 'm), (B', m')\in|\mathfrak{Graph}_\Omega|$, $('B, 'm)\notin|\mathfrak{Graph}_T|$, $'p\colon K\to'B$ and $p'\colon K\to B'$ graph-morphisms satisfying

$$\begin{aligned} {}'B_E\setminus{}'p_E[K_E]&\subseteq\mathrm{dom}(q_{'B})\cap\mathrm{dom}(z_{'B})\\ B'_E\setminus p'_E[K_E]&\subseteq\mathrm{dom}(q_{B'})\cap\mathrm{dom}(z_{B'}).\end{aligned}\qquad(*)$$

If K is a set and p a partial mapping of K into another set, $p[K]$ denotes the set of images of K.

$('B, 'm)$ and (B', m') are called the left-hand and right-hand side of p respectively. To derive a new graph from a given one, the left-hand side of a production is to be replaced by its right-hand side; the auxiliary graph K and the morphisms $'p, p'$ serve to define how to glue (B', m') into the graph which remains after having deleted $('B, 'm)$. Condition (*) ensures an edge in $'B$ or B' without source or target node to be the image of such an edge in K. (This is used in No. 3.2 to show that the pushout-object is in $|\mathfrak{Graph}|$.) For better understanding, the reader should study now the productions given in Section 2, and should read these examples step by step as the notions and constructions are introduced.

If we omit the labelling for a moment, we may introduce the notion of derivability by two pushout-constructions:



The graph H may be called derivable from G. Intuitively, the graph D is the remainder of G after having deleted $'B$ and before inserting B'; in addition, D contains the gluing nodes and gluing edges. One must ask why this diagram includes unlabelled graphs only. Actually, we could also consider this diagram for labelled graphs. But this would be a real restriction for the applications, because a common labelling of K and D would imply that corresponding gluing points in $'B$ and B' must have the same labels. For example, the labels h of '1 and f of 1' in figure 1 would have to be equal. Therefore, we don't label the graph K and proceed in two steps: first, we define the gluing of D and $'B$ (resp. D and B') along K, and then we define the labelling of G (resp. H). Because of the symmetry of this construction, we may restrict our considerations to one part of the diagram:

Theorem 1.6. Let $K\in|\mathfrak{Graph}^-|$, $B\in|\mathfrak{Graph}^-|$, $D\in|\mathfrak{Graph}|$ be such that there exists an injective weak graph-morphism $d\colon K\to D$ and a graph-morphism

$p\colon K \to B$ satisfying

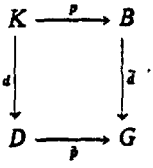
(a) $B_E \setminus p_E[K_E] \subseteq \mathrm{dom}(q_B) \cap \mathrm{dom}(z_B)$,

(b) If $p_E(e) = p_E(e')$ for some $e, e' \in K_E$, then

$$(b1)\ e, e' \in \mathrm{dom}(q_K) \Rightarrow q_D\, d_E(e) = q_D d_E(e'),$$

$$(b2)\ e, e' \in \mathrm{dom}(z_K) \Rightarrow z_D\, d_E(e) = z_D\, d_E(e').$$

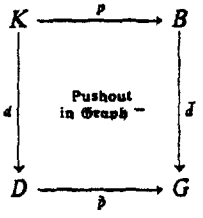
Then, there exists a $G \in |\mathfrak{Graph}|$, such that



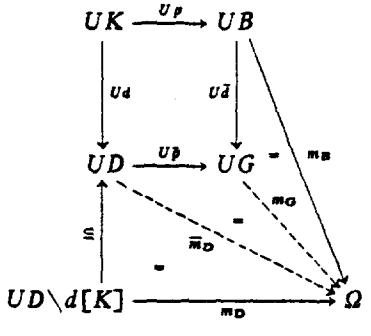
is pushout-diagram in $\mathfrak{Graph}^-$.

Conditions (a) and (b) are used to show that G is a total graph although the pushout-construction is in $\mathfrak{Graph}^-$. If edges without source or target node are identified by p_E, condition (b) ensures the source and target nodes to be identical, too. This theorem is a basis of the following definition and will be proven in No. 3.2.

Definition 1.7. Let be $K \in |\mathfrak{Graph}^-|$, $(B, m_B) \in |\mathfrak{Graph}^-_\Omega|$, $(D, m_D) \in |\mathfrak{Graph}^-_\Omega|$, d and p as in No. 1.6 and $\mathrm{dom}(m_D) = UD \setminus d[K]$, then the pushout-object in



together with the labelling m_G given by



is called *labelled gluing* of (B, m_B) and (D, m_D) along K, abbreviated as:

$$(G, m_G) = (D, m_D) \coprod_{d, p} (B, m_B)$$

The labelling of G is uniquely determined and $\tilde{p}$ and $\tilde{d}$ are morphisms in $\mathfrak{Graph}_\Omega$ and $\mathfrak{Graph}_{\bar{\Omega}}$ respectively. According to injectivity of d,

$$UD\backslash d[K] \xrightarrow{\subseteq} UD \xleftarrow{Ud} UK$$

is a coproduct-diagram in $\mathfrak{Sets}^2$. Therefore, we have a unique $\bar{m}_D$ with $\bar{m}_D \cdot \subseteq = m_D \wedge \bar{m}_D \cdot Ud = m_B \cdot Up$. (Note that $UD\backslash d[K]$ is the domain of m_D.) Because of the pushout-property of $Ud \cdot Up = U\tilde{p} \cdot Ud$, there exists a unique m_G with $m_G \cdot U\tilde{p} = \bar{m}_D \wedge m_G \cdot U\tilde{d} = m_B$.

Remark. Although in this definition, K is not labelled and D only labelled partially, there exist m'_D and m_K such that

$$\begin{array}{ccc} (K, m_K) & \xrightarrow{p} & (B, m_B) \\ {\scriptstyle d}\downarrow & \text{Pushout in } \mathfrak{Graph}_{\bar{\Omega}} & \downarrow{\scriptstyle \tilde{d}} \\ (D, m'_D) & \xrightarrow[\tilde{p}]{} & (G, m_G) \end{array}$$

is pushout-diagram in $\mathfrak{Graph}_{\bar{\Omega}}$, and we can show that for each such pushout there is a unique m_D (the restriction of m'_D) and

$$(G, m_G) = (D, m_D) \coprod_{d, p} (B, m_B).$$

The proof is analoguous to that given in [2].

Now, we can define the derivability:

Definition 1.8. $(H, m_H) \in |\mathfrak{Graph}_\Omega|$ is called *directly derivable* from (G, m_G) in Q:

$$(G, m_G) \underset{Q}{\Longrightarrow} (H, m_H)$$

if and only if there is a production $p \in P$, a partially labelled graph $(D, m_D) \in |\mathfrak{Graph}_\Omega\text{-}|$ and an injective, weak graph-morphism $d: K \to D$ satisfying (1.6b) and

$$\mathrm{dom}(m_D) = UD\backslash d[K],$$

such that, up to isomorphism, the following conditions hold:

$$\begin{aligned} (G, m_G) &= (D, m_D) \coprod_{d, 'p} ('B, 'm) \\ (H, m_H) &= (D, m_D) \coprod_{d, p'} (B', m'). \end{aligned} \qquad (*)$$

Remarks. (i) $e := (d, (D, m_D))$ is called an *enlargement.*

(ii) If we want to denote the production and enlargement used, we write $\underset{(p, e)}{\Longrightarrow}$ instead of $\underset{Q}{\Longrightarrow}$; if there is no doubt we write $\Rightarrow$.

(iii) The derivability relation $\overset{*}{\underset{Q}{\Longrightarrow}}$ is the reflexive, transitive closure of $\underset{Q}{\Longrightarrow}$. The set of Ω-graphs generated by Q, is

$$L(Q) := \{(G, m_G) \mid S \overset{*}{\underset{Q}{\Longrightarrow}} (G, m_G) \wedge (G, m_G) \in |\mathfrak{Graph}_T|\}.$$

In many cases it is necessary to forbid unrestricted use of production:

Definition 1.9. A *graph-grammar with enlargements* $Q=(\Omega, T, S, P, E)$ is given by a graph-grammar (Ω, T, S, P) and a family $E=\langle E_p\rangle_{p\in P}$ of enlargements $e=(d, (D, m_D))$. $(G, m_G)\underset{Q}{\longrightarrow}(H, m_H)$ holds if and only if there exists a production p such that (*) holds with an $e\in E_p$.

Remark. The set of enlargements belonging to a production is not necessarily finite. (See e.g. 4.8.)

Definition 1.10. A graph-grammar $Q=(\Omega, T, S, P)$ is called *contextfree* if and only if for all productions $(('B, 'm), 'p, K, p', (B', m'))$ $'B$ contains exactly one node. Q is called *strongly context-free* if and only if it is context-free and $'B$ doesn't contain edges (for all productions).

2. Examples

Example 2.1. First, we consider the production p the left-hand and right-hand side of which are given in Figure 1. (In our figures, we write $v:x$ to represent a node or edge with denotation v and label x.) The labels of the edges are not of interest in this example and are omitted. In our first example, let K be a discrete graph with (unlabelled) nodes 1 and 2; the graph-morphisms $'p$ and p' are defined by $'p(i):='i$ and $p'(i):=i'$ for $i=1, 2$. Now, we choose a partially labelled graph D as given in Figure 2a with $d(i):=i$. Then, the construction given in No. 1.7 yields the graphs G and H of Figure 2b: H is directly derivable from G by the considered production.

In this example, there is an edge in G between two nodes of the replaced subgraph, but this edge is not part of the subgraph: d is not convex. (See Def. 4.1.) This case is not considered in most of the known approaches.

Example 2.2. Next, we consider an example related to structured programming (programming by step-wise refinement): the aim is to refine the condition in an "if-then-else-statement". In other words, node '1 in G is to be replaced by a compound condition which is composed of 1' and 2' (Fig. 3). The production realizing this derivation step is given in Figure 4, the graph D in Figure 5. As in the former examples, $p'_V(i):=i'$, $p'_E(j):=j'$, and $'p_V(1)='p_V(2)='1$, $'p_E(3)='3$, $'p_E(4)='p_E(5)='4$, $'p_E(6)='6$. We must take into consideration the edges 3, '3,

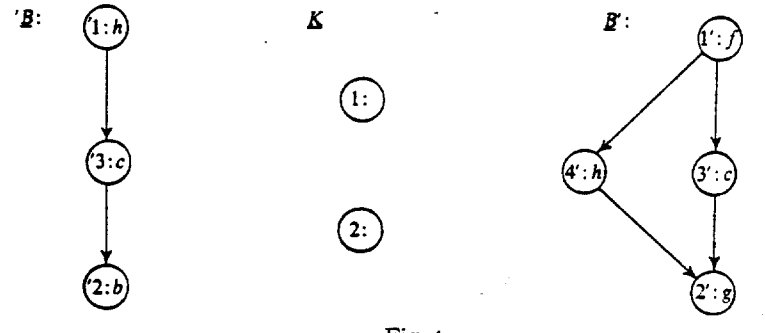


Fig. 1

Fig. 2a

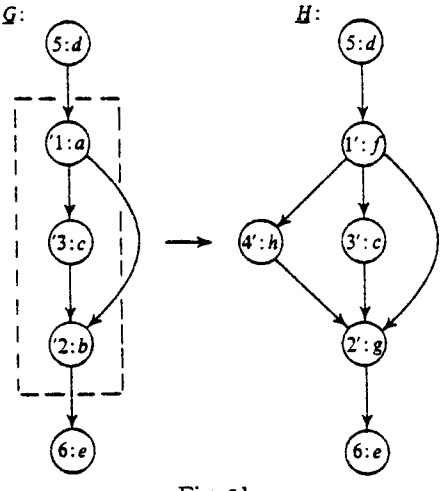


Fig. 2b

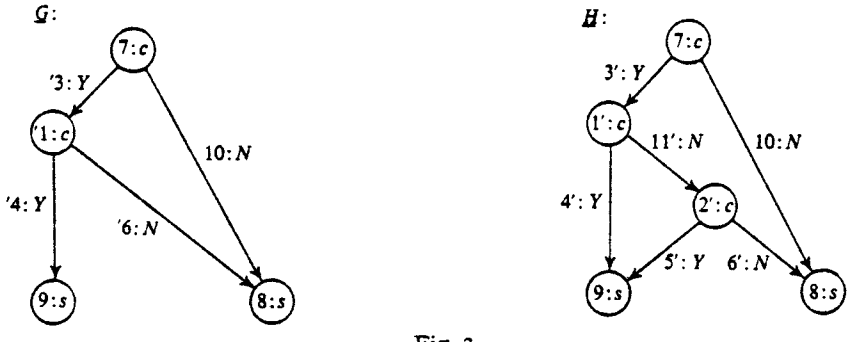


Fig. 3

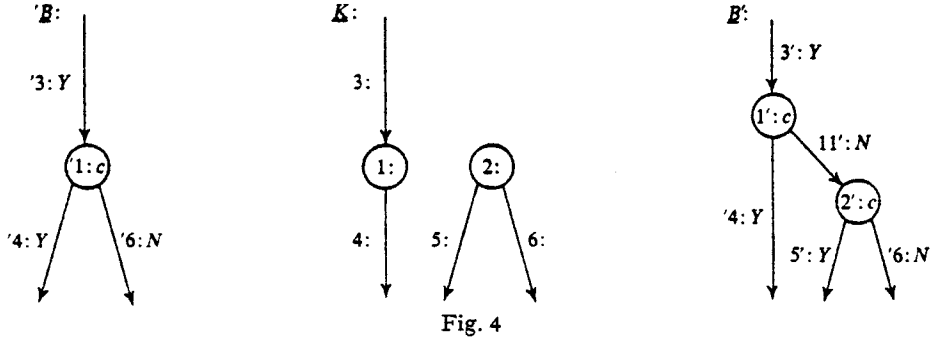


Fig. 4

3', too: if we omit these edges in the production, the node 2' may be the target of 3' because of $'p(1) = 'p(2)$. In such cases, the derivable graph is not uniquely determined. (See Theorem 3.11/12.) More details concerning syntax-directed description of incremental compilers using grammars on partial graphs, are given in [14].

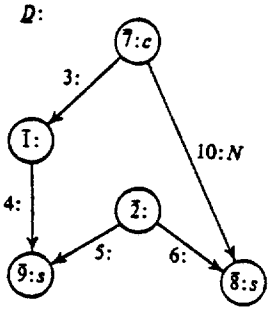


Fig. 5

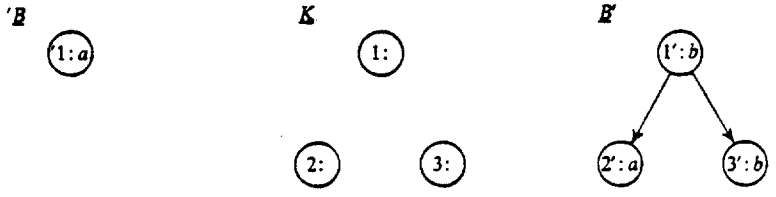


Fig. 6

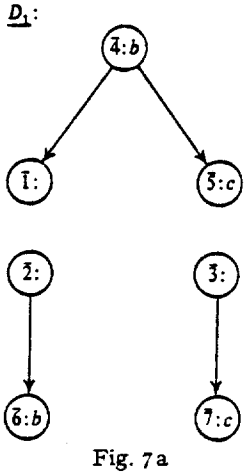


Fig. 7a

Example 2.3. Our last example is concerned with trees. We consider the production given in Figure 6 with $'p$ and p' indicated by the numbers. Using the enlargement, given by the graph D_1 of Figure 7a and $d(i) := i$, we get the derivation $G \rightarrow H_1$ of Figure 7b. However, the resulting graph H_1 depends essentially on the enlargement. If we use a D_2 different from D_1, we get a different H_2 derivable from G by the same production (Fig. 8).

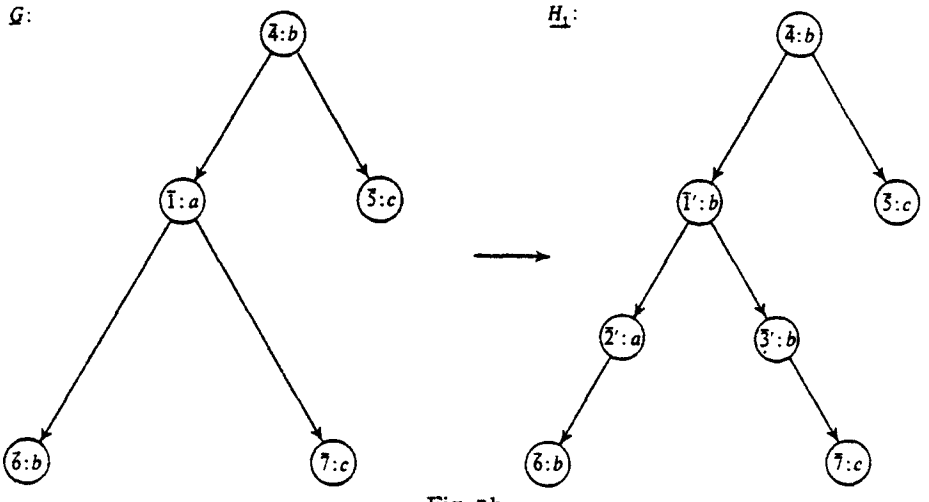


Fig. 7b

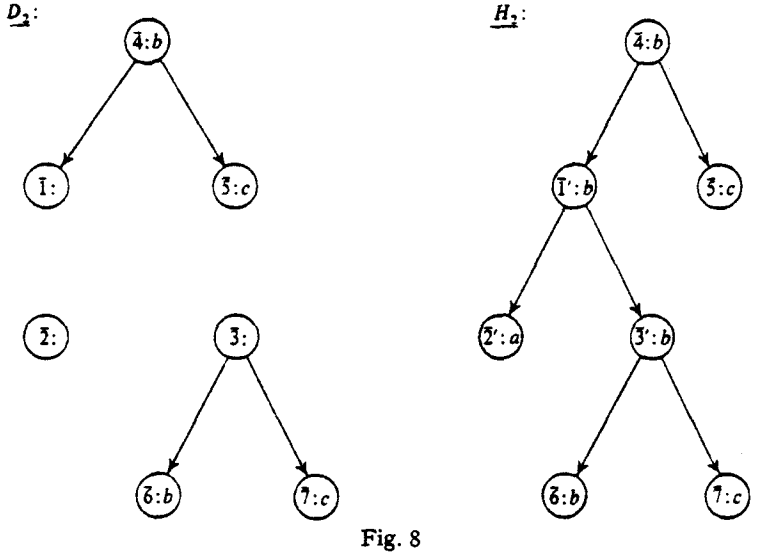
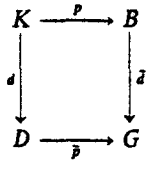


Fig. 8

3. General Results on Graph-Grammars

First, we note a lemma, frequently used:

Lemma 3.1. If



is a pushout-diagram in $\mathfrak{Sets}$ with injective d, we have:

(a) $k \in G \wedge k \notin \tilde{d}[B] \Rightarrow k \in \tilde{p}[D \backslash d[K]]$
$k \in G \wedge k \notin \tilde{p}[D] \Rightarrow k \in \tilde{d}[B \backslash p[K]]$,

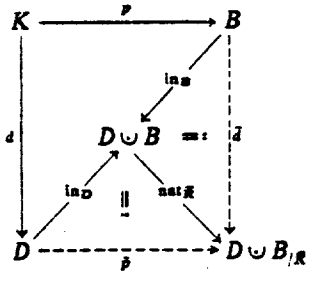
(b) $\tilde{d}(k_1) = \tilde{p}(k_2) \Rightarrow (\dot{\exists} k \in K)(d(k) = k_2 \wedge p(k) = k_1)$,

(c) $\tilde{p}(k_1) = \tilde{p}(k_2) \wedge k_1 \neq k_2$
$\Rightarrow (\exists k_1', k_2' \in K)(d(k_1') = k_1 \wedge d(k_2') = k_2 \wedge p(k_1') = p(k_2'))$,

(d) the restriction $\tilde{p}_0 \colon D \backslash d[K] \to G \backslash \tilde{d}[B]$ of $\tilde{p}$ is bijective.

(e) $\tilde{d}$ is injective.

This lemma follows immediately from explicite construction of pushouts as disjoint union using injectivity of d:



where $\bar{R}$ is the equivalence-relation induced by

$$R := \{(d(k), p(k)) \mid k \in K\}.$$

In No. 1.6, we gave a theorem used to define the labelled gluing, without proof:

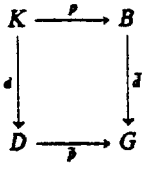
Proof of Theorem (1.6) 3.2. Let $K \in |\mathfrak{Graph}^-|$, $B \in |\mathfrak{Graph}^-|$, $D \in |\mathfrak{Graph}|$ be such that there exists an injective weak graph-morphism $d \colon K \to D$ and a graph-morphism $p \colon K \to B$ satisfying

(a) $B_E \backslash p_E[K_E] \subseteq \mathrm{dom}(q_B) \cap \mathrm{dom}(z_B)$,

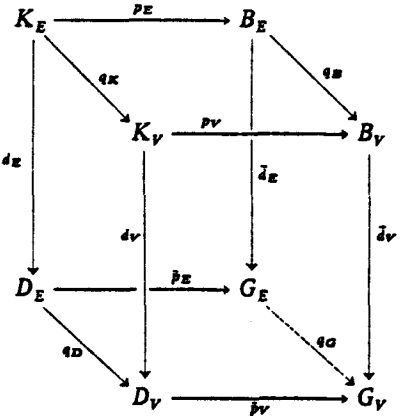
(b) If $p_E(e) = p_E(e')$ for some $e, e' \in K_E$, then

$$(b1)\ e, e' \in \mathrm{dom}(q_K) \Rightarrow q_D\, d_E(e) = q_D\, d_E(e'),$$
$$(b2)\ e, e' \in \mathrm{dom}(z_K) \Rightarrow z_D\, d_E(e) = z_D\, d_E(e').$$

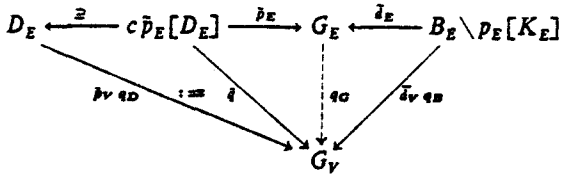
Then, we have to show existence of a $G \in |\mathfrak{Graph}|$, such that



is pushout-diagram in $\mathfrak{Graph}^-$. To prove this, we consider the pushout in $\mathfrak{Sets}^2$:



with partial mappings q_K and q_B. Then, we can construct q_G in the following way: due to No. 3.1 (a, b)



with an arbitrary coretraction c: $\tilde{p}_E[D_E] \to D_E$ of $\tilde{p}_E$, i.e.

$$\tilde{p}_E\, c = id_{\tilde{p}_E[D_E]}$$

is coproduct-diagram. (Because of No. 1.6(a), $B_E \setminus p_E[K_E]$ is in the domain of q_B.) Therefore, we have an uniquely determined q_G, defined everywhere on G_E, such that the diagram commutes. Using a similar construction for z_G, the proof is complete if we can show:

(i) q_G doesn't depend on the choice of c (No. 3.3).

(ii) $\tilde{p}$ is a graph-morphism, $\tilde{d}$ a weak graph-morphism (No. 3.4).

(iii) G is pushout-object in $\mathfrak{Graph}^-$ (No. 3.5).

Remark. (a) In general, the pushout-object in $\mathfrak{Graph}^-$ is not total. Therefore, we must construct it explicitly using the assumptions of No. 1.6.

(b) Here (and in many other proofs), we use the same notation for a mapping and for a restriction if it is clear what we mean.

(c) z_G may be constructed in the same way.

3.3. We consider $e_1, e_2 \in D_E$ with $\tilde{p}_E(e_1) = \tilde{p}_E(e_2) = e$, $e_1 \neq e_2$, and $c_1(e) = e_1$, $c_2(e) = e_2$. With (3.1 c), we have unique existence of $e^1, e^2 \in K_E$ with $d_E(e^i) = e_i$, $p_E(e^1) = p_E(e^2)$. If $q_K(e^i)$ are defined, we have

$$\begin{aligned}\tilde{p}_V\, q_D(e_1) &= \tilde{p}_V\, q_D\, d_E(e^1) = \tilde{p}_V\, d_V\, q_K(e^1) = \tilde{d}_V\, p_V\, q_K(e^1)\\ &= \tilde{d}_V\, q_B\, p_E(e^1) = \tilde{d}_V\, q_B\, p_E(e^2) = \tilde{d}_V\, p_V\, q_K(e^2)\\ &= \tilde{p}_V\, d_V\, q_K(e^2) = \tilde{p}_V\, q_D\, d_E(e^2) = \tilde{p}_V\, q_D(e_2).\end{aligned}$$

p is a graph-morphism; therefore, $q_K(e')$ both are not defined in the other case: (1.6b) yields $q_D(e_1)=q_D(e_2)$. Thus, the total graphs

$$c_1\tilde{p}_E[D_E] \underset{\tilde{z}_1}{\overset{\tilde{q}_1}{\rightrightarrows}} G_V \quad \text{and} \quad c_2\tilde{p}_E[D_E] \underset{\tilde{z}_2}{\overset{\tilde{q}_2}{\rightrightarrows}} G_V$$

with $\tilde{q}_i$ constructed as shown in the diagram of No. 3.2 and $\tilde{z}_i$ constructed analoguously, are isomorphic because

$$\tilde{q}_1(e_1)=\tilde{p}_V\, q_D(e_1)=\tilde{p}_V\, q_D(e_2)=\tilde{q}_2(e_2).$$

3.4. For $e\in c\tilde{p}_E[D_E]$ follows $\tilde{p}_V\, q_D(e)=q_G\tilde{p}_E(e)$ from construction. Let be $e\notin c\tilde{p}_E[D_E]$ and $e':=c\tilde{p}_E(e)$, therefore $\tilde{p}_E(e')=\tilde{p}_E(e)$.

$$\Rightarrow q_G\tilde{p}_E(e)=q_G\tilde{p}_E(e')=\tilde{q}(e')=\tilde{p}_V\, q_D(e'),$$

and the argument of (3.3) yields $\tilde{p}_V\, q_D(e')=\tilde{p}_V\, q_D(e)$. Similarly, $\tilde{d}_V\, q_B(e)=q_G\,\tilde{d}_E(e)$ by construction if $e\in B_E\backslash p_E[K_E]$. If $e\in p_E[\mathrm{dom}(q_K)]$, we consider $e'\in\mathrm{dom}(q_K)$ with $p_E(e')=e$:

$$\begin{aligned} q_G\,\tilde{d}_E(e)&=q_G\,\tilde{d}_E\,p_E(e')=\tilde{p}_V\,q_D\,d_E(e') \qquad (\tilde{p}\text{ is graph-morphism!})\\ &=\tilde{p}_V\,d_V\,q_K(e')=\tilde{d}_V\,p_V\,q_K(e')=\tilde{d}_V\,q_B\,p_E(e')=\tilde{d}_V\,q_B(e).\end{aligned}$$

Since $\tilde{d}$ must be only weak, this completes the proof.

3.5. To prove pushout-property, we consider a weak graph-morphism f: $B\to H$ and a graph-morphism g: $D\to H$ with $f\,p=g\,d$. Because of pushout-property in $\mathfrak{Sets}^2$, there exist unambiguous h_E: $G_E\to H_E$, h_V: $G_V\to H_V$ with

$$h_E\,\tilde{d}_E=f_E, \quad h_V\,\tilde{d}_V=f_V, \quad h_E\tilde{p}_E=g_E, \quad h_V\tilde{p}_V=g_V.$$

The proof is complete, if we can show (h_E, h_V) to be a graph-morphism:

(a) $e\in\tilde{p}_E[D_E]$: We consider e' with $\tilde{p}_E(e')=e$:

$$\begin{aligned} h_V\,q_G(e)&=h_V\,q_G\tilde{p}_E(e')=h_V\tilde{p}_V\,q_D(e')=g_V\,q_D\,(e')=q_H\,g_E(e')\\ &=q_H\,h_E\tilde{p}_E(e')=q_H\,h_E(e).\end{aligned}$$

(b) $e\notin\tilde{p}_E[D_E]\Rightarrow e\in\tilde{d}_E[B_E]$: We consider e' with $\tilde{d}_E(e')=e$. Then $q_B(e')$ is defined. (Otherwise, e' has a pre-image in K_E by (1.6a) and e is in $\tilde{p}_E[D_E]$.)

$$\begin{aligned} h_V\,q_G(e)&=h_V\,q_G\,\tilde{d}_E(e')=h_V\,\tilde{d}_V\,q_B(e')=f_V\,q_B(e')=q_H\,f_E(e')\\ &=q_H\,h_E\,\tilde{d}_E(e')=q_H\,h_E(e) \quad \text{q.e.d.}\end{aligned}$$

Example 3.6. In 1.6, we must suppose p to be a graph-morphism. It is not possible to allow p to be weak. We consider the following example:

$$\begin{array}{ll} K_E=\{e_1\} & K_V=\{v_1, v_2\},\\ z_K(e_1)=v_1 & q_K(e_1)=\text{undefined},\\ B_E=\{e_1', e_2'\} & B_V=\{v_1', v_2', v_3'\},\\ z_B(e_1')=v_1' & z_B(e_2')=v_2',\\ q_B(e_1')=v_3' & q_B(e_2')=v_1',\\ p_E(e_1)=e_1' & p_V(v_1)=v_1' \qquad p_V(v_2)=v_2',\\ D_E=\{\tilde{e}_1, \tilde{e}_3\} & D_V=\{\tilde{v}_1, \tilde{v}_2, \tilde{v}_4, \tilde{v}_5\},\\ z_D(\tilde{e}_1)=\tilde{v}_1 & z_D(\tilde{e}_3)=\tilde{v}_5,\\ q_D(\tilde{e}_1)=\tilde{v}_4 & q_D(\tilde{e}_3)=\tilde{v}_2,\\ d_E(e_1)=\tilde{e}_1 & d_V(v_1)=\tilde{v}_1 \qquad d_V(v_2)=\tilde{v}_2. \end{array}$$

Then, it is not possible to construct q_H unambiguously because of $\tilde{p}_E(\tilde{e}_1)=\tilde{d}_E(\tilde{e}_1')$, but

$$q_H\tilde{p}_E(\tilde{e}_1)=\tilde{p}_V(\tilde{v}_4)\neq\tilde{d}_V(v_3')=q_H\tilde{d}_E(e_1').$$

With similar examples, one can illustrate that (1.6a) and (1.6b) are necessary. We omit the proofs.

Whereas a production of a string-grammar is applicable if its left-hand side is a substring of the given string, a production of a graph-grammar is applicable only if there exists a suitable enlargement:

Theorem 3.7. Let $K\in|\mathfrak{Graph}^-|$, $(B, m_B)\in|\mathfrak{Graph}_\Omega^-|$, $(G, m_G)\in|\mathfrak{Graph}_\Omega|$ be such that there exists a graph-morphism p: $K\to B$ and an injective weak graph$_\Omega$-morphism g: $(B, m_B)\to(G, m_G)$. Then, there is an enlargement $e=(d, (D, m_D))$ with

$$(G, m_G)=(D, m_D)\coprod_{d,p}(B, m_B)\wedge g=\tilde{d}$$

(up to isomorphism) if and only if

(a) $q_G g_E p_E[K_E]\cap g_V[B_V]\subseteq g_V p_V q_K[K_E]$

$$\wedge z_G g_E p_E[K_E]\cap g_V[B_V]\subseteq g_V p_V z_K[K_E],$$

(b) $q_G[\tilde{G}_E]\cup z_G[\tilde{G}_E]\subseteq\tilde{G}_V\cup g_V p_V[K_V]$, where

$$\tilde{G}_E:=G_E\setminus g_E[B_E],\quad \tilde{G}_V:=G_V\setminus g_V[B_V].$$

To prove this theorem,

(i) we construct a complete graph D, together with m_D and d: $K\to D$ (see No. 3.9),

(ii) we must show that

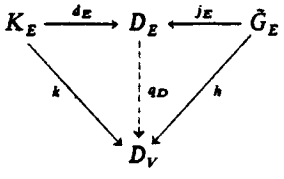
$$(G, m_G)=(D, m_D)\coprod_{d,p}(B, m_B)\quad\text{and}\quad g=\tilde{d}$$

up to isomorphism (see No. 3.10),

(iii) we must show the gluing condition to be necessary, but this is straightforward and will be left to the reader.

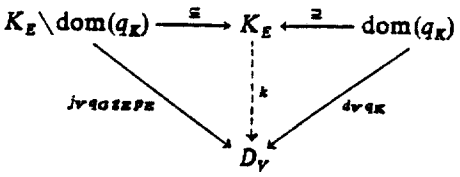
Remark 3.8. In the case of a contextfree production, the gluing condition (3.7b) is trivial on the left-hand side because of $g_V['B_V]=g_V\cdot{}'p_V[K_V]$.

Construction 3.9. Consider $D_E:=\tilde{G}_E\cup K_E$ and $D_V:=\tilde{G}_V\cup K_V$ with the "natural" inclusions d_E: $K_E\to D_E$, d_V: $K_V\to D_V$, j_E: $\tilde{G}_E\to D_E$, j_V: $\tilde{G}\to D_V$. Then



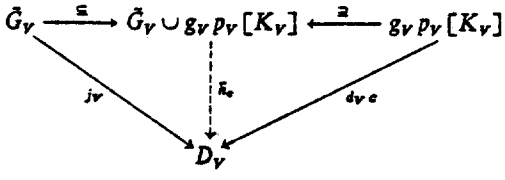
is a coproduct-diagram: if k and h are given, there is a unique q_D with $q_D d_E=k\wedge q_D j_E=h$.

(a) To construct k, we consider another coproduct-diagram:



If e is not in the domain of q_K, $q_G g_E p_E(e)$ is in $\tilde{G}_V$. (Otherwise, it would be in $g_V[B_V]$, and therefore, it is an element of $g_V p_V q_K[K_E]$ because of (3.7a) in contrary to the assumption.) Thus, j_V yields a node of D_V by construction. Note that $d:=(d_E, d_V)$ is a weak graph-morphism because of $d_V q_K = k = q_D d_E$ on the domain of q_K.

(b) In order to construct h, we consider the restriction $r: K_V \to g_V p_V[K_V]$ of $g_V p_V$ and an arbitrary coretraction c of r. (By definition c is a coretraction of r if rc is the identity on $g_V p_V[K_V]$.) Then, we define $h:=\bar{h}_c q_G$, where $\bar{h}_c$ is the resulting morphism in the following coproduct-diagram:



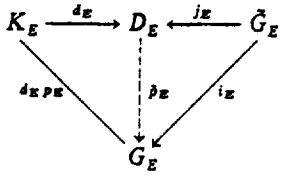
Because of condition (3.7b), $\bar{h}_c q_G$ is defined. With a similar construction for z_D, we have a total graph D.

Finally, we must define the labelling of D:

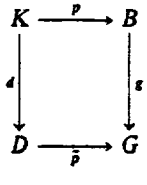
$$m_D(e) := \begin{cases} m_G(e) & \text{if } e \in \tilde{G}_E \\ \text{undefined} & \text{otherwise} \end{cases}$$

and $m_D(v)$ analoguously.

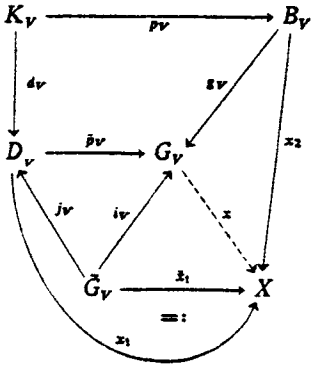
3.10. Using the inclusions $i_E: \tilde{G}_E \to G_E$ and $i_V: \tilde{G}_V \to G_V$, we define $\tilde{p}_E$ by



and $\tilde{p}_V$ in the same way. It is easy to see that $\tilde{p}$ is a graph-morphism and d a weak graph-morphism. To show that



(with the above constructed $D, d, \tilde{p}$) is a pushout-diagram it is sufficient to prove the pushout-property for the V- and E-components in $\mathfrak{Sets}$. Let us consider the V-component: Given $x_1: D_V \to X$ and $x_2: B_V \to X$ with some set X and $x_1 d_V = x_2 p_V$, we must prove existence of a unique $x: G_V \to X$ with $x \tilde{p}_V = x_1 \wedge x g_V = x_2$. Consider the following diagram:



$\tilde{x}_1$ is defined by $x_1 j_V$. Because of the coproduct-property, there is a unique x with $x g_V = x_2 \wedge x i_V = \tilde{x}_1$. Then, we have

$$x \tilde{p}_V j_V = x i_V = \tilde{x}_1 = x_1 j_V,$$
$$x \tilde{p}_V d_V = x g_V p_V = x_2 p_V = x_1 d_V.$$

Since j_V and d_V are coproduct-injections of D_V, we get $x \tilde{p}_V = x_1$. It is easy to show that this is the only way to construct x: If there exists another x' with $x' \tilde{p}_V = x_1$, and $x' g_V = x_2$, we have also $x' i_V = x' \tilde{p}_V j_V = x_1 j_V = \tilde{x}_1$ and hence $x = x'$ by uniqueness of x.

Since we have $m_D = m_G \cdot U\tilde{p} \cdot \leq$ by construction in 3.9, the diagram in No. 1.7, defining m_G, commutes showing that $\tilde{p} \in \mathrm{Mor}\ \mathfrak{Graph}_\Omega$- and m_G is the labelling of $(D, m_D) \coprod_{d,p} (B, m_B)$.

If we have a production $(('B, 'm), 'p, K, p', (B', m'))$ and $g: ('B, 'm) \to (G, m_G)$ satisfying (3.7a, b), then existence of an (H, m_H) with $(G, m_G) \to (H, m_H)$ follows immediately from the existence of an enlargement. But this (H, m_H) is not uniquely determined:

Definition 3.10a. $v, v' \in D_V$ are called *p-equivalent* if either $v = v'$, or there exist $k, k' \in K_V$ with $p_V(k) = p_V(k')$, $v = d_V(k')$, and $v' = d_V(k')$.

Theorem 3.11. If the assumptions of 3.7 are satisfied and if $(d_1, (D_1, m_{D_1}))$ and $(d_2, (D_2, m_{D_2}))$ are enlargements, then there is a pair of bijections $b = (b_E, b_V)$: $UD_1 \to UD_2$ with $b \cdot UD_1 = Ud_2 \wedge m_{D_1} = m_{D_2} \cdot b|_{D_1}$ such that b is a graph-morphism up to p-equivalence of nodes.

Remark. Proposition 3.3 in [3] does not remain true if we allow partial graphs, because it is not possible to simulate assumption (1.6b) without edges in K.

Theorem 3.12. If the assumptions of 3.7 and 3.11 are satisfied and p is injective, then b is an isomorphism in $\mathfrak{Graph}$.

In this case, the enlargement—therefore (H, m_H), too—is unique up to isomorphism. The proofs of 3.11 and 3.12 are the same as in [3].

4. Convex Graph-Grammars

Following an idea of J. Pfaltz, we consider now the replacement of convex subgraphs. However, we use a more general definition as in [9]:

Definition 4.1. A *graph-morphism* $d\colon K \to D$ is called *convex* if and only if it satisfies

$$(\forall e_0, e_1, \ldots, e_n \in D_E)\,(q_D(e_0) \in d_V[K_V] \wedge z_D(e_n) \in d_V[K_V]$$
$$\wedge\, (\forall i)\,(1 \leqq i \leqq n \Rightarrow z_D(e_{i-1}) = q_D(e_i))$$
$$\Rightarrow (\forall i)\,(0 \leqq i \leqq n)\,(e_i \in d_E[K_E])).$$

This means that the edges of a path the first and the last node of which are images of nodes of K, are images of edges of K, too. Thus, there are no new paths in D between the images of nodes of K. If d is injective, K is a convex subgraph of D in the usual sense. Note that we allow $q(e_i) = z(e_i)$.

Corollary 4.2. If $d\colon K \to D$ is convex, we have

$$(\forall i)\,(0 \leqq i \leqq n \Rightarrow q_D(e_i) \in d_V[K_V] \wedge z_D(e_i) \in d_V[K_V])$$

with the above e_i.

Consider $e_i' \in K_E$ with $d_E(e_i') = e_i$, $q_K(e_i')$, and $z_K(e_i')$. Then, we have e.g.: $d_V(q_K(e_i')) = q_D(d_E(e_i')) = q_D(e_i)$.

In Definition 4.1, we have not assumed the pre-images of the edges to form a path:

Corollary 4.3. If $d\colon K \to D$ is convex and injective, then the e_i' with $d_E(e_i') = e_i$ form a path.

Because of injectivity $d_V(q_K(e_i')) = q_D(e_i) = z_D(e_{i-1}) = d_V(z_K(e_{i-1}'))$ yields $q_K(e_i') = z_K(e_{i-1}')$.

Lemma 4.4. If $d\colon K \to D$ and $c\colon D \to C$ are convex and injective, then $c \cdot d$ is convex and injective.

We must only prove convexity. Consider $e_0, e_1, \ldots, e_n \in C_E$ with

$$q_C(e_0) \in c_V\, d_V[K_V] \wedge (\forall i)\,(1 \leqq i \leqq n \Rightarrow z_C(e_{i-1}) = q_C(e_i)) \wedge z_C(e_n) \in c_V\, d_V[K_V].$$

Because of the convexity of c, we have

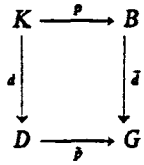
$$(\forall i)\,(0 \leqq i \leqq n \Rightarrow e_i \in c_E[D_E]).$$

c is injective. Therefore, the pre-images e_i' form a path with

$$q_D(e_0') \in d_V[K_V] \wedge z_D(e_n') \in d_V[K_V].$$

Convexity of d yields the proposition.

Theorem 4.5. If



is a pushout-diagram in $\mathfrak{Graph}$ with d injective, then $\tilde{d}$ is convex if and only if d is convex.

This result corresponds to Theorem 8 and 10 in [9].

Proof of (4.5) 4.6. Given $e_0, e_1, \ldots, e_n \in G_E (n \geqq 0)$ with

$$q_G(e_0) \in \tilde{d}_V[B_V] \wedge (\forall i)(1 \leqq i \leqq n \Rightarrow z_G(e_{i-1}) = q_G(e_i)) \wedge z_G(e_n) \in \tilde{d}_V[B_V],$$

we must only show

$$(\forall i)(0 \leqq i \leqq n \Rightarrow e_i \in \tilde{d}_E[B_E])$$

because of Corollary 4.3. Without loss of generally, we assume furthermore

$$(\forall i)(1 \leqq i \leqq n \Rightarrow q_G(e_i) \notin \tilde{d}_V[B_V]). \qquad (*)$$

Otherwise the path in G may be divided into subpaths such that each subpath satisfies the assumptions of 4.5 and (*). From (*) it follows that $e_i \notin \tilde{d}_E(B_E)$. In other words, there is $e_i' \in D_E$ with $\tilde{p}_E(e_i') = e_i$. (Possibly e_i' is not uniquely determined, but this is not important.) Then, we have:

$$\tilde{p}_V(q_D(e_i')) = q_{G_1}(e_i) \in \tilde{p}_V[D_V] \wedge \tilde{p}_V(z_D(e_i')) = z_G(e_i) \in \tilde{p}_V[D_V]. \qquad (**)$$

We consider the following cases:

(a) $(\forall i)(1 \leqq i \leqq n \Rightarrow z_D(e_{i-1}') = q_D(e_i'))$,

(b) $(\exists j)(1 \leqq j \leqq n \wedge z_D(e_{j-1}') \neq q_D(e_j'))$.

(a) In this case, $e_0', e_1', \ldots, e_n'$ are a path from $q_D(e_0') \in d_V[K_V]$ because of $q_G(e_0) \in \tilde{p}_V[D_V] \cap \tilde{d}_V[B_V]$, (see No. 3.1 b) to $z_D(e_n') \in d_V[K_V]$. Since d is convex, there exist $e_i'' \in K_E$ with $d_E(e_i'') = e_i'$. We consider $\tilde{d}_E(p_E(e_i'')) = \tilde{p}_E(d_E(e_i'')) = e_i$. It follows that $e_i \in \tilde{d}_E[B_E]$ for all i showing the assertion. Particularly, this holds for $n = 0$.

(b) We assume j to be minimal, thus $z_D(e_{i-1}') = q_D(e_i')$ for $i < j$. From $z_D(e_{j-1}') \neq q_D(e_j') \wedge \tilde{p}_V(z_D(e_{j-1}')) = \tilde{p}_V(q_D(e_j'))$ and Lemma 3.1 (c) follows the existence of $v_j, v_j' \in K_V$ with $d_V(v_j) = q_D(e_j') \wedge d_V(v_j') = z_D(e_{j-1}') \wedge p_V(v_j) = p_V(v_j')$. We consider

$$\tilde{d}_V(p_V(v_j)) = \tilde{p}_V(d_V(v_j)) = \tilde{p}_V(q_D(e_j')) = q_G(e_j) \quad \text{by (**)}$$
$$\Rightarrow q_G(e_j) = z_G(e_{j-1}) \in \tilde{d}_V[B_V].$$

Therefore, the subpath $e_0, e_1, \ldots, e_{j-1}$ with $j \geqq 1$ satisfies (a). On the other hand, $e_j, e_{j+1}, \ldots, e_n$ either satisfies (a), too, or (b) may be repeated.

Proof of 4.5 (Converse) 4.7. Given $e_0, e_1, \ldots, e_n \in D_E$ with $q_D(e_0), z_D(e_n) \in d_V[K_V]$ and $(\forall i)(1 \leqq i \leqq n \Rightarrow z_D(e_{i-1}) = q_D(e_i))$, we must show $(\forall i)(0 \leqq i \leqq n \Rightarrow e_i \in d_E[K_E])$.

We have $v, v' \in K_V$ with $d_V(v) = q_D(e_0) \wedge d_V(v') = z_D(e_n)$. From

$$\tilde{d}_V(p_V(v)) = \tilde{p}_V(d_V(v)) = \tilde{p}_V(q_D(e_0)) = q_G(\tilde{p}_E(e_0)) \in \tilde{d}_V[B_V]$$
$$\tilde{d}_V(p_V(v')) = z_G(\tilde{p}_E(e_n)) \in \tilde{d}_V[B_V]$$

and

$$(\forall i)\,(0 \leqq i \leqq n \Rightarrow \tilde{p}_E(e_i) \in G_E)$$
$$(\forall i)\,(1 \leqq i \leqq n \Rightarrow z_G(\tilde{p}_E(e_{i-1})) = \tilde{p}_V(z_D(e_{i-1})) = \tilde{p}_V(q_D(e_i)) = q_G(\tilde{p}_E(e_i)))$$

and the convexity of $\tilde{d}$: $B \to G$ it follows that all $\tilde{p}_E(e_i)$ are images of edges in B_E and of edges in K_E, too (Lemma 3.1 (b)).

Theorem 4.5 enables us to characterize grammars the derivations of which replace convex subgraphs only:

Definition 4.8. A *graph-grammar* Q with enlargements is called *convex* if and only if for all productions p of Q E_p contains convex enlargements d: $K \to D$ only.

This classification into convex and non-convex graph-grammars may be used in addition to classifications considered in [3]. Especially, contextfree graph-grammars are convex by definition if there are no cycles, because in this case d is convex and hence also $\tilde{d}$ by Theorem 4.5.

Theorem 4.9. Let G be cycle-free, (p, e) a production and an enlargement of a convex graph-grammar, and $G \xrightarrow[(p,e)]{} H$, then H is cycle-free if and only if the right-hand side B' of the production is cycle-free.

This result corresponds to Theorem 1 in [13].

Proof. Let H contain a cycle. If this cycle doesn't possess a common node v with B', then G contains already this cycle. Otherwise, there is a path in H from v to v totally in B' because of convexity, thus B' is not cycle-free. The converse is trivial.

Acknowledgments. The authors would like to thank K. S. Vijayan and the unknown reviewers for their helpful suggestions.

References

1. Denert, E., Franck, R., Streng, W.: PLAN2D — Ein Ansatz zur Definition einer zweidimensionalen Programmiersprache. Ges. f. Informatik, Jahrestagung 1974, Berlin. Lecture Notes in Computer Science 26. Berlin-Heidelberg-New York: Springer 1975. p. 202–213
2. Ehrig, H., Pfender, M., Schneider, H. J.: Kategorielle Konstruktionen in der Theorie der Graph-Grammatiken. Arb.berichte Inst. Math. Masch. Dat.verarb., Univ. Erlangen, vol. 6, No. 3 (1973), p. 30–55
3. Ehrig, H., Pfender, M., Schneider, H. J.: Graph-Grammars—an algebraic approach. Proc. Conf. Switch. Automat. Theory 1973, p. 167–180
4. Ehrig, H., Tischer, K. W.: Graph-grammars applied to the specialization of organisms. Proc. Conf. Biologically Motivated Automata Theory, McLean (Virginia) 1974, p. 158–165
5. Milgram, D., Rosenfeld, A.: Array automata and array grammars. Proc. IFIP Congress 1971, Amsterdam: Noth-Holland 1972, p. 69–74
6. Nagl, M.: Eine Präzisierung des Pfaltz-Rosenfeldschen Produktionsbegriffes bei mehrdimensionalen Grammatiken. Arb.berichte Inst. Math. Masch. Dat.verarb., Univ. Erlangen, vol. 6, No. 3 (1973), p. 56–71

7. Nagl, M.: Formale Sprachen von markierten Graphen. Arb.berichte Inst. Math. Masch. Dat.verarb., Univ. Erlangen, vol. 7, No. 4 (1974)
8. Pfaltz, J. L., Rosenfeld, A.: Web grammars. Proc. Int. Joint Conf. Artifical Intelligence, Washington 1969, p. 609–619
9. Pfaltz, J. L.: Convexity in directed graphs. J. Combinat. Theory **10**, 143–162 (1971)
10. Pratt, T. W.: Pair grammars, graph languages, and string-to-graph translation. J. Computer System Sciences **5**, 560–595 (1971)
11. Rosen, B. K.: Deriving graphs from graphs by applying a production. Acta Informatica **4**, 337–357 (1975)
12. Schneider, H. J.: Chomsky-Systeme für partielle Ordnungen. Arb.berichte Inst. Math. Masch. Dat.verarb., Univ. Erlangen, vol. 3, No. 3 (1970)
13. Schneider, H. J.: A necessary and sufficient condition for Chomsky-productions over partially ordered symbol sets. Ges. f. Informatik, Jahrestagung 1972. Lecture Notes in Economics and Mathematical Systems **78**. Berlin-Heidelberg-New York: Springer 1973
14. Schneider, H. J.: Syntax-directed description of incremental compilers. Gesellschaft für Informatik, Jahrestagung 1974, Berlin. Lecture Notes in Computer Science **26**, Berlin-Heidelberg-New York: Springer 1975, p. 192–201
15. Encarnacao, J.: Datenstrukturen für graphische Informationsverarbeitung. Gesellschaft für Informatik, Bericht No. 2, Fachtagung Computer Graphics, Berlin 1971, p. 15–49

H. J. Schneider
Institut für Math. Maschinen
und Datenverarbeitung (II)
der Univ. Erlangen-Nürnberg
Egerlandstr. 13
D-8520 Erlangen
Federal Republic of Germany

H. Ehrig
Forschungsgruppe Automatentheorie
und Formale Sprachen der
TU Berlin
Otto-Suhr-Allee 18–20
D-1000 Berlin 10