11

Adhesive HLR Categories for Typed Attributed Graphs

In Chapters 8–10, we have presented the main concepts and results for typed attributed graph transformation systems. However, we have postponed most of the proofs, because they have been given already for adhesive HLR systems in Part II. It remains to instantiate them for typed attributed graph transformation systems.

For this purpose, we have to show that the category $\mathbf{AGraphs_{ATG}}$ of typed attributed graphs is an adhesive HLR category. In Theorem 11.3, we show that the category $\mathbf{AGraphs_{ATG}}$ is isomorphic to a category of algebras over a suitable signature $AGSIG(ATG)$, which is uniquely defined by the attributed type graph ATG. In fact, it is much easier to verify the categorical properties of adhesive HLR categories for the category of algebras **AGSIG(ATG)-Alg** and to show the isomorphism between **AGSIG(ATG)-Alg** and $\mathbf{AGraphs_{ATG}}$ than to show the categorical properties directly for the category $\mathbf{AGraphs_{ATG}}$.

In Theorem 11.11, we show that **AGSIG(ATG)-Alg**, and hence also $\mathbf{AGraphs_{ATG}}$, is an adhesive HLR category. In fact, we show this result for the category **AGSIG-Alg**, where $AGSIG$ is a more general kind of attributed graph structure signatures in the sense of [LKW93, CL95, FKTV99]. This allows us to obtain the results of our theory for other kinds of attributed graphs, also. However, we shall not discuss these other instantiations in more detail. By combining the results given in Sections 11.1 and 11.2 with those in Part II, we are able to verify in Section 11.3 that the following basic results stated in Chapters 9 and 10 are valid for typed attributed graph transformations:

1. The Local Church–Rosser, Parallelism, and Concurrency Theorems.
2. The Embedding and Extension Theorems.
3. The completeness of critical pairs and the Local Confluence Theorem.

An alternative way to show that $\mathbf{AGraphs_{ATG}}$ is an adhesive HLR category is given at the end of Section 11.2, where **AGraphs** is represented as a subcategory of a comma category and $\mathbf{AGraphs_{ATG}}$ as a slice category of

AGraphs. However, for showing the existence of initial pushouts, it is easier to give the corresponding construction in the more general context of the category **AGSIG-Alg**. Moreover, this allows us to apply the results to the general attributed graph structure signatures mentioned above.

11.1 Attributed Graph Structure Signatures and Typed Attributed Graphs

Attributed graph structure signatures were introduced in [LKW93] to model attributed graphs and the corresponding transformations. In fact, this concept is general enough to model various kinds of attributed graphs, especially attributed graphs with node attributes only, as presented in [HKT02], and our concept with node and edge attributes introduced in Chapter 8.

In this section, we review attributed graph structure signatures and show that, for each type graph ATG, there is a graph structure signature $AGSIG(ATG)$ such that the category $\mathbf{AGraphs_{ATG}}$ of attributed graphs typed over ATG and the category **AGSIG(ATG)-Alg** of $AGSIG(ATG)$-algebras are isomorphic.

We start with the definition of attributed graph structure signatures.

Definition 11.1 (attributed graph structure signature). *A* graph structure signature $GSIG = (S_G, OP_G)$ *is an algebraic signature with unary operations* $op : s \to s'$ *in* OP_G *only.*

An attributed graph structure signature $AGSIG = (GSIG, DSIG)$ *consists of a graph structure signature* $GSIG$ *and a data signature* $DSIG = (S_D, OP_D)$ *with attribute value sorts* $S'_D \subseteq S_D$ *such that* $S'_D = S_D \cap S_G$ *and* $OP_D \cap OP_G = \varnothing$.

$AGSIG$ *is called* well structured *if, for each* $op : s \to s'$ *in* OP_G, *we have* $s \notin S_D$.

The next steps are to introduce the category **AGSIG-Alg** of attributed graph structure signatures and the special case **AGSIG(ATG)-Alg**, which allows us to construct an isomorphism with the category $\mathbf{AGraphs_{ATG}}$. In Example 11.4, we construct an explicit attributed graph structure signature $AGSIG(ATG)$ for the attributed type graph in Example 8.9.

Definition 11.2 (category AGSIG-Alg). *Given an attributed graph structure signature* $AGSIG = (GSIG, DSIG)$, *the category of all* $AGSIG$*-algebras and* $AGSIG$*-homomorphisms is denoted by* **AGSIG-Alg**, *where* **AGSIG-Alg** *corresponds to the category* $\mathbf{Alg}(\mathbf{\Sigma})$ *(see Definition B.9) with*

$$\Sigma = GSIG \cup DSIG.$$

Theorem 11.3 (isomorphism $\mathbf{AGraphs_{ATG}} \cong$ AGSIG(ATG)-Alg). *For each attributed type graph ATG, there is a well-structured attributed graph structure signature $AGSIG(ATG)$ such that the category* $\mathbf{AGraphs_{ATG}}$ *is isomorphic to the category* **AGSIG(ATG)-Alg***:*
$\mathbf{AGraphs_{ATG}} \cong$ **AGSIG(ATG)-Alg**.

Proof. For a given attributed type graph ATG, we suppose that $S_D \cap V_G^{TG} = \varnothing$ and $S_D \cap E_j^{TG} = \varnothing$ for all $j \in \{G, NA, EA\}$. This means that data sorts cannot be graph node types or the type of any kind of edge. Otherwise, we would rename them accordingly.

We first construct the corresponding attributed graph structure signature $AGSIG(ATG)$. Then we find a functor $F : \mathbf{AGraphs_{ATG}} \to$ **AGSIG(ATG)-Alg** and an inverse functor F^{-1}: **AGSIG(ATG)-Alg** $\to \mathbf{AGraphs_{ATG}}$ that show the isomorphism.

For an attributed type graph $ATG = (TG, Z)$ with a final $DSIG$-algebra Z, a type graph $TG = (V_G^{TG}, V_D^{TG}, E_G^{TG}, E_{NA}^{TG}, E_{EA}^{TG}, (source_j^{TG}, target_j^{TG})_{j \in \{G,NA,EA\}})$ and $S'_D \subseteq S_D$, we define $AGSIG(ATG) = (GSIG, DSIG)$, where $GSIG = (S_G, OP_G)$, $S_G = S_V \dot\cup S_E$, $S_V = V_G^{TG} \dot\cup V_D^{TG}$, $S_E = E_G^{TG} \dot\cup E_{NA}^{TG} \dot\cup E_{EA}^{TG}$, and $OP_G = \dot\cup_{e \in S_E} OP_e$, with $OP_e = \{src_e, tar_e\}$ defined by

- $src_e : e \to v(e)$ for $e \in E_G^{TG}$ with $v(e) = source_G^{TG}(e) \in V_G^{TG}$,
- $tar_e : e \to v'(e)$ for $e \in E_G^{TG}$ with $v'(e) = target_G^{TG}(e) \in V_G^{TG}$,
- src_e, tar_e for $e \in E_{NA}^{TG}$ and $e \in E_{EA}^{TG}$ are defined analogously.

$AGSIG(ATG)$ is a well-structured attributed graph structure signature, since we have only unary operations and, from $V_D^{TG} = \dot\cup_{s \in S'_D} Z_s = S'_D$, $V_G^{TG} \cap S_D = \varnothing$ and $E_j^{TG} \cap S_D = \varnothing$ for all $j \in \{G, NA, EA\}$, we have $S_D \cap S_G = S_D \cap V_D^{TG} = S'_D$ and the well-structuredness follows.

The functor $F :$ $\mathbf{AGraphs_{ATG}} \to$ **AGSIG(ATG)-Alg** is defined, for objects $(AG, t : AG \to ATG)$ with $AG = (G, D)$, $t_G : G \to TG$ and $t_D : D \to Z$, by $F(AG, t) = A$, with the following $AGSIG(ATG)$-algebra A:

- $A_s = t_{G,V_i}^{-1}(s) \subseteq V_i$ for $s \in V_i^{TG} \subseteq S_V$, $i \in \{G, D\}$;
- $A_e = t_{G,E_j}^{-1}(e) \subseteq E_j$ for $e \in E_j^{TG} \subseteq S_E$, $j \in \{G, NA, EA\}$;
- $A_s = t_{D_s}^{-1}(s) = D_s$ for $s \in S_D$;
- $src_e^A(a) = source_G(a)$ for $e \in E_G^{TG} \subseteq S_E$, $a \in t_{G,E_G}^{-1}(e) = A_e \subseteq E_G$;
- $tar_e^A(a) = target_G(a)$ for $e \in E_G^{TG}$, $a \in t_{G,E_G}^{-1}(e) = A_e \subseteq E_G$;
- analogously for $src_e^A(a)$, $tar_e^A(a)$, with $e \in E_j^{TG}$, $a \in t_{G,E_j}^{-1}(e)$, $j \in \{NA, EA\}$;
- $op^A = op^D$ for all $op \in OP_D$.

For a typed attributed graph morphism $f : (AG^1, t^1) \to (AG^2, t^2)$, we have $F(f) = h : F(AG^1, t^1) = A \to F(AG^2, t^2) = B$; h is an algebra homomorphism defined by

- $h_s(a) = f_{G,V_i}(a)$ for $s \in V_i^{TG} \subseteq S_V$, $a \in A_s$, $i \in \{G, D\}$,
- $h_e(a) = f_{G,E_j}(a)$ for $e \in E_j^{TG} \subset S_E$, $a \in A_e$, $j \in \{G, NA, EA\}$,
- $h_s = f_{D,s}$ for $s \in S_D$.

In the other direction, the functor F^{-1} is defined for an $AGSIG(ATG)$-algebra A by $F^{-1}(A) = (AG = (G, D), t : AG \to ATG)$, with

- $V_i = \dot{\cup}_{s \in V_i^{TG}} A_s$, $E_j = \dot{\cup}_{e \in E_j^{TG}} A_e$ for $i \in \{G, D\}$, $j \in \{G, NA, EA\}$;
- $source_j(a) = src_e^A(a)$ for $e \in E_j^{TG}$, $a \in A_e$, $j \in \{G, NA, EA\}$;
- $target_j(a) = tar_e^A(a)$ for $e \in E_j^{TG}$, $a \in A_e$, $j \in \{G, NA, EA\}$;
- $t_{G,V_i}(a) = s$ for $a \in A_s$, $s \in V_i^{TG}$, $i \in \{G, D\}$;
- $t_{G,E_j}(a) = e$ for $a \in A_e$, $e \in E_j^{TG}$, $j \in \{G, NA, EA\}$;
- $D = A|_{DSIG}$;
- $t_{D,s}(a) = s$ for $a \in A_s$, $s \in S_D$.

For a homomorphism $h : A \to B$, we define $F^{-1}(h) = f : F^{-1}(A) \to F^{-1}(B)$ by

- $f_{G,V_i}(a) = h_s(a)$ for $a \in A_s$, $s \in V_i^{TG}$, $i \in \{G, D\}$;
- $f_{G,E_j}(a) = h_e(a)$ for $a \in A_e$, $e \in E_j^{TG}$, $j \in \{G, NA, EA\}$;
- $f_D = h|_{DSIG}$.

The constructed morphisms F and F^{-1} are well defined; they are actually functors and isomorphisms (as proven in Section C.3). □

Example 11.4 (corresponding $AGSIG(ATG)$ and algebra). We present the corresponding signature $AGSIG(ATG)$ for the the type graph ATG defined in Example 8.9 and the resulting $AGSIG(ATG)$-algebra for the typed attributed graph (AG, t).

$AGSIG(ATG) = (GSIG, DSIG)$ for type graph $ATG = (TG, Z)$ has the same data signature $DSIG$. $GSIG$ has the following structure:

$$
\begin{array}{lll}
GSIG: & sorts: & Method, Parameter, Class, \\
 & & string, nat, parameterDirectionKind, \\
 & & param, type, noOfPars, mname, pname, cname, \\
 & & kind, order \\
 & opns: & src_{param} : param \to Method \\
 & & tar_{param} : param \to Parameter \\
 & & src_{type} : type \to Parameter \\
 & & tar_{type} : type \to Class \\
 & & \vdots \\
 & & src_{order} : order \to param \\
 & & tar_{order} : order \to nat
\end{array}
$$

All node and edge types in the type graph correspond to a sort, and for all edge types we define operation symbols that describe the types of the source and target.

The corresponding $AGSIG(ATG)$-algebra A for $AG = (G, D)$ is defined as follows:

$$
\begin{array}{lll}
A: & A_{Method} & = \{m\} \\
 & A_{Parameter} & = \{par_1, par_2, par_3\} \\
 & A_{Class} & = \{c\} \\
 & A_{nat} & = D_{nat} \\
 & \vdots & \\
 & A_{param} & = \{mpar_1, mpar_2, mpar_3\} \\
 & A_{type} & = \{par_1c, par_2c, par_3c\} \\
 & \vdots & \\
 & A_{order} & = \{order_1, order_2, order_3\} \\
 & A_s & = D_s \text{ for all } s \in S_D \\
 & & \\
 & src^A_{param} & : A_{param} \to A_{Method} \ : \ mpar_i \mapsto m \\
 & tar^A_{param} & : A_{param} \to A_{Parameter} \ : \ mpar_i \mapsto par_i \\
 & src^A_{type} & : A_{type} \to A_{Parameter} \ : \ par_ic \mapsto par_i \\
 & tar^A_{type} & : A_{type} \to A_{Class} \ : \ par_ic \mapsto c \\
 & \vdots & \\
 & src^A_{order} & : A_{order} \to A_{param} \ : \ order_i \mapsto mpar_i \\
 & tar^A_{order} & : A_{order} \to A_{nat} \ : \ order_i \mapsto i \\
 & op_A & = op_D \text{ for all } op \in OP_D
\end{array}
$$

□

11.2 Definition of Concrete Adhesive HLR Categories

In this section, we consider a fixed attributed graph structure signature $AGSIG = (GSIG, DSIG)$. We prove that the category **AGSIG-Alg** over $AGSIG$ with a distinguished class $\mathcal{M}$ (defined in the following) fulfills all properties of an adhesive HLR category. By Theorem 11.3, $\mathbf{AGraphs_{ATG}}$ is also an adhesive HLR category.

Finally, we sketch an alternative way to show that $\mathbf{AGraphs_{ATG}}$ is an adhesive HLR category using comma categories.

Definition 11.5 (class $\mathcal{M}$ in AGSIG-Alg and $\mathbf{AGraphs_{ATG}}$). *The class $\mathcal{M}$ in* **AGSIG-Alg** *is the class of all algebra homomorphisms $f = (f_{GSIG}, f_{DSIG})$, where f_{GSIG} is injective and f_{DSIG} is an isomorphism. The notation $f = (f_{GSIG}, f_{DSIG})$ means that f_{GSIG} and f_{DSIG} are the restrictions of f to GSIG and DSIG, respectively, where the two restrictions coincide on $S'_D = S_D \cap S_G$.*

In $\mathbf{AGraphs_{ATG}}$*, the morphism class $\mathcal{M}$ is the class of all morphisms $f = (f_G, f_D)$, where f_G is injective and f_D is an isomorphism on the data part (see Definition 8.10).*

Remark 11.6. We use the same notation for the morphism classes $\mathcal{M}$ in $\mathbf{AGraphs_{ATG}}$ and in **AGSIG(ATG)-Alg** because they correspond to each other, owing to the construction of the functors F and F^{-1} in the proof of Theorem 11.3.

We prove step by step the properties necessary for an adhesive HLR category. First we check the closure properties of $\mathcal{M}$.

Lemma 11.7 (properties of $\mathcal{M}$). *The class $\mathcal{M}$ in* **AGSIG-Alg** *as defined in Definition 11.5 is closed under isomorphisms, composition, and decomposition.*

Proof. An algebra homomorphism is injective or isomorphic if all its components are injective or isomorphic, respectively, in **Sets**. In **Sets**, the class of injective morphisms and the class of isomorphic morphisms are closed under isomorphism, composition and decomposition. Therefore this property holds for the class $\mathcal{M}$ of injective homorphisms with an isomorphic data part. □

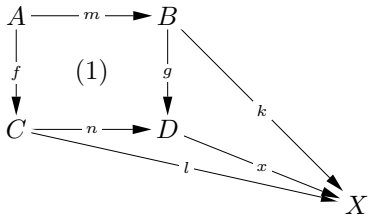
For the second property, we need to the prove existence and closedness of pushouts and pullbacks along $\mathcal{M}$-morphisms. This is done with the following lemmas, where we show that pushouts along $\mathcal{M}$-morphisms and pullbacks can be constructed componentwise in **Sets**. Note that general pushouts in **AGSIG-Alg** exist, but in general cannot be constructed componentwise, which is essential for the proof of the VK property in Lemma 11.10. On the other hand, general pullbacks in **AGSIG-Alg** can be constructed componentwise.

Lemma 11.8 (POs in AGSIG-Alg along $\mathcal{M}$-morphisms). *For given morphisms $m : A \to B \in \mathcal{M}$ and $f : A \to C$, there is a pushout (1) in* **AGSIG-Alg** *with $n \in \mathcal{M}$:*

$$\begin{array}{ccc} A & \xrightarrow{m} & B \\ {\scriptstyle f}\downarrow & (1) & \downarrow{\scriptstyle g} \\ C & \xrightarrow{n} & D \end{array}$$

Moreover, given that (1) is commutative with $m \in \mathcal{M}$, then (1) is a pushout in **AGSIG-Alg** *iff (1) is a componentwise pushout in* **Sets**. *Then $m \in \mathcal{M}$ implies $n \in \mathcal{M}$.*

Proof. **Part 1.** If (1) is commutative, $m \in \mathcal{M}$, and $(1)_s$ are componentwise pushouts in **Sets**, we can show that (1) is a pushout in **AGSIG-Alg**. Consider an object X with morphisms $k : B \to X$ and $l : C \to X$ such that $k \circ m = l \circ f$:



Then, for each $s \in S_G \cup S_D$, there exists a unique $x_s : D_s \to X_s$ such that $x_s \circ g_s = k_s$ and $x_s \circ n_s = l_s$.

We show that $x = (x_s)_{s \in S_G \cup S_D}$ is a homomorphism as follows:

1. $op \in OP_D$. For $s \in S_D$, m_s being an isomorphism implies that n_s is an isomorphism. This gives the compatibility of x with $op \in OP_D$, because l is a homomorphism.
2. $op \in OP_G$. Since $(1)_s$ is a pushout, n_s and g_s are jointly surjective. This means that for every $d \in D_S$ there is a $b \in B_s$ with $g_s(b) = d$ or a $c \in C_s$ with $n_s(c) = d$. Then, for $op : s \to s' \in OP_G$ it holds that $op_X(x_s(d)) = op_X(x_s(g_s(b))) = op_X(k_s(b)) = k_{s'}(op_B(b)) = x_{s'}(g_{s'}(op_B(b))) = x_{s'}(op_D(g_s(b))) = x_{s'}(op_D(d))$ or $op_X(x_s(d)) = op_X(x_s(n_s(c))) = op_X(l_s(c)) = l_{s'}(op_C(c)) = x_{s'}(n_{s'}(op_C(c))) = x_{s'}(op_D(n_s(c))) = x_{s'}(op_D(d))$.

x is unique, since all its components are unique, and therefore (1) is a pushout in **AGSIG-Alg**.

Part 2. Now we construct a pushout object D with morphisms n and g for given objects and morphisms $C \overset{f}{\leftarrow} A \overset{m}{\to} B$, with $m \in \mathcal{M}$. For all $s \in S_D$, let $D_s = C_s$, $g_s = f_s \circ m_s^{-1}$, and $n_s = id_{C_s}$. For $op \in OP_D$, we define $op_D = op_C$. Since m_s is an isomorphism and n_s is the identity, $(1)_s$ is obviously a pushout in **Sets**.

For $s \in S_G \backslash S_D$, let $B_s \overset{g_s}{\to} D_s \overset{n_s}{\leftarrow} C_s$ be the pushout over $C_s \overset{f_s}{\leftarrow} A_s \overset{m_s}{\to} B_s$ in **Sets**, and for $op : s \to s' \in OP_G$ we define

$$op_D(d) = \begin{cases} n_{s'} \circ f_{s'}(op_A(a)) & : \exists a \in A_s : n_s(f_s(a)) = d \\ g_{s'}(op_B(b)) & : \exists b \in B_s \backslash m_s(A_s) : g_s(b) = d \\ n_{s'}(op_C(c)) & : \exists c \in C_s \backslash f_s(A_s) : n_s(c) = d \end{cases}$$

We have to show that these operations are well defined. Since $(1)_s$ is a pushout, exactly one of the cases above applies. In the second or the third case b or c, respectively, must be unique. In the first case we have the result that m_s being injective implies that n_s is injective. For $a_1, a_2 \in A_s$ with $n_s(f_s(a_1)) = n_s(f_s(a_2)) = d$, it holds that $f_s(a_1) = f_s(a_2)$. Then $n_{s'} \circ f_{s'}(op_A(a_1)) = n_{s'}(op_C(f_s(a_1))) = n_{s'}(op_C(f_s(a_2))) = n_{s'} \circ f_{s'}(op_A(a_2))$ follows.

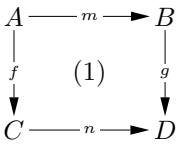
$n = (n_s)_{s \in S_G \cup S_D}$ is a homomorphism. This is clear for all $op \in OP_D$. Consider an operation $op : s \to s' \in OP_G$. Then, for all $c \in C_s$, it holds that $op_D(n_s(c)) = n_{s'}(op_C(c))$: if $c \in C_s \backslash f_s(A_s)$, then $op_D(n_s(c)) = n_{s'}(op_C(c))$ by

definition. Otherwise, $c \in f_s(A_s)$. There then exists an $a \in A_s$ with $f_s(a) = c$, and it holds that $op_D(n_s(c)) = op_D(n_s(f_s(a))) \overset{Def.}{=} n_{s'}(f_{s'}(op_A(a))) = n_{s'}(op_C(f_s(a))) = n_{s'}(op_C(c))$. Obviously, we have $n \in \mathcal{M}$. That $g = (g_s)_{s \in S_G \cup S_D}$ is a homomorphism follows analogously.

Since $g \circ m = n \circ f$ and we have componentwise pushouts, it follows by part 1 that (1) with the constructed D, n, and g is a pushout in **AGSIG-Alg**.

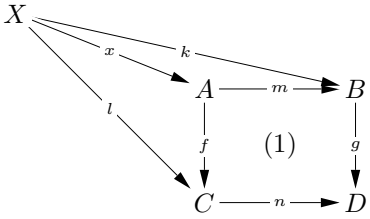
Part 3. Let (1) be a pushout in **AGSIG-Alg** with $m \in \mathcal{M}$. By the construction in part 2 and the uniqueness of pushouts up to isomorphism, it follows directly that $n \in \mathcal{M}$ and that (1) is a componentwise pushout in **Sets**. □

Lemma 11.9 (PBs in AGSIG-Alg). *Given $g : B \to D$ and $n : C \to D$, then there is a pullback (1) in* **AGSIG-Alg***:*



Moreover, given that (1) is commutative, then (1) is a pullback in **AGSIG-Alg** *iff (1) is a componentwise pullback in* **Sets**. *If $n \in \mathcal{M}$, then $m \in \mathcal{M}$ also.*

Proof. **Part 1.** If (1) is commutative and $(1)_s$ are componentwise pullbacks in **Sets**, we can show that (1) is a pullback in **AGSIG-Alg**. Consider an object X with morphisms $k : X \to B$ and $l : X \to C$ such that $n \circ l = g \circ k$:



Then, for each $s \in S_G \cup S_D$, there exists a unique $x_s : X_s \to A_s$ such that $m_s \circ x_s = k_s$ and $f_s \circ x_s = l_s$.

We show that $x = (x_s)_{s \in S_G \cup S_D}$ is a homomorphism. For each operation $op : s_1 ... s_n \to s \in OP_G \cup OP_D$ and $y_i \in X_{s_i}$, it holds that

- $f_s(x_s(op_X(y_1, ..., y_n))) = l_s(op_X(y_1, ..., y_n)) = op_C(l_{s_1}(y_1), ..., l_{s_n}(y_n)) = op_C(f_{s_1}(x_{s_1}(y_1)), ..., f_{s_n}(x_{s_n}(y_n))) = f_s(op_A(x_{s_1}(y_1), ..., x_{s_n}(y_n)))$ and
- $m_s(x_s(op_X(y_1, ..., y_n))) = k_s(op_X(y_1, ..., y_n)) = op_B(k_{s_1}(y_1), ..., k_{s_n}(y_n)) = op_B(m_{s_1}(x_{s_1}(y_1)), ..., m_{s_n}(x_{s_n}(y_n))) = m_s(op_A(x_{s_1}(y_1), ..., x_{s_n}(y_n)))$.

Since $(1)_s$ is a pullback in **Sets**, m_s and f_s are jointly injective (see Fact 2.23). This means that if $m_s(a_1) = m_s(a_2)$ and $f_s(a_1) = f_s(a_2)$, it follows that $a_1 = a_2$. Therefore $x_s(op_X(y_1, ..., y_n)) = op_A(x_{s_1}(y_1), ..., x_{s_n}(y_n))$, and x is a homomorphism.

x is unique, since all its components are unique. Therefore (1) is a pullback in **AGSIG-Alg**.

Part 2. Now we construct a pullback object A with morphisms f and m for given objects and morphisms $C \overset{n}{\to} D \overset{g}{\leftarrow} B$.

For $s \in S_G \cup S_D$, let $B_s \overset{m_s}{\leftarrow} A_s \overset{f_s}{\to} C_s$ be the pullback over $C_s \overset{n_s}{\to} D_s \overset{g_s}{\leftarrow} B_s$ in **Sets**. For $op : s_1...s_n \to s \in OP_G \cup OP_D$, we define $op_A(a_1, ..., a_n) = a$, with $op_B(m_{s_1}(a_1), ..., m_{s_n}(a_n)) = m_s(a)$ and $op_C(f_{s_1}(a_1), ..., f_{s_n}(a_n)) = f_s(a)$. This a exists and is unique, since $(1)_s$ is a pullback in **Sets**.

$f = (f_s)_{s \in S_G \cup S_D}$ and $m = (m_s)_{s \in S_G \cup S_D}$ are homomorphisms by construction. Since $g \circ m = n \circ f$ and we have componentwise pullbacks, it follows by part 1 that (1), with the constructed object A and morphisms f and m is a pullback in **AGSIG-Alg**.

Part 3. Let (1) be a pullback in **AGSIG-Alg**. By the construction in part 2 and the uniqueness of pullbacks up to isomorphism, it follows directly that (1) is a componentwise pullback in **Sets**.

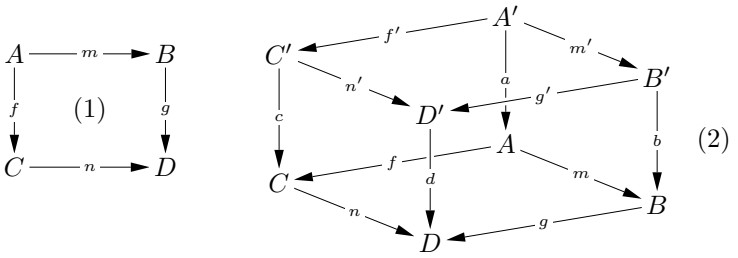
Part 4. If $n \in \mathcal{M}$, we have the result that n_s is injective for all $s \in S_G$, and it is an isomorphism for $s \in S_D$. Since pullbacks in **Sets** are closed under monomorphisms and isomorphisms, it follows that m_s is injective for $s \in S_G$ and is an isomorphism for $s \in S_D$, which means that $m \in \mathcal{M}$. □

Lemma 11.10 (VK property of POs along $\mathcal{M}$-morphisms). *A pushout in* **AGSIG-Alg** *along an $\mathcal{M}$-morphism is a VK square.*

Proof. Consider the pushout (1) below with $m \in \mathcal{M}$, and the commutative cube (2), where (1) is in the bottom and the back faces are pullbacks.

Part 1. If the front faces are pullbacks, then the top is a pushout.

Let the front faces be pullbacks. By applying Lemmas 11.8 and 11.9, we can decompose the cube for every component $s \in S_G \cup S_D$ such that the bottom is a pushout with $m_s \in \mathcal{M}$, and the front and back faces are pullbacks in **Sets**. Then $(1)_s$ is a VK square in **Sets**. From the VK square property, we obtain the top as a componentwise pushout. The fact that $m \in \mathcal{M}$ and the back right square is a pullback implies that $m' \in \mathcal{M}$, owing to Lemma 11.9. By Lemma 11.8, the top of cube (2) is a pushout in **AGSIG-Alg**.



Part 2. If the top is a pushout, then the front faces are pullbacks.

Let the top be a pushout. The fact that $m \in \mathcal{M}$ and the back right is a pullback implies, from Lemma 11.9, that $m' \in \mathcal{M}$. By applying Lemmas 11.8 and 11.9, we can decompose the cube for every component $s \in S_G \cup S_D$ such that the bottom and the top are pushouts, $m_s \in \mathcal{M}$ and the back faces are pullbacks in **Sets**. Then $(1)_s$ is a VK square in **Sets**. From the VK square property, we obtain the result that the front faces are componentwise pullbacks. By Lemma 11.9, the front faces of cube (2) are pullbacks in **AGSIG-Alg**. □

Theorem 11.11 (AGSIG-Alg and $\mathbf{AGraphs_{ATG}}$ are adhesive HLR categories). *The categories* (**AGSIG-Alg**, $\mathcal{M}$) *and* ($\mathbf{AGraphs_{ATG}}$, $\mathcal{M}$) *with morphism classes* $\mathcal{M}$ *as defined in Definition 11.5 are adhesive HLR categories.*

Proof. In **AGSIG-Alg**, $\mathcal{M}$ is a class of monomorphisms since monomorphisms in **AGSIG-Alg** are componentwise monomorphisms in **Sets**, which means that they are componentwise injective (and isomorphisms are both injective and surjective). The closure properties of $\mathcal{M}$ are explicitly proven in Lemma 11.7.

The existence and closedness of pushouts and pullbacks along $\mathcal{M}$-morphisms follow from Lemmas 11.8 and 11.9.

In Lemma 11.10, it is shown that pushouts along $\mathcal{M}$-morphisms are VK squares.

Therefore (**AGSIG-Alg**, $\mathcal{M}$) is an adhesive HLR category.

For each attributed type graph ATG there is by Theorem 11.3, a corresponding graph structure signature $AGSIG(ATG)$ such that $\mathbf{AGraphs_{ATG}}$ is isomorphic to **AGSIG(ATG)-Alg**. The morphism classes $\mathcal{M}$ in $\mathbf{AGraphs_{ATG}}$ and **AGSIG(ATG)-Alg** are isomorphic. Therefore ($\mathbf{AGraphs_{ATG}}$, $\mathcal{M}$) is also an adhesive HLR category. □

Finally, let us sketch an alternative way to show that (**AGraphs**, $\mathcal{M}$) and ($\mathbf{AGraphs_{ATG}}$, $\mathcal{M}$) are adhesive HLR categories. First we use a comma category construction to show this for **AGraphs**, and then we use a slice category construction for $\mathbf{AGraphs_{ATG}}$.

Fact 11.12 (comma category construction for AGraphs). *The category* **AGraphs** *is isomorphic to a subcategory* $\mathbf{ComCat}(\mathbf{V_1}, \mathbf{V_2}; \mathbf{Id})$ *of the comma*

category $\mathbf{ComCat}(\mathbf{V_1}, \mathbf{V_2}; \mathbf{I})$ *defined below, where* $I = \{1\}$*, which implies that* (**AGraphs**, $\mathcal{M}$) *with* $\mathcal{M}$ *as defined in Definition 8.10 is an adhesive HLR category.*

Construction. Let V_1 and V_2 be the forgetful functors defined by

- V_1 : **E-Graphs** $\to$ **Sets** with $V_1(G) = V_D^G$ and $V_1(f_G) = f_{G,V_D}$;
- V_2 : **DSIG-Alg** $\to$ **Sets** with $V_2(D) = \dot{\cup}_{s \in S'_D} D_s$ and $V_2(f_D) = \dot{\cup}_{s \in S'_D} f_{D_s}$.

$\mathbf{ComCat}(\mathbf{V_1}, \mathbf{V_2}; \mathbf{Id})$ is the subcategory of $\mathbf{ComCat}(\mathbf{V_1}, \mathbf{V_2}; \mathbf{I})$ where $I = \{1\}$ and the objects $(G, D, op : V_1(G) \to V_2(D))$ satisfy $V_1(G) = V_2(D)$ and $op = id$. □

Proof. For attributed graphs $AG = (G, D)$, we have, by Definition 8.10, $\dot{\cup}_{s \in S'_D} D_s = V_D^G$, and for morphisms $f = (f_G, f_D) : AG^1 \to AG^2$, we have commutativity of (1) for all $s \in S'_D$, which is equivalent to commutativity of (2):

$$\begin{array}{ccc} D_s^1 & \xrightarrow{f_{D_s}} & D_s^2 \\ \downarrow & (1) & \downarrow \\ V_D^1 & \xrightarrow{f_{G,V_D}} & V_D^2 \end{array} \qquad \begin{array}{ccc} \dot{\cup}_{s \in S'_D} D_s^1 & \xrightarrow{\dot{\cup} f_{D_s}} & \dot{\cup}_{s \in S'_D} D_s^2 \\ \downarrow{\scriptstyle id} & (2) & \downarrow{\scriptstyle id} \\ V_D^1 & \xrightarrow{f_{G,V_D}} & V_D^2 \end{array}$$

Using $V_1(G) = V_2(D)$, we have $V_D^i = \dot{\cup}_{s \in S'_D} D_s^i$ for $i = 1, 2$, and (2) expresses exactly the compatibility of the morphisms f_G and f_D in $\mathbf{ComCat}(\mathbf{V_1}, \mathbf{V_2}; \mathbf{ID})$, because $V_1(f_G) = f_{G,V_D}$ and $V_2(f_D) = \dot{\cup}_{s \in S'_D} f_{D_1}$. This implies that
$\mathbf{AGraphs} \cong \mathbf{ComCat}(\mathbf{V_1}, \mathbf{V_2}; \mathbf{ID})$.

Let $\mathcal{M}_1$ be the class of injective E-graph morphisms and $\mathcal{M}_2$ the class of $DSIG$-isomorphisms. Then (**E-Graphs**, $\mathcal{M}_1$) is a functor category over (**Sets**, $\mathcal{M}_1$) and hence an adhesive HLR category by Theorem 4.15, item 3. Moreover, each category (**C**, $\mathcal{M}_{iso}$) with the class $\mathcal{M}_{iso}$ of all isomorphisms is an adhesive HLR category; hence, so is (**DSIG-Alg**, $\mathcal{M}_2$). This implies, by Theorem 4.15 item 4, that $\mathbf{ComCat}(\mathbf{V_1}, \mathbf{V_2}; \mathbf{I})$ with $I = \{1\}$ and $\mathcal{M} = (\mathcal{M}_1 \times \mathcal{M}_2) \cap Mor_{\mathbf{ComCat}(\mathbf{V_1}, \mathbf{V_2}; \mathbf{I})}$ is an adhesive HLR category, provided that V_1 preserves pushouts along $\mathcal{M}_1$-morphisms and V_2 preserves pullbacks. But pushouts in **E-graphs** are constructed componentwise in **Sets**; hence V_1 preserves pushouts. Also, pullbacks in **DSIG-Alg** are constructed componentwise, and the disjoint union functor $\dot{\cup}_{s \in S'_D} : \mathbf{Sets}^{S'_D} \to \mathbf{Sets}$ preserves pullbacks; therefore V_2 preserves pullbacks. This implies that $\mathbf{ComCat}(\mathbf{V_1}, \mathbf{V_2}; \mathbf{I})$ with $I = \{1\}$ and $\mathcal{M}$ as above, which corresponds to $\mathcal{M}$ in **AGraphs**, is an adhesive HLR category. Finally, it can be shown that a special choice of pushouts and pullbacks in **E-graphs** and **DSIG-Alg** leads also to pushouts and pullbacks in the subcategory $\mathbf{ComCat}(\mathbf{V_1}, \mathbf{V_2}; \mathbf{ID})$, which allows us to

conclude that $(\mathbf{ComCat}(\mathbf{V_1}, \mathbf{V_2}; \mathbf{ID}), \mathcal{M})$ and hence also (**AGraphs**, $\mathcal{M}$) are adhesive HLR categories. □

Fact 11.13 (slice category construction for $\mathbf{AGraphs_{ATG}}$). *The category (**AGraphs**$_{\mathbf{ATG}}$, $\mathcal{M}$) is a slice category of (**AGraphs**, $\mathcal{M}$) with $\mathcal{M}$ as defined in Definition 8.10, and hence it is an adhesive HLR category.*

Proof. This is a direct consequence of Theorem 4.15, item 2, and Fact 11.12. □

11.3 Verification of the Main Results for Typed AGT Systems

In this section, we give proofs for the results in Chapters 9 and 10 based on the corresponding results for adhesive HLR categories and systems in Part II. In Section 11.2, we have shown that the categories (**AGSIG-Alg**, $\mathcal{M}$) (see Definition 11.2) and $(\mathbf{AGraphs_{ATG}}, \mathcal{M})$ are adhesive HLR categories with a morphism class $\mathcal{M}$ as defined in Definition 11.5.

This is already sufficient to prove the Local Church–Rosser and Embedding Theorems. For the Parallelism Theorem, we need binary coproducts compatible with $\mathcal{M}$ in addition. For the construction of the boundary and context and the Extension Theorem, we also use initial pushouts. Finally, we need an $\mathcal{E}'$–$\mathcal{M}'$ pair factorization for the Concurrency Theorem, the completeness of critical pairs, and the Local Confluence Theorem. We show all these properties for the category **AGSIG-Alg**, where in some cases we need to require that *AGSIG* is well structured. By Theorem 11.3, the results are also true for the category $\mathbf{AGraphs_{ATG}}$.

Theorem 11.14 (main results for typed AGT systems). *Given a typed attributed graph transformation system $GTS = (DSIG, ATG, P)$, we have the following results:*

1. *Local Church–Rosser Theorem (Theorem 9.14);*
2. *Parallelism Theorem (Theorem 9.18);*
3. *Concurrency Theorem (Theorem 9.26);*
4. *E-related transformations (Fact 9.27);*
5. *Embedding Theorem (Theorem 10.10);*
6. *Extension Theorem (Theorem 10.11);*
7. *completeness of critical pairs (Theorem 10.14);*
8. *Local Confluence Theorem (Theorem 10.18).*

Proof.

1. This follows from Theorems 5.12 and 11.11.
2. This follows from Theorems 5.18 and 11.11 and Lemma 11.15 below.
3. This follows from Theorems 5.23 and 11.11.

4. This follows from Fact 5.29, Theorem 11.11, and Lemma 11.16 below.
5. This follows from Theorems 6.14 and 11.11.
6. This follows from Theorems 6.16 and 11.11 and Lemma 11.17 below.
7. This follows from Lemma 6.22 and Theorem 11.11.
8. This follows from Theorems 6.28 and 11.11 and Lemma 11.17.

□

It remains to state and prove all the lemmas which are needed in the proof of Theorem 11.14 above.

Lemma 11.15 (binary coproducts compatible with $\mathcal{M}$). *Given a well-structured attributed graph structure signature AGSIG, then the categories* **AGSIG-Alg** *and* $\mathbf{AGraphs_{ATG}}$ *have binary coproducts compatible with $\mathcal{M}$. This means that f, $g \in \mathcal{M}$ implies $f + g \in \mathcal{M}$.*

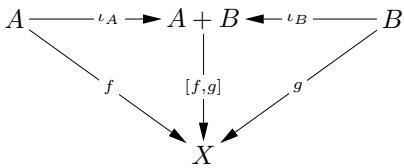
Proof. By Theorem 11.3, it suffices to show the property for **AGSIG-Alg**. Given algebras A and B in (**AGSIG-Alg**, $\mathcal{M}$) with a well-structured attributed graph structure signature $AGSIG$, for all sorts $s \in S_G \backslash S_D$ we construct the componentwise coproduct $(A + B)_s = A_s + B_s$ with coproduct injections $\iota_{A,s}$ and $\iota_{B,s}$ in **Sets**. For the data part, we construct the algebra coproduct $(A+B)_D = A|_{DSIG} + B|_{DSIG}$ with coproduct injections $\iota_{A,D}$ and $\iota_{B,D}$.

For the coproduct $A + B$, we combine these components into the coproduct object $A + B = (((A + B)_s)_{s \in S_G \backslash S_D}, (A + B)_D)$ with morphisms $\iota_i = ((\iota_{i,s})_{s \in S_G \backslash S_D}, \iota_{i,D})$ for $i = A, B$. For $op : s \to s' \in OP_G$ the operation op_{A+B} is defined by

$$op_{A+B} : (A + B)_s \to (A + B)_{s'} : x \mapsto \begin{cases} \iota_{A,s'}(op_A(y)) & : \exists y \in A_s : \iota_{A,s}(y) = x \\ \iota_{B,s'}(op_B(y)) & : \exists y \in B_s : \iota_{B,s}(y) = x \end{cases}.$$

This is well defined, since $AGSIG$ is well structured and therefore it holds that $s \in S_G \backslash S_D$ and $(A + B)_s = A_s \dot\cup B_s$. If $s' \in S_D$, we know that $A_{s'} \dot\cup B_{s'} \subseteq (A+B)_{D,s'}$; otherwise, $s' \in S_G \backslash S_D$ and we have $(A+B)_{s'} = A_{s'} \dot\cup B_{s'}$.

We have to show that the constructed object $A+B$ is indeed a coproduct. Consider morphisms $f : A \to X$ and $g : B \to X$ as in the following diagram. There then has to be a unique morphism $[f, g] : A + B \to X$ such that $[f, g] \circ \iota_A = f$ and $[f, g] \circ \iota_B = g$:



Since $(A+B)_D$ is the coproduct for the data part, there is a morphism $[f,g]_D : (A+B)_D \to X|_{DSIG}$ with $[f,g]_D \circ \iota_{A,D} = f|_{DSIG}$ and $[f,g]_D \circ \iota_{B,D} = g|_{DSIG}$. Similarly, we have for each $s \in S_G \backslash S_D$ a morphism $[f,g]_s : (A+B)_s \to X_s$ such that $[f,g]_s \circ \iota_{A,s} = f_s$ and $[f,g]_s \circ \iota_{B,s} = g_s$.

For the data operations, it is clear that $[f,g]_D$ is a homomorphism. Consider an operation $op : s \to s' \in OP_G$. Since $s \in S_G \backslash S_D$, it holds that $(A+B)_s = A_s \dot\cup B_s$. For an $x \in (A+B)_s$, suppose without loss of generality that there is a $y \in A_s$ with $\iota_{A,s}(y) = x$. We then have the result that $[f,g]_{s'}(op_{A+B}(x)) = [f,g]_{s'}(\iota_{A,s'}(op_A(y))) = f_{s'}(op_A(y)) = op_X(f_s(y)) = op_X([f,g]_s(\iota_{A,s}(y))) = op_X([f,g]_s(x))$. Therefore $[f,g] = (([f,g]_s)_{s \in S_G \backslash S_D}, [f,g]_D)$ is a homomorphism.

The result that $[f,g] \circ \iota_A = f$ and $[f,g] \circ \iota_B = g$ follows by definition, and $[f,g]$ is unique, since all its components are unique. Therefore $[f,g]$ is the required morphism, and $A+B$ is the coproduct of A and B in **AGSIG-Alg**.

It remains to show the compatibility with $\mathcal{M}$. Given $f : A \to A'$ and $g : B \to B'$ with $f,g \in \mathcal{M}$, we construct the coproduct morphism $f+g : A+B \to A'+B'$. In **Sets**, binary coproducts are compatible with monomorphisms, and therefore $(f+g)_s = [\iota_{A',s} \circ f_s, \iota_{B',s} \circ g_s]$ is injective for all $s \in S_G \backslash S_D$.

The coproduct can be considered as a functor and hence preserves isomorphisms. Therefore, if $f|_{DSIG}$ and $g|_{DSIG}$ are isomorphisms, $(f+g)|_{DSIG} = [(\iota_{A'} \circ f)|_{DSIG}, (\iota_{B'} \circ g)|_{DSIG}] = f|_{DSIG} + g|_{DSIG}$ is also an isomorphism.

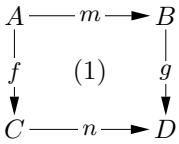
If *AGSIG* is not well structured, we still have binary coproducts in **AGSIG-Alg**, but they may not be compatible with $\mathcal{M}$. □

Lemma 11.16 (closure properties of $\mathcal{M}$–$\mathcal{M}'$ and PO–PB decompositions). *Let $\mathcal{M}' = \mathcal{M}'_1$ or $\mathcal{M}' = \mathcal{M}'_2$ in* $\mathbf{AGraphs_{ATG}}$ *as given in Example 9.22 or consider the corresponding morphism classes in* **AGSIG-Alg**. *We then have:*

1. *$\mathcal{M}'$ is closed under pushouts and pullbacks along $\mathcal{M}$-morphisms in* **AGSIG-Alg** *and* $\mathbf{AGraphs_{ATG}}$.
2. *The $\mathcal{M}$–$\mathcal{M}'$ pushout–pullback decomposition property holds in the categories* **AGSIG-Alg** *and* $\mathbf{AGraphs_{ATG}}$.
3. *$\mathcal{M}'$ is closed under composition and decomposition.*

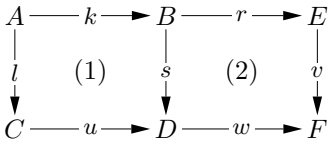
Proof. It suffices to prove the lemma for **AGSIG-Alg** with the class $\mathcal{M}'_2$. For the class $\mathcal{M}'_1 = \mathcal{M}$ of all monomorphisms, the lemma follows from the properties of adhesive HLR categories.

1. Given the pushout (1) below, with $m,n \in \mathcal{M}$ and $f \in \mathcal{M}'_2$, Lemma 11.8 implies that (1) is a componentwise pushout in **Sets**. In **Sets**, if f_s is injective, so is g_s. This means that g_s is injective for all $s \in S_G \backslash S_D$, and therefore $g \in \mathcal{M}'_2$.



It follows analogously, by Lemma 11.9, that if (1) is a pullback with $m, n \in \mathcal{M}$ and $g \in \mathcal{M}'_2$, then $f \in \mathcal{M}'_2$ also.

2. Consider the following diagram, where $l \in \mathcal{M}$, $w \in \mathcal{M}'_2$, (1) + (2) is a pushout, and (2) is a pullback. It follows that $v, s \in \mathcal{M}$ and $r \in \mathcal{M}'_2$, since $\mathcal{M}$ is closed under pushouts and pullbacks and $\mathcal{M}'_2$ is closed under pullbacks along $\mathcal{M}$-morphisms.



We have to show that (1) and (2) are pushouts in **AGSIG-Alg**.

Using Lemmas 11.8 and 11.9, we can split up this diagram componentwise and obtain a pushout $(1)_s$ + $(2)_s$ and a pullback $(2)_s$ in **Sets** for each $s \in S_G \cup S_D$.

a) For $s \in S_D$, l is an isomorphism. In **Sets**, a commmutative square along an isomorphism is a pushout. Therefore $(1)_s$ is a pushout.
b) For $s \in S_G \backslash S_D$, we have the fact that l, s, v, w, and r are injective. Since **Sets** is an adhesive HLR category, we can conclude from Theorem 4.26 that $(1)_s$ is a pushout in **Sets**.

By Lemma 11.8, (1) is a pushout in **AGSIG-Alg**. By pushout decomposition, (2) is also a pushout.

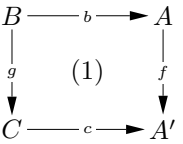
3. By the definition of $\mathcal{M}' = \mathcal{M}'_2$, we have the result that $f : A \to B \in \mathcal{M}'$, and $g : B \to C \in \mathcal{M}'$ implies $g \circ f \in \mathcal{M}'$ and $g \circ f, g \in \mathcal{M}'$ implies $f \in \mathcal{M}'$, since injectivity is preserved componentwise.

□

Lemma 11.17 (initial POs in AGSIG-Alg). *The categories* (**AGSIG-Alg**, $\mathcal{M}$)*, where AGSIG and* (**AGraphs$_{\mathbf{ATG}}$**, $\mathcal{M}$) *are well structured, have initial pushouts over general morphisms.*

Construction. *By Theorem 11.3, it suffices to show the construction and property for* (**AGSIG-Alg**, $\mathcal{M}$)*. For an explicit construction in* (**AGraphs$_{\mathbf{ATG}}$**, $\mathcal{M}$)*, see Definition 10.5 and Fact 10.7.*

Consider a well-structured AGSIG, which means that for all op $: s' \to s$ *in* OP_G *we have* $s' \notin S_D$*. Given* $f : A \to A'$*, the initial pushout over* f *is constructed by the following diagram:*



Here, the objects B, and C and the morphisms b, c, and g are defined as follows:

- $B = \cap\{B' \subseteq A \mid B'_{DSIG} = A_{DSIG}$ *and* $A^*_s \subseteq B'_s$ *for all* $s \in S_G \backslash S_D\}$ *with* $A^*_s = \{a \in A_s \mid \exists op : s' \to s \in OP_G \; \exists a' \in A'_{s'} \backslash f_{s'}(A_{s'}) : f_s(a) = op_{A'}(a')\} \cup \{a \in A_s \mid \exists a' \in A_s, a \neq a' : f_s(a) = f_s(a')\}$,
- $C_s = \begin{cases} A'_s & : \quad s \in S_D \\ A'_s \backslash f_s(A_s) \cup f_s(B_s) & : \quad s \in S_G \backslash S_D \end{cases}$, $op_C = \begin{cases} op_{A'} & : \quad op \in OP_D \\ op_{A'}|_{C_s} & : \quad op : s \to s' \in OP_G \end{cases}$,
- $b : B \to A$, $c : C \to A'$ *are inclusions with identical data type parts and hence* $b, c \in \mathcal{M}$. $g : B \to C$ *is defined by* $g_s = f_s|_{B_s}$ *for all* $s \in S_G \cup S_D$.

Initial pushouts are closed under double pushouts (see Lemma 6.5).

Proof. It can be shown that this construction is well defined and is indeed an initial pushout over f (see the proof in Section C.4). □