Amalgamation of Graph Transformations with Applications to Synchronization

Paul Boehm

Harald-Reto Fonio

Annegret Habel

Technische Universität Berlin

D-1000 Berlin 10, Franklinstr. 28/29

Abstract: In the present paper we generalize the well-known PARALLELISM THEOREM for graph derivations to the AMALGAMATION THEOREM. In this theorem the assumption of "parallel independence" is dropped. For each pair of productions together with a relational production (allowing productions to be associated with each other) we construct a single "amalgamated" production. The AMALGAMATION THEOREM states that graph derivations which respect the given associations can be amalgamated to a single derivation via the "amalgamated" production.
The amalgamation mechanism can be used to handle synchronization phenomena. The amalgamation concept is applied to synchronization of graph manipulations in a simplified railway control system as well as in GDS, a graph grammar formalism for distributed systems.

1. Introduction

Graphs and transformations of graphs are important in many areas of computer science (see /CER 79/ and /ENR 83/). There are many different ways how to generalize string productions and derivations to graphs. A survey over several approaches including an extensive bibliography is given in /Na 79/. This paper is based on the gluing approach defined in /EPS 73/ and /Ro 75/ and extensively described in /Eh 79/.

Some of the most fundamental concepts of graph transformation theory are graph productions and graph derivations. One important class of problems and phenomena can be characterized by the following situation:
Given an initial graph, there can be several productions applicable. What to do now? There are several interesting possibilities, e.g. to use some productions in parallel, or to use a well-defined sequence of productions, or to use some specific "composed" productions, or... . These and similar phenomena are investigated under the headings parallelism, sequentialization, concurrency (see e.g. /ER 79a/, /EK 80/, /JKRE 82/, /Eh 83/, /Pe 80/), e.t.c.. One further concept belonging to this class is amalgamation which is the topic of this paper.

In the present paper we generalize the well-known PARALLELISM THEOREM for graph derivations to the AMALGAMATION THEOREM. In both theorems we consider a graph G and productions p1 and p2 which are applicable to G. If the corresponding graph derivations $G \Longrightarrow H1$ via p1 and $G \Longrightarrow H2$ via p2 are "parallel independent", i.e. the occurrences of p1 and p2 in G are allowed to overlap in items which remain preserved in each derivation, the PARALLELISM THEOREM states that the productions can be applied

one after the other or "in parallel". In general p1 and p2 may have common items which remain not preserved. Then the productions p1 and p2 only can be applied "synchronously" , provided that the derivations via p1 and p2 are "amalgamable", i.e. that the occurrences of p1 and p2 in G are allowed to overlap in items which shall be preserved - or deleted - by both productions. For each pair of productions (which are allowed to possess a common part) we construct a single "amalgamated" production. The AMALGAMATION THEOREM states that "amalgamable" graph derivations can be amalgamated to a single derivation. Applying the amalgamated production of p1 and p2 has essentially the same effect as applying first the common part of p1 and p2 and then the remainders of p1 and p2.

Let us illustrate our AMALGAMATION CONCEPT by a simplified example of a railway control system: States in the railway system are represented by graphs, tracks and trains are some types of nodes. "Blowprints" for changes of states are described by graph productions and the actual changes of states by graph derivations. Here the main problem is whether planned changes for a subnet are consistent with those of other subnets, i.e., with respect to the topic of this paper, whether plans for subnets can be amalgamated to a single plan for the whole railway net.
Consider the elementary productions MOVE and HALT given in Fig. 1.1 which are part of the small railway system studied in /MW 82/. More precisely MOVE and HALT are production rules where the node labels have to be recolored by actual parameters.

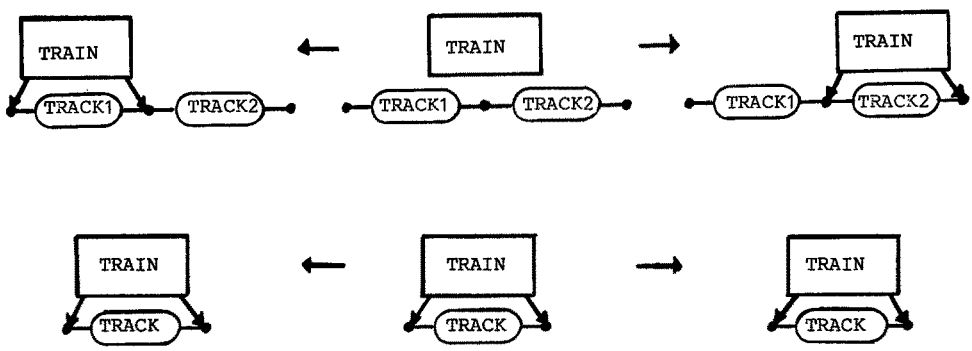


Figure 1.1: Productions for moving and halting of trains

Moving a train means to apply the production rule MOVE with suitable actual parameters to the current state of the railway system which is represented as a graph.

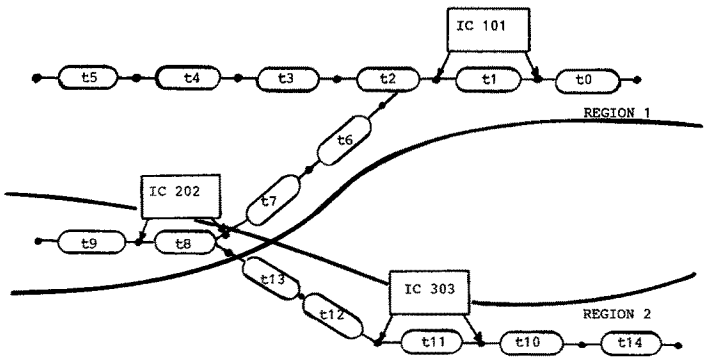


Figure 1.2: Graph representing the current state of the railway system

The railway net is covered by regions which are controlled by controllers. Each controller designs a plan for his region (controller is meant to be a person, not to be a system). A plan is represented as a complex production p consisting of an elementary production for each train in the region. A plan may be executed if it harmonizes with the plans for the adjacent regions. Now there may be one of the following situations:

Situation 1: The controller of region 1 has the plan that the train IC 101 moves from the track t1 to the turnout t2 (we assume that the turnout t2 is directed to t3) and IC 202 halts on t8. The controller of region 2 wants that IC 202 halts on t8 and that IC 303 moves from t11 to t12. In this case the plans for region 1 and region 2 harmonize and can be executed one after the other or "in parallel".

Situation 2: The controller of region 1 has the plan that IC 101 moves from t1 to t2 and IC 202 moves from t8 to t9. The controller of region 2 has a plan which harmonizes with the first plan: He wants that IC 202 moves from t8 to t9 and that IC 303 moves from t11 to t12. These plans cannot be executed one after the other, but only "synchronously".

The plans for the single regions are represented by

p1=MOVE (IC 101,t1 ,t2)+MOVE (IC 202,t8,t9),

p2=MOVE (IC 303,t11,t12)+MOVE (IC 202,t8,t9).

The plan for the region of common control is represented by

r=MOVE (IC 202,t8,t9).

The plan for the whole region is an amalgamation of the single plans with respect to r. It is represented by the r-amalgamated production

$p \oplus_r p'$=MOVE(IC 101,t1,t2)+MOVE(IC 202,t8,t9)+MOVE(IC 303,t11,t12).

Applying this production to the railway graph G we obtain a derivation $G \Longrightarrow X$ via $p \oplus_r p'$ which can be seen as an amalgamation of the derivations $G \Longrightarrow H1$ via p1 and $G \Longrightarrow H2$ via p2. The AMALGAMATION THEOREM applied to this example states that executing the plan $p1 \oplus_r p2$ has essentially the same effect as first executing the plan p1

and than executing the remainder of plan p2.

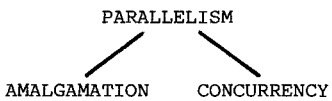
Situation 3: The controller of region 1 has the following plan: He wants that IC 101 moves from t1 to t2 and IC 202 halts on t8. The controller of region 2 has a conflicting plan: He wants that the train IC 202 leaves the turnout t8 and moves to t7 (such that the turnout becomes free and may be switched) and that train IC 303 moves from t11 to t12. These plans are represented by the complex productions

p1=MOVE(IC 101,t1 ,t2)+HALT(IC 202,t8)

p2=MOVE(IC 303,t11,t12)+MOVE(IC 202,t8,t7).

In this case the uniquely determined occurrences of p1 and p2 in the railway graph G given in Fig. 1.2 overlap in a not-allowed way: The edges determining the location of the train IC 202 at turnout t8 shall be preserved by production p1 and deleted by production p2, i.e. the occurrences of p1 and p2 in G are in "conflict". Such conflict cases will not be treated in the present paper.

The AMALGAMATION THEOREM, the main result of this paper, is formulated and proved in the framework of the algebraic theory of graph grammars using pushout and pullback constructions in the category of labeled graphs.



The AMALGAMATION THEOREM generalizes the well-known PARALLELISM THEOREM considerably. The CONCURRENCY THEOREM presented e.g. in /ER 79a/ and /EHR 83/ can be seen as another type of generalization of the PARALLELISM THEOREM. Although the amalgamation and the concurrency concepts are different it is shown how we can profit from the CONCURRENCY THEOREM in the proof of the AMALGAMATION THEOREM.

Some basic notions from the theory of graph grammars are reviewed in Section 2. The main result is stated in Section 3 and proved in Section 4. Finally an application of the amalgamation concept to special synchronization problems in distributed systems is discussed in Section 5. Due to the limitation of space the proofs of the technical lemmata are omitted. A complete formal treatment can be found in /BFH 84/.

Acknowledgements

We would like to thank Hartmut Ehrig and Hans-Jörg Kreowski for fruitful discussions. For excellent typing we are most grateful to H. Barnewitz.

2. Preliminaries

This section provides some basic notions concerning graphs, graph productions and graph derivations used in the following. For more details we refer to the tutorial survey /Eh 79/.

The object of our considerations are directed, labeled graphs over a fixed pair of labeling alphabets $C=(C_N,C_A)$. A graph $G=(G_N,G_A,s,t,m_N,m_A)$ over (C_N,C_A) consists of a set of nodes G_N, a set of arcs G_A, maps $s,t:G_A \to G_N$ assigning source and target to each arc of G, and labeling maps $m_N:G_N \to C_N$, $m_A:G_A \to C_A$ assigning a node labeling to each node and an arc labeling to each arc of G.

Given two graphs G and G' a graph morphism $f:G \to G'$ is a pair of maps $(f_N:G_N \to G'_N, f_A:G_A \to G'_A)$ satisfying $f_N s=s' f_A$, $f_N t=t' f_A$, $m'_N f_N=m_N$, and $m'_A f_A=m_A$ (composition of maps). f is called injective if the maps f_N and f_A both are injective. The composition of graph morphisms $f:G \to G'$ and $f':G' \to G''$ is defined by the composition of their components.

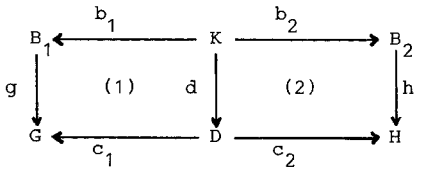
Following /Eh 79/ a graph production $p=(B_1 \xleftarrow{b_1} K \xrightarrow{b_2} B_2)$ consists of a pair of graphs (B_1,B_2) and an auxiliary graph K called gluing graph, which is related to B_1 and B_2 by injective graph morphisms $b_1:K \to B_1$, $b_2:K \to B_2$.

Given a graph G, a graph production $p=(B_1 \leftarrow K \rightarrow B_2)$ and a graph morphism $g:B_1 \to G$, p is applicable to G with respect to $g:B_1 \to G$ if the gluing condition

$$\text{BOUNDARY}(B_1 \to G) \subseteq b_1 K$$

is satisfied. [$\text{BOUNDARY}(B_1 \to G)$ consists of dangling items, i.e. items of $\text{DANGLING}(B_1 \to G)=\{n \in B_{1N} \mid \exists a \in (G-gB_1)_A : gn=sa \text{ or } gn=ta\}$ and identification items, i.e. items of $\text{IDENTIFICATION}(B_1 \to G)=\{x \in B_1 \mid \exists y \in B_1 : x \neq y \text{ and } gx=gy\}$].

In this case the "context" graph D can be constructed in such a way that G becomes the gluing of D and B_1 along the gluing graph K. Now the result of the application H of p to G with respect to $g:B_1 \to G$ is the gluing of D and B_2 along the gluing graph K.



We write $G \overset{p}{\Longrightarrow} H$ and say that $g:B_1 \to G$ is the occurrence of p in G. $G \Longrightarrow H$ is also called direct derivation via p based on g. (Note that G and H are pushout objects in the diagrams (1) and (2) constructed in the category of labeled graphs. We will often use the short notations PO and PB for pushouts and pullbacks (/AM 75/). Moreover, we will use a linear notation for squares if the notion of morphisms is not essential, e.g. PO (1) above will be written as KB_1DG or KDB_1G.

Two direct derivations $G \Rightarrow H$ via p based on g and $G \Rightarrow H'$ via p' based on g' are parallel independent if the intersection of B_1 and B'_1 in G (which are the occurrences of p and p' in G) consists of common gluing items only, that means

$$gB_1 \cap g'B'_1 \subseteq gb_1 K \cap g'b'_1 K'$$

On the other hand we can construct the parallel production of p and p'

$$p+p'=(B_1+B'_1 \leftarrow K+K' \rightarrow B_2+B'_2)$$

built up by componentwise disjoint union from the single productions

$p=(B_1 \leftarrow K \rightarrow B_2)$ and $p'=(B_1' \leftarrow K' \rightarrow B_2')$. Derivations via parallel productions are called parallel derivations.

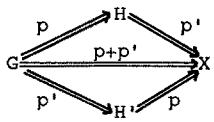
The connection between parallel independent derivations and parallel derivations is established in the PARALLELISM THEOREM (see /ER 79a/):

PARALLELISM THEOREM

Let p and p' be productions and p+p' the corresponding parallel production. Then we have

1. SYNTHESIS Given parallel independent derivations $G \Rightarrow H$ via p and $G \Rightarrow H'$ via p' then there are a graph X, direct derivations $H \Rightarrow X$ via p' and $H' \Rightarrow X$ via p and a parallel derivation $G \Rightarrow X$ via p+p'.

2. ANALYSIS Given a parallel derivation $G \Rightarrow X$ via p+p' then there are derivation sequences $G \Rightarrow H \Rightarrow X$ via (p,p') and $G \Rightarrow H' \Rightarrow X$ via (p',p) such that the direct derivations $G \Rightarrow H$ via p and $G \Rightarrow H'$ via p' are parallel independent.



3. The operations SYNTHESIS and ANALYSIS are inverse to each other in the following sense: there is a bijective correspondence between parallel independent derivations and parallel derivations.

3. Amalgamation of Transformations

In this section we will study the problem of "amalgamating" direct transformations to a single transformation using only one "amalgamated" production. Applying the amalgamated production the original productions shall be executed at the same time and hence synchronously.

We introduce the notion of a relational production r for productions, r-amalgamable derivations, and the construction of r-amalgamated productions and derivations. The connection between r-amalgamable derivations and r-amalgamated derivations is established in the Amalgamation Theorem which will be stated together with a number of corollaries.

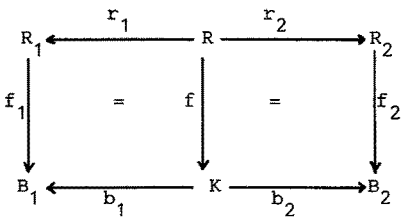
3.1 DEFINITION

1. Given a production $p=(B_1 \leftarrow K \rightarrow B_2)$ a production $r=(R_1 \leftarrow R \rightarrow R_2)$ together with graph morphisms $R_1 \rightarrow B_1$, $R \rightarrow K$, $R_2 \rightarrow B_2$ is called a subproduction of p if the diagrams RR_1KB_1 and RR_2KB_2 commute and the conditions

(1) $\text{BOUNDARY}(R_1 \rightarrow B_1) \subseteq r_1R$ and $\text{IDENTIFICATION}(R_2 \rightarrow B_2) \subseteq r_2R$

(2) $f_1^{-1} b_1K \subseteq r_1R$ and $f_2^{-1} b_2K \subseteq r_2R$

are satisfied.



2. Given two productions $p=(B_1 \leftarrow K \rightarrow B_2)$, $p'=(B_1' \leftarrow K' \rightarrow B_2')$ a production $r=(R_1 \leftarrow R \rightarrow R_2)$ together with graph morphisms $B_1 \leftarrow R_1 \rightarrow B_1'$, $K \leftarrow R \rightarrow K'$, $B_2 \leftarrow R_2 \rightarrow B_2'$ is called a relational production for p and p' if r is a subproduction of p as well as p'.

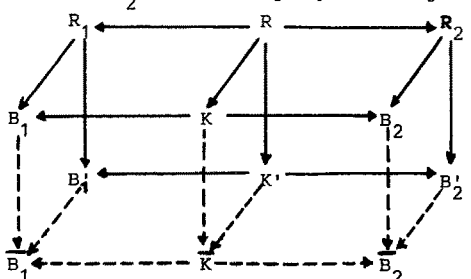
3. Let p and p' be productions and r be a relational production for p,p'. Then the r-amalgamated production $\bar{p}$ is defined by the following construction:

(1) Let $\bar{B}_1$ be the gluing of B_1 and B_1' along R_1,

(2) let $\bar{K}$ be the gluing of K and K' along R,

(3) and $\bar{B}_2$ be the gluing of B_2 and B_2' along R_2.

Moreover let $\bar{K} \rightarrow \bar{B}_1$ and $\bar{K} \rightarrow \bar{B}_2$ be the uniquely existing morphisms.



Then $\bar{p}=(\bar{B}_1 \leftarrow \bar{K} \rightarrow \bar{B}_2)$ is called amalgamation of p and p' with respect to r, written $p \oplus_r p'$. $\bar{p}$ also is called r-amalgamated production. A direct derivation $G \Rightarrow X$ via $\bar{p}$ is called r-amalgamated derivation.

REMARKS

1. Let p be a production and r be a subproduction of p. Then there is a uniquely determined production p_o and a relation S for (r,p_o) such that $p=r*_S p_o$. The production p_o is called remainder of p with respect to r.

2. The relational production r relates the productions p and p'. In the case $r=\emptyset$, i.e. the empty production $(\emptyset \leftarrow \emptyset \rightarrow \emptyset)$, the amalgamation of p and p' w.r.t. r is equal to the parallel production $p+p'=(B_1+B_1' \leftarrow K+K' \rightarrow B_2+B_2')$.

3. The construction of r-amalgamated productions and derivations can be iterated.

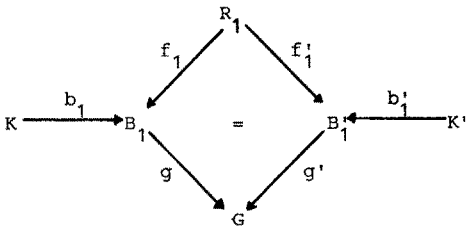
3.2 DEFINITION

1. Given two productions $p=(B_1 \leftarrow K \rightarrow B_2)$ and $p'=(B_1' \leftarrow K' \rightarrow B_2')$ two direct derivations $G \Rightarrow H$ via p based on g and $G \Rightarrow H'$ via p' based on g' are called amalgamable if the intersection of B_1 and B_1' in G (which are the occurrences of p and p' in G) consists of common gluing items or items which shall be deleted by p as well as by p'. That means precisely

$gB_1 \cap g'B_1' \subseteq (g(B_1-b_1K) \cap g'(B_1'-b_1'K')) \cup (gb_1K \cap g'b_1'K')$.

2. Let r be a relational production for p and p'. Two direct derivations $G \Rightarrow H$ via p based on g and $G \Rightarrow H'$ via p' based on g' are called r-amalgamable if the diagram $R_1B_1B_1'G$ commutes and

$gB_1 \cap g'B_1' \subseteq gf_1R_1 \cup (gb_1K \cap g'b_1'K')$.



REMARKS: 1. Amalgamability generalizes parallel independency. 2. r-amalgamable derivations are amalgamable.

3.3 LEMMA

Let p and p' be productions and $G \Rightarrow H$ via p, $G \Rightarrow H'$ via p' be amalgamable direct derivations. Then there is a relational production r for p and p', such that the given direct derivations become r-amalgamable.

CONSTRUCTION

The relational production $r=(R_1 \leftarrow R \rightarrow R_2)$ can be constructed in the following way:

(1) Let R_1 be the PB-object of $B_1 \rightarrow G \leftarrow B_1'$.

(2) Let R be the PB-object of $K \rightarrow G \leftarrow K'$ where $K \rightarrow G = K \rightarrow B_1 \rightarrow G$ and $K' \rightarrow G = K' \rightarrow B_1' \rightarrow G$. (Note that there is a uniquely determined morphism $R \rightarrow R_1$ such that the diagrams RR_1KB_1 and RR_2KB_2 commute.)

(3) Let $R_2=R$ and $R \rightarrow B_2 = R \rightarrow K \rightarrow B_2$, $R \rightarrow B_2' = R \rightarrow K' \rightarrow B_2'$.

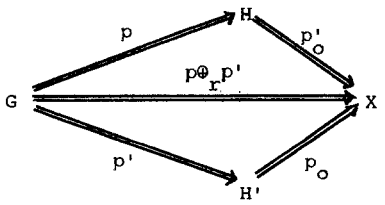
Now we will state the main theorem of this paper:

3.4 AMALGAMATION THEOREM

Let r be a relational production for productions p,p' and $\bar{p}=p\Theta_r p'$ the corresponding r-amalgamated production. Moreover, let p_o, p_o' be the remainders of p resp. p' w.r.t. r. Then we have

1. SYNTHESIS Given r-amalgamable direct derivations $G \Rightarrow H$ via p and $G \Rightarrow H'$ via p' then there are a graph X, direct derivations $H \Rightarrow X$ via p_o' $H' \Rightarrow X$ via p_o, and an r-amalgamated derivation $G \Rightarrow X$ via $p\Theta_r p'$.

2. ANALYSIS Given an r-amalgamated derivation $G \Rightarrow X$ via $p\Theta_r p'$ then there are derivation sequences $G \Rightarrow H \Rightarrow X$ via (p,p_o') and $G \Rightarrow H' \Rightarrow X$ via (p',p_o) such that the direct derivations $G \Rightarrow H$ via p and $G \Rightarrow H'$ via p' are r-amalgamable.



3. The operations SYNTHESIS and ANALYSIS are inverse to each other in the following sense: There is a bijective correspondence between r-amalgamable and r-amalgamated derivations.

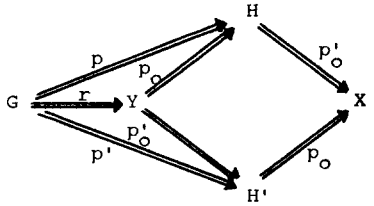
The proof of the AMALGAMATION THEOREM and some useful lemmata are given in Sect.4. We conclude the present section with some corollaries.

3.5 COROLLARY 1

Taking the empty production as relational production in the AMALGAMATION THEOREM we obtain the PARALLELISM THEOREM.

3.6 COROLLARY 2

Given an r-amalgamated derivation $G \Longrightarrow X$ via $p \oplus_r p'$ there is a "complete" analysis into derivation sequences $G \Longrightarrow Y \Longrightarrow H \Longrightarrow X$ via (r, p_o, p'_o) and $G \Longrightarrow Y \Longrightarrow H' \Longrightarrow X$ via (r, p'_o, p_o).



3.7 COROLLARY 3

Let $G \Longrightarrow H$ via p, $G \Longrightarrow H'$ via p' be r-amalgamable derivations and $G \Longrightarrow Y$ the corresponding derivation via r. Moreover, let $G \Longrightarrow X$ be the r-amalgamated derivation via $p \oplus_r p'$. Then X can be constructed from H and H' by gluing of H and H' along Y, provided that RR_1KB_1 and RR_2KB_2 are gluing diagrams (i.e. PO's).

4. The Proof of the Amalgamation Theorem

In this Section we give the proof of the AMALGAMATION THEOREM 3.4 together with some lemmata used in the proof. Since we can profit from the CONCURRENCY THEOREM presented in /ER 79a/ and /EHR 83/ in the proof of the AMALGAMATION THEOREM the CONCURRENCY THEOREM is stated first.

Let us review the problem of simulating a transformation sequence by a single transformation using only one "concurrent" production instead of a sequence of productions (see /ER 79a/ and /Ha 80/). Applying the concurrent production we can execute the relevant parts of the productions at the same time and hence concurrently, although

the productions themselves are not necessary applicable in parallel.

Given productions $p=(B_1 \leftarrow K \rightarrow B_2)$, $p'=(B_1' \leftarrow K' \rightarrow B_2')$ a (dependency) relation for (p,p') is a graph S which is related to the right side B_2 of p and the left side B_1' of p' by graph morphisms $S \rightarrow B_2$ and $S \rightarrow B_1'$. A derivation sequence $G \Rightarrow H \Rightarrow X$ via (p,p') which reflects the dependencies given by the relation S is called S-related.

On the other hand we can construct the S-concurrent production of p and p'

$p *_S p' = (B_1^* \leftarrow K^* \rightarrow B_2^*)$

where - roughly speaking - the left side B_1^* consists of B_1 and the non-S-related parts of B_1' and similar the right side B_2^* consists of B_2' and the non-S-related parts of B_2. A direct derivation $G \Rightarrow X$ via a S-concurrent production $p *_S p'$ is called S-concurrent. (For formal versions of the notions above we refer to /ER 79a/ and /EHR 83/.)

The connection between S-related sequences and S-concurrent derivations is established in the Concurrency Theorem.

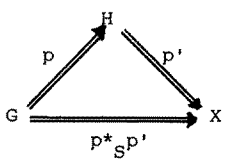
CONCURRENCY THEOREM

Let S be a relation for a pair of productions p,p' and $p *_S p'$ the corresponding S-concurrent production.

1. SYNTHESIS Given a S-related derivation $G \Rightarrow H \Rightarrow X$ via (p,p') then there is a canonical synthesis leading to a direct derivation $G \Rightarrow X$ via $p *_S p'$.

2. ANALYSIS Given a direct derivation $G \Rightarrow X$ via $p *_S p'$ then there is a canonical analysis into a S-related derivation $G \Rightarrow H \Rightarrow X$ via (p,p').

3. The operations SYNTHESIS and ANALYSIS are inverse to each other in the following sense: There is a bijective correspondence between S-related derivations $G \Rightarrow H \Rightarrow X$ via (p,p') and S-concurrent derivations $G \Rightarrow X$ via $p *_S p'$.



Using the CONCURRENCY THEOREM and some technical lemmata the proof of the AMALGAMATION THEOREM becomes very simple:

PROOF OF THE AMALGAMATION THEOREM

1. SYNTHESIS: Let $G \Rightarrow H$ via p, $G \Rightarrow H'$ via p' be r-amalgamable direct derivations.

1.1 Then there are derivation sequences

$G \Rightarrow Y \Rightarrow H$ via (r, p_o) (S-related)

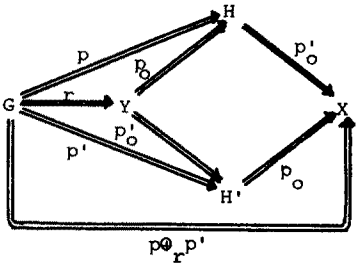
$G \Rightarrow Y \Rightarrow H'$ via (r, p_o') (S'-related)

because p,p' can be decomposed into $p = r *_S p_o$, $p' = r *_S p_o'$ (Lemma 1, Concurrency Theorem).

1.2 The direct derivations $Y \Rightarrow H$ via p_o, $Y \Rightarrow H'$ via p_o' are parallel independent, because $G \Rightarrow H$ via p, $G \Rightarrow H'$ via p' are r-amalgamable (Lemma 3). Hence there are a graph X and direct derivations $H \Rightarrow X$ via p_o', $H' \Rightarrow X$ via p_o such that the derivation

sequences $Y \Longrightarrow H \Longrightarrow X$ via (p_o, p'_o), $Y \Longrightarrow H' \Longrightarrow X$ via (p'_o, p_o) become sequentially independent (see CHURCH-ROSSER-PROPERTY I e.g. in /Eh 79/).

1.3 The derivation sequence $G \Longrightarrow H \Longrightarrow X$ via (p, p'_o) is S'-related, because $G \Longrightarrow Y \Longrightarrow H'$ via (r, p'_o) is S'-related (Lemma 4). Moreover, the S'-concurrent production $p *_{S'} p'_o$ is equal to the r-amalgamated production $p \oplus_r p'$ (Lemma 2). Hence there is a canonical synthesis leading to a direct derivation $G \Longrightarrow X$ via $p \oplus_r p'$.



2. ANALYSIS: Let $G \Longrightarrow X$ via $p \oplus_r p'$ be an r-amalgamated derivation.

2.1 There are derivation sequences

$G \Longrightarrow H \Longrightarrow X$ via (p, p'_o) (S'-related)

$G \Longrightarrow H' \Longrightarrow X$ via (p', p_o) (S-related)

because $p \oplus_r p'$ possesses decompositions $p *_{S'} p'_o$ and $p' *_S p_o$ (Lemma 2, Concurrency Theorem).

2.2 The direct derivations $G \Longrightarrow H$ via p, $G \Longrightarrow H'$ via p' are r-amalgamable, which can be seen as follows: There are derivation sequences

$G \Longrightarrow Y \Longrightarrow H$ via (r, p_o) (S-related)

$G \Longrightarrow Y \Longrightarrow H'$ via (r, p'_o) (S'-related)

because p and p' possess decompositions $p = r *_S p_o$ and $p' = r *_{S'} p'_o$ (Lemma 1, Concurrency Theorem). Now the S'-relatedness of $G \Longrightarrow H \Longrightarrow X$ implies the sequential independency of $Y \Longrightarrow H \Longrightarrow X$ (Lemma 5) and the parallel independency of $Y \Longrightarrow H$, $Y \Longrightarrow H'$ (CHURCH-ROSSER-PROPERTY II, /Eh 79/). By Lemma 3 $G \Longrightarrow H$, $G \Longrightarrow H'$ become r-amalgamable.

3. The bijective correspondence between r-amalgamable derivations and r-amalgamated derivations is an immediate consequence of the bijective correspondence between related derivation sequences and concurrent derivations.

Finally we state the lemmata used in the proof. For the proofs of these lemmata we refer to /BFH 84/ and /Fo 84/.

LEMMA 1 (SEPARATION OF PRODUCTIONS)

Let r be a relational production for p and p'. Then there are uniquely determined productions p_o and p'_o and relations S for (r, p_o) and S' for (r, p'_o) such that

$p = r *_S p_o$ and $p' = r *_{S'} p'_o$.

LEMMA 2 (DECOMPOSITION OF AMALGAMATED PRODUCTIONS)

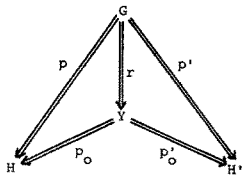
Let r be a relational production for p and p' and $\bar{p}$ the corresponding r-amalgamated production. Then there are uniquely determined decompositions

$\bar{p}=p*_S p'_o$ and $\bar{p}=p'*_S p_o$,

where p'_o and p_o are the remainders of p' and p and S' and S are uniquely determined relations for (p,p'_o) and (p',p_o) with $S' \rightarrow B_2 = S' \rightarrow R_2 \rightarrow B_2$ and $S \rightarrow B'_2 = S \rightarrow R_2 \rightarrow B'_2$.

LEMMA 3

Let $G \Rightarrow H$, $G \Rightarrow H'$ be direct derivations via p resp. p' and r be a relational production for p and p'. Let $G \Rightarrow Y \Rightarrow H$, $G \Rightarrow Y \Rightarrow H'$ be the corresponding derivation sequences via (r,p_o) resp. (r,p'_o). Then

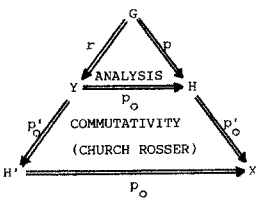


$G \Rightarrow H$ via p and $G \Rightarrow H'$ via p' are r-amalgamable iff

$Y \Rightarrow H$ via p_o and $Y \Rightarrow H'$ via p'_o are parallel independent.

LEMMA 4

Let $G \Rightarrow H$ be a direct derivation via p and $G \Rightarrow Y \Rightarrow H$ be the corresponding derivation sequence via (r,p_o). Let $Y \Rightarrow H \Rightarrow X$ be a sequentially independent derivation sequence via (p_o,p'_o) and $Y \Rightarrow H' \Rightarrow X$ the corresponding derivation sequence (p'_o,p_o). Then



$G \Rightarrow H \Rightarrow X$ via (p,p'_o) is S'-related if

$G \Rightarrow Y \Rightarrow H'$ via (r,p'_o) is S'-related.

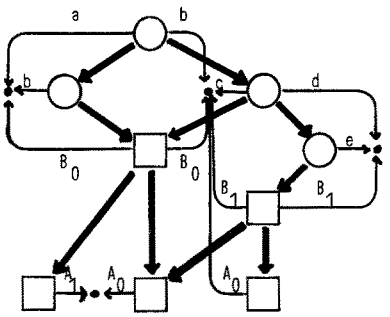
LEMMA 5

Let $G \Rightarrow H \Rightarrow X$ be a derivation sequence via (p,p'_o) and $G \Rightarrow Y \Rightarrow H$ via (r,p_o) the analysis of $G \Rightarrow H$ via p. Then S'-relatedness of $G \Rightarrow H \Rightarrow X$ via (p,p'_o) implies sequential independency of $Y \Rightarrow H \Rightarrow X$ via (p_o,p'_o).

5. Application to Synchronization in Distributed Systems

In this section we exemplify the application of the amalgamation mechanism to synchronization problems within Montanari's graph grammar formalism GDS (Grammars for Distributed Systems) /CM 83/, /DM 83/, /CDM 84/, a formalism for modelling the behaviour of nondeterministic dynamic process nets with distributed synchronization. Following Kung /Ku 80/, who classified the class of concurrent systems with respect to their module granularity, communication geometry and concurrency control (synchronization), GDS supports essential features of synchronization and distributedness. For details, concerning the link to other specification techniques, we refer to /DM 83/, where the relationship of GDS with Petri-Nets /Pe 80/, Milner's SCCS /Mi 82/, and Hoare's CSP /HBR 81/ is discussed. The graph grammar formalism GDS, introduced in /CM 83/, is based on labeled, partially ordered hypergraphs, called

distributed systems. A distributed system models both the spatial and temporal aspects of a real system through the relations of adjacency and causality. The productions of a grammar represent the possible stand-alone evaluations of system components. For modelling synchronized evaluation of adjacent system components the productions have to be synchronized. The synchronization of productions as well as the application of synchronized productions is described by two procedures A and B. The (terminal) distributed systems derived within a given grammar represent the alternative deterministic, concurrent computations of a single nondeterministic system which is completely modeled by the grammar.
In the following we will show that the algorithmic procedures A and B, given in /CM 83/, easily can be expressed in terms of amalgamation and application of graph productions. (Note that our considerations are based on graphs instead of hypergraphs.)



Representation Conventions:

Processes, events and ports are special colored nodes whereby boxes (□) denote processes, circles (○) events and bullets (•) ports.

Lowercase letters (a,b,c,d,e) correspond to terminal labels (actions) and uppercase letters (A_0,A_1,B_0,B_1) to nonterminal labels (process-types).

Fig. 5.1: distributed system G

A distributed system consists of three different kinds of nodes, processes, events and ports and three different kinds of arcs, terminal labeled arcs from events to ports, nonterminal labeled arcs from processes to ports and causal arcs (bold arcs) between subsystems, i.e. processes and events. A terminal label marks the action that happened, a nonterminal label marks the activated process type. The causal arcs induce a partial ordering on the set of subsystems. If two subsystems are related with respect to the partial ordering, they are called causally or sequentially dependent, otherwise they are concurrent. If subsystems are linked by a common port, they are called adjacent. Adjacency and concurrency may be interpreted as spatial and temporal overlapping. Events protocolize evaluation steps in the past while processes represent the possible nondeterministic future of a distributed system. Ports correspond to common storage, channels etc. Various requirements are formulated for distributed systems which ensure technical and logical consistency. The most important ones are, that no process may preceed an event with respect to the partial ordering and that an event cannot be concurrent with an adjacent subsystem.
A (GDS-)production $p=(B_1 \xleftarrow{b_1} K \xrightarrow{b_2} B_2)$ consists of distributed systems B_1,K,B_2 and injective graph morphisms b_1,b_2, such that certain technical conditions are satis-

fied. In general p describes the evaluation of processes with corresponding process types and ports together with the local modification of the partial ordering.

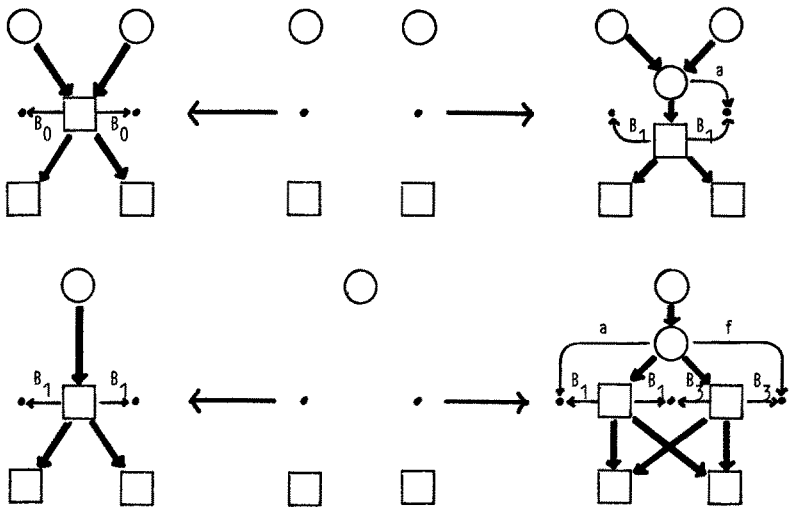


Fig. 5.2: productions p,p'

A production $p=(B_1 \leftarrow K \rightarrow B_2)$ can be applied to a distributed system G if there is an embedding morphism $g:B_1 \rightarrow G$, satisfying the Gluing Condition as well as an Injectivity, a Locallity and a Concurrency Constraint. The Concurrency Constraint requires that no process or event in $G-gB_1$, which is concurrent to all processes in B_1-b_1K is connected to an image g(s) of a synchronization port s of B_1 (synchronization ports s of B_1 are those ports, whose images $b_2b_1^{-1}(s)$ are connected to an event in B_2). The result H of applying a production p to a distributed system G w.r.t. an embedding morphism g is defined by the direct derivation $G \xRightarrow{p,g} H$.

In our example neither p nor p' can be applied to G: each embedding morphism does not satisfy the Concurrency Constraint. This calls for synchronization of p,p'; the underlying two processes have to be evaluated synchronously.

Synchronization of two GDS-productions $p=(B_1 \leftarrow K_1 \rightarrow B_2)$, $p'=(B_1' \leftarrow K' \rightarrow B_2')$ w.r.t. embedding morphisms $g:B_1 \rightarrow G$, $g':B_1' \rightarrow G$ and a distributed system G is done as follows: First we try to construct a relational production r for p,p' w.r.t. g,g'. If there is a relational production, then p,p' are synchronizable w.r.t. g,g', otherwise other productions or embedding morphisms have to be chosen. In the case of synchronizability, we can construct the amalgamation $p \otimes_r p'$ of p,p' w.r.t. r, which we call the synchronized production or synchronization of p,p' w.r.t. g,g'.

REMARK: An interpretation of synchronizability is, that processes which overlap in time and space must generate same actions at common ports. The procedure mentioned above, can be iterated. Because of that, synchronization of more than two productions is possible. This is necessary, if the synchronized production is not

applicable, i.e. there are no embedding morphisms which satisfy the applicability constraints.

In our example p,p' are not applicable but synchronizable w.r.t. the distributed system G. The constructed relational production r for p,p' is given in Fig. 5.3. The amalgamated production $p\Theta_r p'$ of p,p' with respect to r is presented in Fig. 5.4.

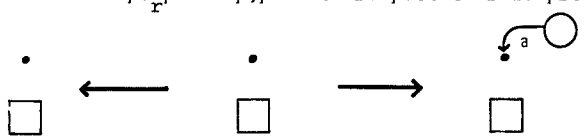


Fig. 5.3: relational production r for p,p'

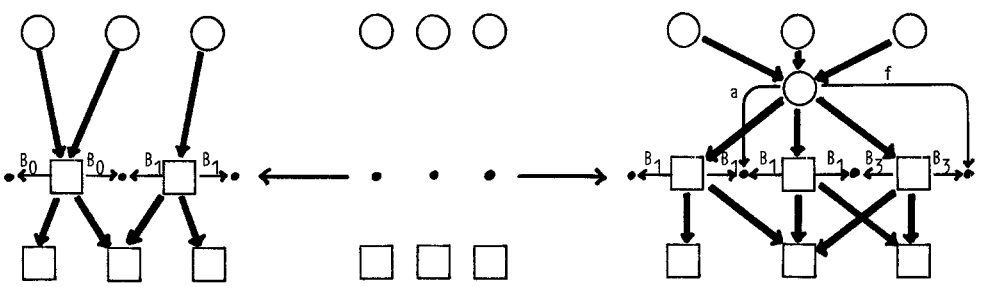


Fig. 5.4: amalgamated production $p\Theta_r p'$ of p,p' w.r.t. r

The amalgamated production $p\Theta_r p'$ of p,p' with respect to the relational production r is applicable, i.e. the uniquely determined embedding morphism into G satisfies all applicability constraints. Application of the amalgamated production $p\Theta_r p'$ to G leads to the distributed system H given in Fig. 5.5.

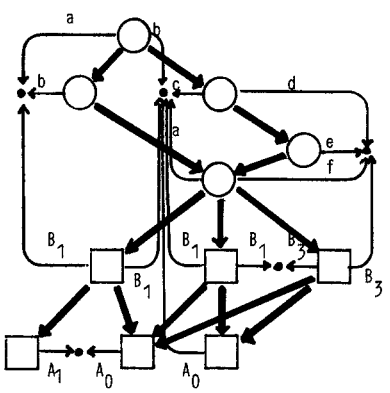


Fig. 5.5: derived distributed system H

The amalgamation concept, introduced in Section 3, can be used to describe the synchronization of productions in the sense of /CM 83/. This approach avoids the algorithmic procedures for synchronization and application of productions. As a consequence we get easy proofs with respect to iterated synchronization and consistency:

(1) The synchronization of two GDS-productions w.r.t. a relational GDS-production leads to a GDS-production.

(2) The application of a GDS-production to a distributed system leads to a distributed system.

With regard to a formal treatment we refer to /BFH 84/. We can state that the

parallelism and concurrency concept together with the amalgamation concept give us excellent possibilities for modelling the behaviour of distributed systems.

References

/AM 75/ M.A. Arbib, E.G. Manes: Arrows, Structures, and Functors, Academic Press, New York

/BFH 84/ P. Boehm, H. Fonio, A. Habel: On Amalgamation of Graph Manipulations; in preparation

/CER 79/ V. Claus, H. Ehrig, G. Rozenberg, (eds.): Graph Grammars and Their Application to Computer Science and Biology, LNCS 73 (1979)

/CDM 84/ A. Corradini, P. Degano, U. Montanari: Specifying Highly Concurrent Data Structure Manipulation; Comp. Sci. Dept., Univ. of Pisa, Pisa, April 1984

/CM 83/ I. Castellani, U. Montanari: Graph Grammars for Distributed Systems, LNCS 153, 20-38 (1983)

/DM 83/ P. Degano, U. Montanari: A Model of Distributed Systems Based on Graph Rewriting, Note Cnet 111, Comp. Sci. Dept., Univ. of Pisa, Pisa 1983, submitted for publication

/Eh 79/ H. Ehrig: Introduction to the Algebraic Theory of Graph Grammars, LNCS 73 (1979), 1-69

/Eh 83/ --: Aspects of Concurrency in Graph Grammars, LNCS 153 (1983), 58-81

/EHR 83/ H. Ehrig, A. Habel, B.K. Rosen: Concurrent Transformations of Structures: From Graphs to Relational Data Structures; TU Berlin, FB 20, Technical Report No. 83-01, January 1983

/EK 80/ H. Ehrig, H.-J. Kreowski: Applications of Graph Grammar Theory to Consistency, Synchronization and Scheduling in Data Base Systems, Inform. Syst., Vol. 5, pp. 225-238, Pergamon Press Ltd., 1980

/ENR 83/ H. Ehrig, M. Nagl, G. Rozenberg (eds.): Graph Grammars and Their Application to Computer Science, LNCS 153 (1983)

/EPS 73/ H. Ehrig, M. Pfender, H.J. Schneider: Graph-Grammars: An Algebraic Approach, Proc. of the IEEE Conf. on Automata and Switching Theory, Iowa City 1973, pp. 167-180

/ER 79a/ H. Ehrig, B.K. Rosen: Parallelism and Concurrency of Graph Manipulations, Theoret. Comp. Sci. 11 (1980), pp. 247-275

/ER 79b/ --: Decomposition of Graph Grammar Productions and Derivations, LNCS 73 (1979), pp. 192-205

/Fo 84/ H.-R. Fonio: Amalgamation of Graph Transformations with Application to Synchronization in Distributed Systems, to appear as Techn. Report, TU Berlin, FB 20

/Ha 80/ A. Habel: Concurrency in Graph-Grammatiken, TU Berlin, FB 20, Technical Report No. 80-11, March 1980

/HBR 81/ C.A.R. Hoare, S.D. Brookes, A.W. Roscoe: A Theory of Communicating Sequential Processes, Techn. Monograph PRG-16, Progr. Research Group, Oxford Univ., 1981

/JKRE 82/ D. Janssens, H.-J. Kreowski, G. Rozenberg, H. Ehrig: Concurrency of Node-label Controlled Graph Transformations, Techn. Report No. 82-38, Univ. of Antwerp, U.I.A. (1982)

/Ku 80/ H.T. Kung: The Structures of Parallel Algorithms, Advances in Computers, Vol. 19, 65-108, Academic Press Inc. 1980

REFERENCES (cont'd)

/MW 82/ B. Mahr, A. Wilharm: Graph Grammars as a Tool for Description in Computer Processed Control: A Case Stude, Proc. 8th Conf. on Graph-theoretic Concepts in Comp. Sci. (WG'82), 165-176 (1982)

/Mi 83/ R. Milner: Calculi for Synchrony and Asynchrony, Theoret. Comp. Sci. 25 (1983), pp. 267-310

/Na 79/ M. Nagl: A Tutorial and Bibliographical Survey on Graph Grammars, LNCS 73 (1979), 70-126

/Pe 80/ C.A. Petri: Concurrency, Proc. Net Theory and Applications, LNCS 84 (1980), 251-260

/Ro 75/ B.K. Rosen: Deriving Graphs from Graphs by Applying a Production, Acta Informatica, 4 (1975), pp. 337-357