THE FORMAL SPECIFICATION COLUMN

BY

HARTMUT EHRIG

Technical University of Berlin, Department of Computer Science
Franklinstraße 28/29, D-10587 Berlin, Germany
ehrig@cs.tu-berlin.de

ATTRIBUTED GRAPHS AND TYPING: RELATIONSHIP BETWEEN DIFFERENT REPRESENTATIONS

Hartmut Ehrig
Institut für Softwaretechnik und Theoretische Informatik
Technische Universität Berlin, Germany
ehrig@cs.tu-berlin.de

Abstract

Attributed graphs and typing play an important role in theory and applications of graph grammars, graph transformation systems, visual languages and metamodelling. Attributed graphs can be represented basically as pairs of graphs and algebras on one hand and as algebras of suitable algebraic signatures on the other hand. In this note the different notions are compared on the syntactical and on the semantical level. Two different kinds of algebraic signatures for attributed algebras are discussed leading to different results on both levels. In the case of attributed graph signatures the corresponding category of algebras is isomorphic to the category of typed attributed graphs, while we have only a non-surjective functor in the more general case of attributed algebras for graph structure signatures. An overview of all results is given in the last section of this paper.

1 Introduction

In this note we discuss different versions of attributed graphs and typing in the area of graph transformation systems and discuss their relationship. Perhaps the most intuitive notion is to define attributed graphs as pairs $AG = (G, A)$ of a graph G and an algebra A of a given data type signature $DSIG$, where specific domains of A, the attribute values, are considered as part of the vertices of the graph G (see [4]). Together with suitable attributed graph morphisms, consisting of pullback-compatible pairs of graph and algebra homomorphisms, we obtain the category **AGraphs** of attributed graphs. Given in addition an attributed type graph $ATG = (TG, Z)$, where Z is the final $DSIG-algebra$, we obtain as comma category the category $\mathbf{AGraphs}_{ATG}$ of ATG-typed attributed graphs of graph transformation (see [2], [6]).

From a theoretical point of view, especially in view of high-level replacement systems, [1],we would like to represent $\mathbf{AGraphs}_{ATG}$ as a category of algebras for a given algebraic signature. The most obvious choice for this purpose seem to be attributed graph structure signatures $ASSIG = (SIG, DSIG)$: They consist of a graph structure signature SIG, an algebraic signature with unary operation symbols only, and a data type signature $DSIG$, where the sorts of SIG and $DSIG$ overlap in attribute value sorts. In fact, there is an injective functor from the category of $ASSIG-algebras$ to the category $\mathbf{AGraphs}_{ATG}$, which - however - is not an isomorphism of categories in general. This functor is based on a representation of graph structure signatures as graphs, where all sorts of the signature are considered as vertices of the graph and the unary operations as edges. A more compact representation as graph can be obtained if the sorts of the graph structure signature are divided into vertex and edge sorts. In this case the vertex sorts are represented as vertices and the edge sorts as edges of the graph, where source and target of the edges are defined by the signature of the operation symbols. This special kind of graph structure signature is called graph signature. Taking graph signatures $GSIG$ instead of graph structure signatures SIG and attributed graph signatures $ASIG = (GSIG, DSIG)$ instead of attributed graph structure signatures has three advantages:

1. The representation of graph signatures as graphs is much more intuitive and compact than that of graph structure signatures, where some of the sorts of the signature have to be interpreted as attribute carrier nodes in the graph. In a similar way the representation of attributed graph signatures as attributed graphs is much more intuitive and compact than that of the more general attributed graph structure signatures.

2. Based on the more compact representation discussed above, there is an isomorphism of the category **ASIG-Alg** of $ASIG-algebras$ with the category

AGraphs$_{ATG}$, where ATG is the compact representation of the attributed graph signature $ASIG$.

3. The category **ASIG-Alg**$_{TA}$ of typed $ASIG-algebras$, where the type algebra TA is an arbitrary $ASIG-algebra$, is isomorphic to the category **AGraphs**$_{TAG}$ of typed attributed graphs, where the attributed type graph TAG is obtained from TA according to the isomorphism discussed above. This means that TAG is typed over ATG, such that the attributed graphs in **AGraphs**$_{TAG}$ are double typed.

In Sections 2 and 3 of this note we present attributed graphs and attributed graph signatures as discussed above. In Section 4 we show the bijective correspondence between attributed graph signatures and attributed type graphs $ATG = (TG, Z)$ with final $DSIG-algebra\ Z$. In Section 5 we show the isomorphism **ASIG-Alg** $\xrightarrow{\sim}$ **AGraphs**$_{ATG}$ and the isomorphism **ASIG-Alg**$_{TA}$ $\xrightarrow{\sim}$ **AGraphs**$_{TAG}$ together with some examples in Section 6. In Section 7 we briefly sketch the corresponding results for general graph structure signatures instead of the more specific graph signatures. It is important to note that we obtain an isomorphic representation of typed attributed graphs in Sections 5 and 6, but only a non-surjective functorial representation in Section 7. In Section 8 we give an overview of all the results. For all of the results we have proof sketches, but only the main constructions are presented together with examples in this note.

Acknowledgements

We are grateful to the GRA-GRA group at TU Berlin and to Reiko Heckel for fruitful discussions concerning this note and to Mara Oswald and Claudia Ermel for careful typing and figure drawing.

2 Attributed Graphs and Typing

In this section we follow the approach in [4] to introduce attributed graphs $AG = (G, A)$ as pairs of a graph G and an algebra A with data type signature $DSIG$ in the sense of algebraic signatures and data types (see [3]). Some of the domains of A serve as attribute values. For this reason they are considered as part of the vertices of the graph G such that the attribute assignment can be given by edges in G. Attributed graph morphisms are pairs of graph morphisms and algebra homomorphisms with a suitable compatibility, called pullback-compatibility, which is stronger than the compatibility required in [4]. For typing we consider specific attributed graphs as attributed type graphs $ATG = (TG, Z)$, where Z is required to be the final $DSIG-algebra$. A typed attributed graph over ATG consists of an

attributed graph AG and a type morphism $t : AG \to ATG$. Note that the data part $t_D : A \to Z$ is uniquely determinated by the fact that Z is a final algebra.

Altogether we obtain two important categories, the category **AGraphs** of attributed graphs and the category $\mathbf{AGraphs}_{ATG}$ of ATG-typed attributed graphs.

Definition 1

Given a data type signature $DSIG = (S_D, OP_D)$ with attribute value sorts $S' \subseteq S_D$, an attributed graph $AG = (G, A)$ consists of a

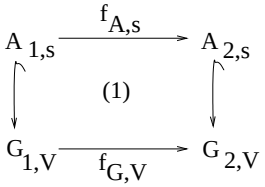
graph $G = (G_V, G_E, source_G, target_G)$ and an

algebra A which is a $DSIG$-algebra with $\dot{\bigcup}_{s \in S'} A_s \subseteq G_V$ (attribute values are vertices)

An attributed graph morphism $f : AG_1 \longrightarrow AG_2$ from $AG_1 = (G_1, A_1)$ to $AG_2 = (G_2, A_2)$ is a pair $f = (f_G, f_A)$,

$f_G : G_1 \longrightarrow G_2$ graph morphism

$f_A : A_1 \longrightarrow A_2$ algebra homomorphism with PB-compatibility, i.e. (1) is pullback in the category **Sets** for all $s \in S'$.



This leads to the category **AGraphs** of attributed graphs where attribute graphs are the objects and the morphisms are given by attributed graph morphisms.

An attributed type graph $ATG = (TG, Z)$ is an attributed graph, where Z is a final $DSIG$-algebra with $Z_s = \{s\}$ for all $s \in S_D$.

This leads to the category $\mathbf{AGraphs}_{ATG}$ of ATG-typed attributed graphs, defined as comma category over **AGraphs**.

This means that the objects of $\mathbf{AGraphs}_{ATG}$ are given by pairs $(AG, t : AG \to ATG)$ of attributed graphs AG with type morphism $t : AG \to ATG$, and the morphisms in $\mathbf{AGraphs}_{ATG}$ are given by $f : (AG_1, t_1) \to (AG_2, t_2)$, where $f : AG_1 \to AG_2$ is an attributed graph morphism with $t_1 = t_2 \circ f$.

Examples will be given in sections 4 to 7.

Remarks

1. For attributed graphs $AG = (G, A)$ only attribute values $A_s (s \in S')$ are required to be vertices in G. Edges in G with sources in data domains A_s for $s \in S'$ are allowed for graphs without typing, but can be prohibited for ATG-typed attributed graphs if ATG has no edges with source in S'.

2. For attributed graph morphisms the PB-compatibility is stronger than inclusions $f_{A,s} \subseteq f_{G_V}$ for $s \in S'$. It makes sure that we have in addition $f_{G_V}^{-1}(A_{2,s}) = A_{1,s}$ which is essential for most of the results.

3. For attributed type graphs $ATG = (TG, Z)$ the algebra Z is final, but we will also consider arbitrary attributed graphs as type graphs, called $TAG = (TG, A)$ in this case, s.t. $\mathbf{AGraphs}_{TAG}$ = Category of TAG-typed attributed graphs.

3 Attributed Graph Signatures

In this section we follow the idea of [5] to consider attributed graphs as algebras of a specific algebraic signature. In [5] graph structure signatures have been considered for the graph part. This idea will be discussed in Section 7. In this section we consider a specific subclass of graph structure signatures, called graph signatures, which allow a very compact representation as graphs (see section 4). Graph signatures $GSIG$ can be extended by data type signatures $DSIG$ leading to attributed graph signatures $ASIG = (GSIG, DSIG)$. This approach allows to consider attributed graphs as $ASIG-algebras$, where the GSIG-part corresponds to the graph and the DSIG-part to the attribute algebra. An overlap of sorts in $GSIG$ and $DSIG$ leads to a connection between both parts which represents the attribute functions. The correspondence between attributed graph signatures and attributed type graphs will be discussed in Section 4.

Definition 2

A graph signature $GSIG = (S_G, OP_G)$ is an algebraic signature with

$S_G = S_V \dot{\cup} S_E$ (distinction of vertex and edge sorts),
$OP_G = \dot{\cup}_{e \in S_E} OP_e \quad OP_e = \{src_e, tar_e\}, \quad e \in S_E$ with
$src_e : e \to v_1(e)$ and $tar_e : e \to v_2(e)$ for $v_1(e), v_2(e) \in S_V$.

An attributed graph signature $ASIG = (GSIG, DSIG)$ consists of

$GSIG = (S_G, OP_G)$ graph signature,
$DSIG = (S_D, OP_D)$ data type signature with value sorts $S' \subseteq S_D$ and $S' \subseteq S_V$ (value sorts of $DSIG$ are vertex sorts of $GSIG$),
$S_A = S_G \cup S_D$ (sorts of $ASIG$ defined by union),
$OP_A = OP_G \cup OP_D$ (opns of $ASIG$ defined by union where $OP_G \cap OP_D = \emptyset$).

The class of all attributed graph signatures $ASIG = (GSIG, DSIG)$ with fixed data type signature $DSIG$ is denoted by **AGraphSig**.

Remarks

1. A graph signature is a special kind of graph structure signature SIG, where all operation symbols are unary. In a graph signature vertex sorts correspond to vertices and edge sorts to edges of a graph, while for graph structure signatures all sorts correspond to vertices of a graph.

2. Attributed graph signatures are a special case of LKW-signatures considered by Löwe, Korff and Wagner [5].

3. In order to avoid edges in the graph part with source in a value sort we can require $v_1(e) \in S_V - S'$ for $src_e : e \to v_1(e)$.

Example 1

An example of an attributed graph signature is given by $ASIG = (GSIG, DSIG)$ below, where the sort nat is shared between $GSIG$ and $DSIG$.

$$
\begin{array}{lll}
GSIG = & \underline{\text{vertex sorts}}: & v_1, v_2, nat \\
& \underline{\text{edge sorts}}: & e_1, e_2, e_0 \\
& \underline{\text{opns}}: & src_{e_1} : e_1 \to v_1 \\
& & tar_{e1} : e_1 \to v_2 \\
& & src_{e2} : e_2 \to v_2 \\
& & tar_{e2} : e_2 \to v_1 \\
& & src_{e0} : e_0 \to v_2 \\
& & tar_{e0} : e_0 \to nat \\
DSIG = \mathbf{nat} & = & \\
& \underline{\text{value sorts:}} & nat \\
& \underline{\text{opns:}} & ZERO :\to nat \\
& & SUC : nat \to nat \\
& & ADD : nat\ nat \to nat
\end{array}
$$

Two different graph representations of $GSIG$ are considered as type graphs TG in example 2 and as TG_1 in example 4.

4 Bijective Correspondence between Attributed Graph Signatures & Attributed Type Graphs

In this section we show the close correspondence between attributed graph signatures and attributed type graphs as introduced in the previous sections. In fact we have a bijective correspondence.

Lemma 1

Let **AGraphSig** be the class of attributed graph signatures,and **ATGGraphs** the class of attributed type graphs, then we have bijective correspondence
AGraphSig $\xrightarrow{\sim}$ **ATGGraphs**.

Construction

In the following we give two transformations $AGra$ and $AGra^{-1}$ between the classes **AGraphSig** and **ATGGraphs** which are inverse to each other, where the data type signature $DSIG$ is given explicitly in $ASIG = (GSIG, DSIG)$ and implicitly in $ATG = (TG, Z)$ because Z is the final $DSIG - algebra$.

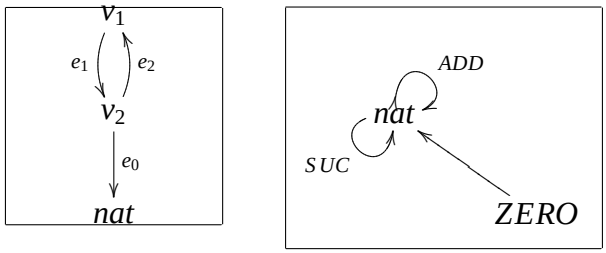
$AGra$: **AGraphSig**	$\xrightarrow{\sim}$	**ATGGraphs**
$ASIG = (GSIG, DSIG)$	$\mapsto$	$AGra(ASIG) = (TG, Z)$ with
with $S' \subseteq S_V$		$TG_V = S_V, TG_E = S_E$
$S_G = S_V \dot{\cup} S_E$		$source_{TG}(e) = v_1(e)$ for $src_e : e \to v_1(e)$
		$target_{TG}(e) = v_2(e)$ for $tar_e : e \to v_2(e)$
		Z final DSIG-algebra with
		$S' \subseteq S_V = TG_V$
$AGra^{-1}$: **ATGGraphs**	$\xrightarrow{\sim}$	**AGraphSig**
$ATG = (TG, Z)$	$\mapsto$	$ASIG = (GSIG, DSIG)$*with*
with $S' \subseteq S_V$		$S_G = S_V \dot{\cup} S_E, S_V = TG_V, S_E = TG_E$
		$src_e : e \to v_1(e)$ for $source_{TG}(e) = v_1(e)$
		$tar_e : e \to v_2(e)$ for $target_{TG}(e) = v_2(e)$

Remark

This bijection is not an isomorphism of categories, unless the signature morphisms for **AGraphSig** are restricted to preserve vertex sorts S_V and and edge sorts S_E separately.

Example 2

The attributed type graph $AGra(ASIG)$ with $ASIG$ from previous example 1 is given by $AGra(ASIG) = (TG, Z)$ with



5 Isomorphism between Categories of Attributed Graph Algebras

Based on the bijective correspondence on the syntactical level in the previous section we show now a bijective correspondence on the semantical level. This is a bijection between the class of $ASIG-algebras$ and the class of attributed graphs typed over ATG, where $ASIG$ and ATG correspond to each other according to Lemma 1.

Theorem 1

Let **ASIG-Alg** be the category of $ASIG-algebras$ for an attributed graph signature $ASIG$, and $\mathbf{AGraphs}_{ATG}$ category of ATG-typed attributed graphs with $ATG = AGra(ASIG)$ (see Lemma 1).
Then we have an isomorphism T_{AGra} : **ASIG-Alg** $\xrightarrow{\sim}$ $\mathbf{AGraphs}_{ATG}$.

Construction

In the following we present the construction for objects of the corresponding categories. But it can also be shown for morphisms leading to an isomorphism between the two categories.

T_{AGra}: **ASIG-Alg**	$\tilde{\longrightarrow}$	**AGraphs**$_{ATG}$
A	$\mapsto$	$T_{AGra}(A) = (AG, t : AG \to ATG)$
with $ASIG = (GSIG, DSIG)$		with $AG = (G, D)$
$GSIG = (S_G, OP_G)$		$G_V = \dot{\cup}_{s \in S_V} A_s$
$S_G = S_V \dot{\cup} S_E$		$t_V(a) = s$ for $a \in A_S, s \in S_V$
$OP_G = \dot{\cup} OP_e$		with $t_V^{-1}\{s\} = A_s$
$OP_e = \{src_e, tar_e\}$		$G_E = \dot{\cup}_{e \in S_E} A_e$
		$t_E(a) = e$ for $a \in A_E, e \in S_E$
		with $t_e^{-1}\{e\} = A_e$
		$source_G(a) = src_e^A(a)$ for $a \in A_e$
		$target_G(a) = tar_e^A(a)$
		$D = V_{DSIG}(A)$
		$t_D(A) = s$ for $a \in A_s, s \in S_D$

T_{AGra}^{-1} : **AGraphs**$_{ATG}$	$\tilde{\longrightarrow}$	**ASIG-Alg**
$(AG, t : AG \to ATG)$	$\mapsto$	A with
$AG = (G, D)$		$A_s = t_V^{-1}\{s\} \subseteq G_V$ for $s \in S_V$
$ATG = (TG, Z)$		$A_e = t_E^{-1}\{e\} \subseteq G_E$ for $e \in S_E$
$t_v : G_v \to TG_v = S_v$		$src_e{}^A(a) = source_G(a)$ for $e \in S_E$
with $t_V^{-1}\{s\} = D_s$ for $s \in S_D$		$A_s = t_V^{-1}\{s\} = D_s$ for $s \in S_D$

An example for the transformation T_{AGra} : **ASIG-Alg** $\to$ **AGraphs**$_{ATG}$ is given in Example 3 below, where A is an $ASIG-algebra$ and $T_{AGra}(A) = (AG, t : AG \to ATG)$ an ATG-typed attributed graph with $t = Tt \circ t'_A$.

6 Isomorphism between Categories of Typed Attributed Graph Algebras & Typed Attributed Graphs

In this section we extend attributed graph algebras considered as $ASIG-algebras$ in the previous section by typing. This means that we extend category **ASIG-Alg** to the comma category **ASIG-Alg**$_{TA}$ with $ASIG-algebra\ TA$ as type algebra. The objects in **ASIG-Alg**$_{TA}$ are pairs $(A, t_A : A \to TA)$ with $ASIG-algebra\ A$ and type morphism $t_A : A \to TA$, called typed attributed graph algebras. Based on theorem 1 in the previous section we can show now that typed attributed graph algebras are in bijective correspondence with doubly typed attributed graphs $(AG, t'_A : AG \to TAG)$. In fact, the type graph TAG corresponding to the type algebra TA by theorem 1 is itself typed over the attributed type graph ATG corresponding to $ASIG$ according to lemma 1.

Theorem 2

Let **ASIG-Alg**$_{TA}$ be the category of typed attributed graph algebras for the attributed graph signature $ASIG$ and $ASIG-algebra\ TA$ as type algebra, and let **AGraphs**$_{TAG}$ be the category of TAG-typed attributed graphs with

$TAG = T_{AGra}(TA)$ (see theorem 1).

Then we have an isomorphism TT_{AGra} : **ASIG-Alg**$_{TA} \xrightarrow{\sim}$ **AGraphs**$_{TAG}$.

Remarks

1. TAG is typed over ATG by $Tt : TAG \to ATG$ where $ATG = AGra(ASIG)$. Hence attributed graphs in **AGraphs**$_{TAG}$ are double typed, where typing over TAG is more restrictive then typing over ATG. Typing over TAG allows restrictions concerning specific attribute values and not only concerning attribute value sorts.

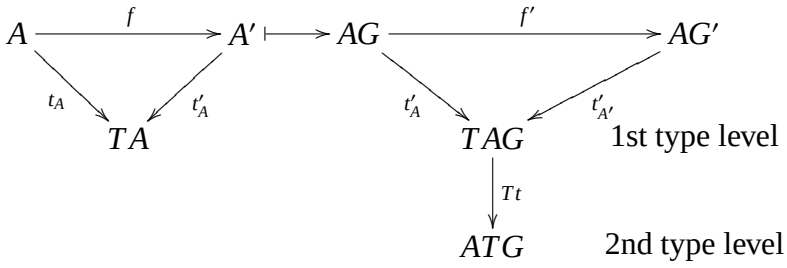
2. More precisely we have $T_{AGra}(TA) = (TAG, Tt : TAG \to ATG)$.

Construction

Similar to the previous section we present the constructions only for the objects of the corresponding categories using the consturctions T_{AGra} and T_{AGra}^{-1} from theorem 1. The extended constuctions are denoted by TT_{AGra} and TT_{AGra}^{-1}.

TT_{AGra} : **ASIG-Alg**$_{TA} \xrightarrow{\sim}$ **AGraphs**$_{TAG}$

$(A, t_A : A \to TA) \longmapsto (AG, t'_A : AG \to TAG)$ with



TT_{AGra}^{-1} : **AGraphs**$_{TAG} \xrightarrow{\sim}$ **ASIG-Alg**$_{TA}$

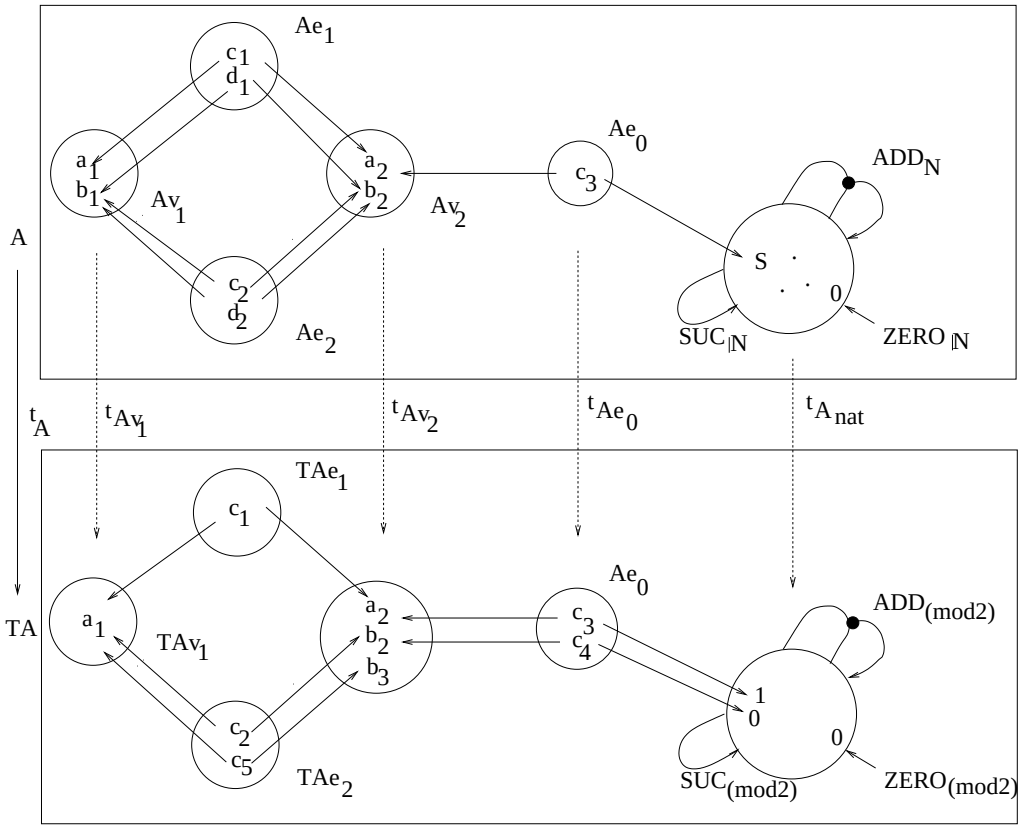
$(AG, t'_A : AG \to TAG) \longmapsto (A, t_A : A \to TA)$ with

$$A = T_{AGra}^{-1}(AG, AG \xrightarrow{t'_A} TAG \xrightarrow{Tt} ATG),$$

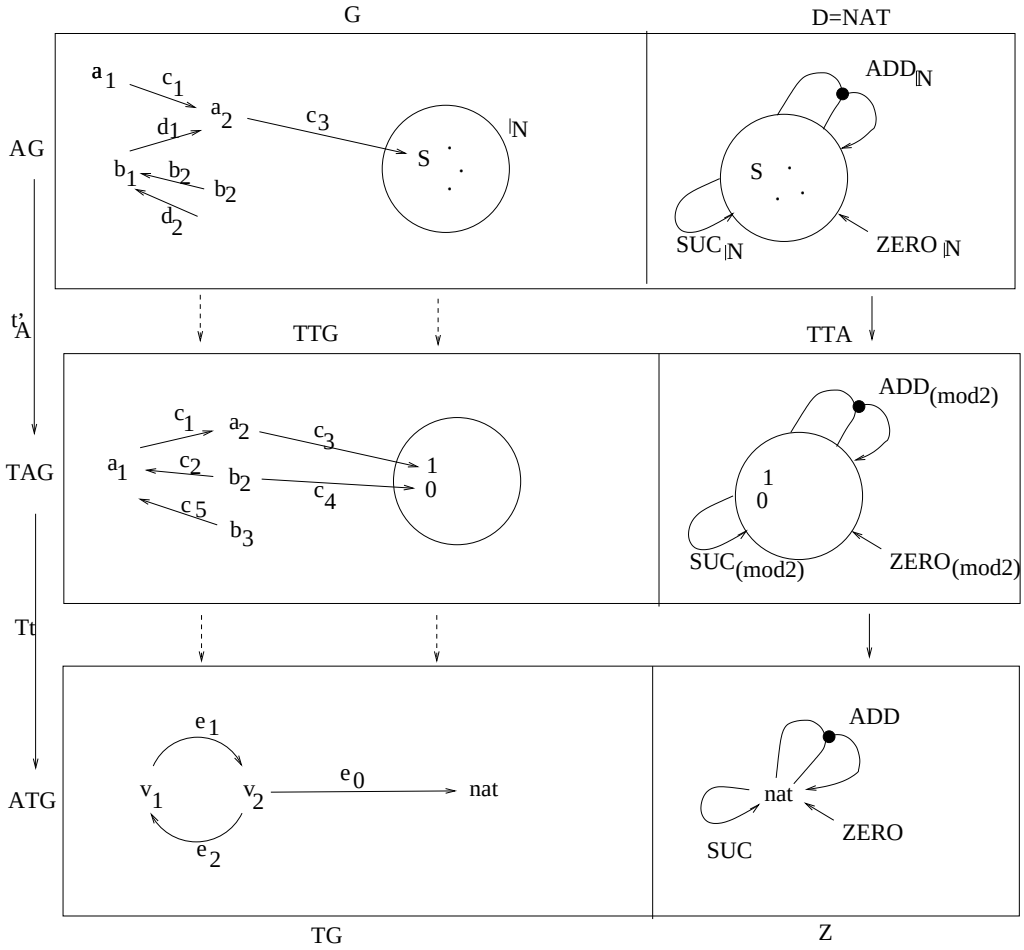
$$t_A = T_{AGra}^{-1}(t'_A) : A \to TA$$

Example 3

In the following we present an example for the transformation TT_{AGra} applied to an $ASIG-algebra\ A$ typed over an $ASIG-algebra\ TA$, which is given by $(A, t_A : A \to TA)$. The result $TT_{AGra}(A, t_A)$ is an attributed algebra typed over $TAG = T_{AGra}(TA)$, which is given by $(AG, t'_A : AG \to TAG)$ with TAG typed over ATG by $Tt : TAG \to ATG$. Let $ASIG-algebras\ A, TA$ and typing $t_A : A \to TA$ be given by the following diagram



The TAG-typed graph $TT_{AGra} = (AG, t'_A : AG \to TAG)$ with $AG = (G, D)$, $TAG = (TTG, TTA)$ and $ATG = (TG, Z)$ is given by the following diagram



7 Transformations Based on Attributed Graph Structure Signatures

In section 3 we have introduced graph signatures as a special case of graph structure signature, because they allow a very compact representation as graphs. Actually vertex sorts have been interpreted as vertices and edge sorts as edges of the corresponding graphs. In the more general case of graph structure signatures, which are studied in this section, all sorts of the signature are interpreted as vertices and all unary operation symbols as edges. This correspondence is stated in lemma 2 and leads to a less compact representation as graph as shown in example 4. In fact, we introduce in definition 3 graph structure signatures and attributed graph structure signatures, and lemma 2 and example 4 are dealing with the attributed case. In the attributed case graph structure signatures allow attribution of vertices and edges, while graph signatures allow only attribution of vertices, but

not of edges. This is certainly an advantage. The disadvantage compared with attributed graph signatures is that the corresponding transformations from algebras to attributed graphs in analogy to theorem 1 and 2 lead to functors in theorem 3 and 4, which are no longer bijective. This means that they do not define isomorphisms of the corresponding categories, but only non surjective functors (see counterexample 6). In contrast to the previous sections we only state the results and show the changes in examples 4 and 5 compared with examples 2 and 3 based on the same signature in example 1.

A graph structure signature $SIG = (S, OP)$ is an algebraic signature, where all operation symbols in OP are unary. An attributed graph structure signature $ASSIG = (SIG, DSIG)$ consists of a graph structure signature SIG and a data type signature $DSIG$ with attributed value sorts $S' \subseteq S$. Similar to lemma 1 in section 4 we have now

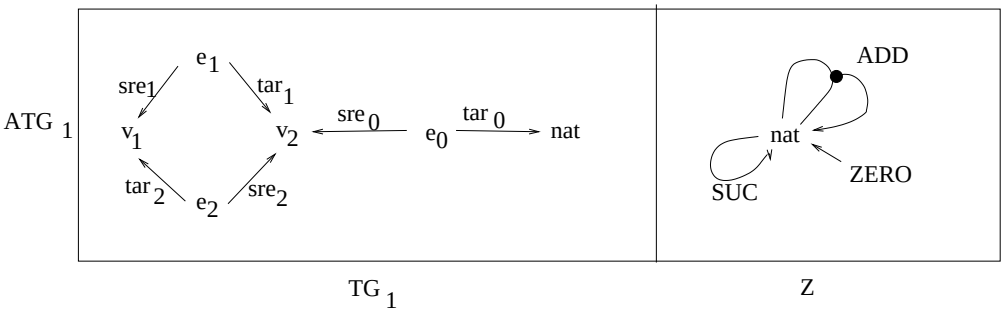
Lemma 2

Let **AGraphStructSig** be the category of attributed graph structure signatures and **ATGGraphs** the category of attributed type graphs (with identity in the algebra part). Then we have isomorphism $ASGra$: **AGraphStructSig** $\xrightarrow{\sim}$ **ATGGraphs**.

According to this isomorphism we obtain from the example $ASIG$ in section 3, considered now as attributed graph structure signature, the following attributed type graph (TG, Z), which is certainly more complex than the type graph (TG, Z) from example 2.

Example 4

$ASGra(ASIG)$ with $ASIG$ from Example 1 is given by
$ASGra(ASIG) = (TG_1, Z) = ATG_1$.



Similar to Theorem 1 we obtain now a functor T_{ASGra} from category **ASSIG-Alg** of ASSIG-algebras (for a given attributed graph structure signature **ASSIG**) to the category $AGraphs_{ATG}$, where **ATG** is given now by $ATG = ASGra(ASSIG)$ according to lemma 2.

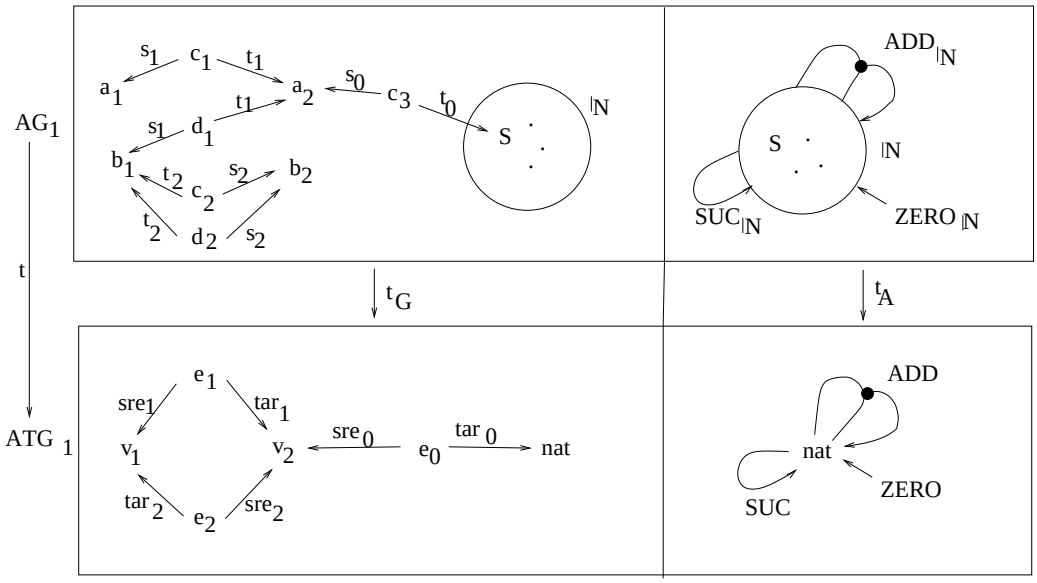
Theorem 3

$\mathbf{T}_{ASGra}$: **ASSIG-Alg** $\rightarrow$ $\mathbf{AGraphs}_{ATG}$ is functor based on $ASGra$ in lemma 2 with $ATG = ASGra(ASSIG)$.

In contrast to Theorem 1 this functor is in general not surjective and hence no isomorphism as shown in Example 5.

Example 5

Given the ASSIG-algebra A as shown in Example 3 the ATG_1-typed graph $T_{ASGra}(A) = (AG_1, t : AG_1 \rightarrow ATG_1)$ with $t = (t_G, t_A)$ and $t_G(s_i) = src_i, t_G(t_i) = tar_i$ for $i = 0, 1, 2$ is given by



Counterexample 6

We modify Example 5 in order to show that the functor T_{ASGra} is not surjective in general. Consider the ATG_1-typed graph (AG'_1, t'), which is AG_1 together with an additional edge form c_2 to a_2 mapped to src_2 in ATG_1 by t'_G. According to the construction of $T_{ASGra}AG'_1$ would correspond to a modification A' of the ASSIG-algebra A in example 3, where c_2 is mapped by src_{2A} to a_2. But c_2 is already mapped by src_{2A} to b_2, s.t. src_{2A} is no longer a function and A' no longer an ASSIG-algebra. In fact, A' would become a relational ASSIG-algebra and we may obtain an isomorphism in Thm 3 for relational ASSIG-algebras. Finally we obtain similar to Thm 2 a functor TT_{ASGra} from the category $\mathbf{ASSIG\text{-}Alg}_{TA}$ of typed ASSIG-algebras (for a given ASSIG-algebra TA as type algebra) to the category $\mathbf{AGraphs}_{TAG}$, where $TAG = T_{ASGra}(TA)$ according to Theorem 3.

Theorem 4

TT_{ASGra} : **ASSIG-Alg**$_{TA}$ $\rightarrow$ **AGraphs**$_{TAG}$ is a functor based on T_{ASGra} in Thm 3 with $TAG = T_{ASGra}(TA)$.

Remarks

1. More precisely we have $T_{ASGra}(TA) = (TAG, Tt : TAG \rightarrow ATG)$.

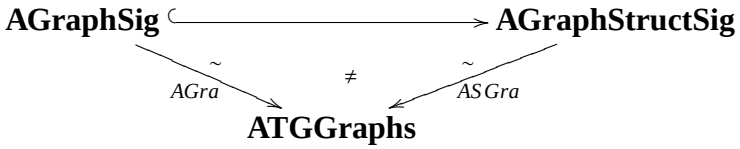
2. Similar to T_{ASGra} in theorem 3 also the functor TT_{ASGra} in the typed case is non surjective in general.

8 Overview of Results

In this section we summarize the transformations on the syntactical level (see lemma 1 and 2) and those on the semantical level (theorems 1-4). It is important to note that the transformations on the syntactical level define bijections between **AGraphSig** and **ATGGraphs** by lemma 1 and between **AGraphStructSig** and **ATGGraphs** by lemma 2 although **AGraphSig** is a proper subclass of category **AGraphStructSig**. This implies that the corresponding diagrams on the semantical level do not commute in general.

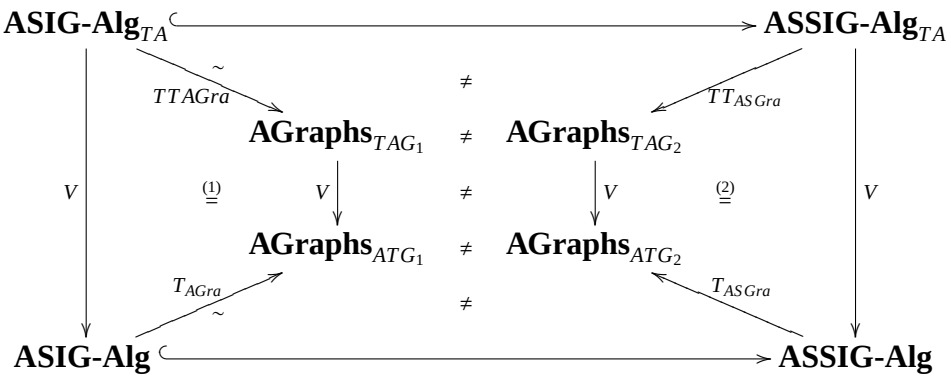
Syntactical Transformations (Lemma 1+2)

Summarizing lemma 1 and 2 we obtain the following non commutative diagram:



Semantical Transformations (Thms 1-4)

Summarizing theorems 1 to 4 we obtain the following digram, where it is important to distinguish $ATG_1 = AGra(ASIG)$ from $ATG_2 = ASGra(ASSIG)$ and similarly $TAG_1 = T_{AGra}(TA)$ from $TAG_2 = T_{ASGra}(TA)$.



Remark

V denotes different kinds of forgetful functors s.t. diagrams (1) and (2) commute, but all the other ones do not commute except of the outer diagram.

References

[1] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. 1991. Parallelism and concurrency in high-level replacement systems. *Math. Struct. in Comp. Science*, Vol. 1, pp.:361–404.

[2] H. Ehrig. 1979. Introduction to the Algebraic Theory of Graph Grammars (A Survey). In *Graph Grammars and their Application to Computer Science and Biology*, Springer LNCS 73, pp.:1–69.

[3] H. Ehrig and B. Mahr. 1985. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer Verlag, Berlin.

[4] R. Heckel, J. Küster, and G. Taentzer. 2002. Towards Automatic Translation of UML Models into Semantic Domains . In H.-J. Kreowski, editor, *Proc. of APPLIGRAPH Workshop on Applied Graph Transformation (AGT 2002)*, pp.: 11 – 22.

[5] M. Löwe, M. Korff, and A. Wagner. 1993. An Algebraic Framework for the Transformation of Attributed Graphs. In M.R. Sleep, M.J. Plasmeijer, and M.C. van Eekelen, editors, *Term Graph Rewriting: Theory and Practice*, chapter 14, John Wiley & Sons Ltd, pp.: 185–199.

[6] G. Rozenberg, editor. 1997. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. World Scientific.