Multilevel Typed Graph Transformations

Uwe Wolter[1](✉), Fernando Macías[2], and Adrian Rutle[3]

[1] University of Bergen, Bergen, Norway
Uwe.Wolter@uib.no
[2] IMDEA Software Institute, Madrid, Spain
fernando.macias@imdea.org
[3] Western Norway University of Applied Sciences, Bergen, Norway
aru@hvl.no

Abstract. Multilevel modeling extends traditional modeling techniques with a potentially unlimited number of abstraction levels. Multilevel models can be formally represented by multilevel typed graphs whose manipulation and transformation are carried out by multilevel typed graph transformation rules. These rules are cospans of three graphs and two inclusion graph homomorphisms where the three graphs are multilevel typed over a common typing chain. In this paper, we show that typed graph transformations can be appropriately generalized to multilevel typed graph transformations improving preciseness, flexibility and reusability of transformation rules. We identify type compatibility conditions, for rules and their matches, formulated as equations and inequations, respectively, between composed partial typing morphisms. These conditions are crucial presuppositions for the application of a rule for a match—based on a pushout and a final pullback complement construction for the underlying graphs in the category **Graph**—to always provide a well-defined canonical result in the multilevel typed setting. Moreover, to formalize and analyze multilevel typing as well as to prove the necessary results, in a systematic way, we introduce the category **Chain** of typing chains and typing chain morphisms.

Keywords: Typing chain · Multilevel typed graph transformation · Pushout · Pullback complement

1 Introduction

Multilevel modeling (MLM) extends conventional techniques from the area of Model-Driven Engineering by providing model hierarchies with multiple levels of abstraction. The advantages of allowing multiple abstraction levels (e.g. reducing accidental complexity in software models and avoiding synthetic type-instance anti-patterns) and flexible typing (e.g. multiple typing, linguistic extension and deep instantiation), as well as the exact nature of the techniques used for MLM are well studied in the literature [1,4–6,8,10,17]. Our particular approach [19,20] to MLM facilitates the separation of concerns by allowing integration of different

F. Gadducci and T. Kehrer (Eds.): ICGT 2020, LNCS 12150, pp. 163–182, 2020.
https://doi.org/10.1007/978-3-030-51372-6_10

multilevel modeling hierarchies as separate aspects of the system to be modelled. In addition, we enhance reusability of concepts and their behaviour by allowing the definition of flexible transformation rules which are applicable to different hierarchies with a variable number of levels. In this paper, we present a revised and extended formalisation of these rules using graph theory and category theory.

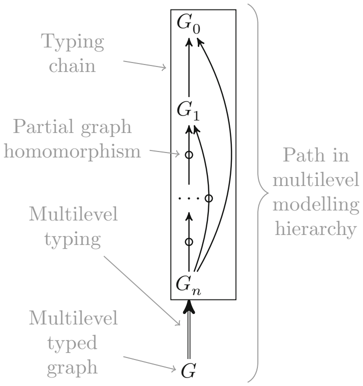

Fig. 1. MLM terminology

As models are usually represented abstractly as graphs, we outline in this paper the graph theoretic foundations of our approach to MLM using multilevel typed graphs, prior to introducing our formalisation of multilevel typed rule definition and application. Multilevel models are organized in hierarchies, where any graph G is *multilevel typed* over a *typing chain* of graphs (see Fig. 1). The typing relations of elements within each graph are represented via graph morphisms. Since we allow for deep instantiation [4–6,8], which refers to the ability to instantiate an element at any level below the level in which it is defined, these morphisms need to be *partial graph homomorphisms*. Moreover, more than one model can be typed by the same typing chain (or, conversely, models can be instantiated more than once), hence, all the *paths* that contain such typing relations constitute a full, tree-shaped *multilevel modelling hierarchy* (see Example 1). Finally, the topmost model G_0 in any hierarchy is fixed, and the typing relations of all models (and the elements inside them) must converge, directly or via a sequence of typing morphisms, into G_0. Therefore, the graph morphisms into G_0 are always total.

Multilevel typed graph transformation rules are cospans $L \xhookrightarrow{\lambda} I \xhookleftarrow{\rho} R$ of inclusion graph homomorphisms, with $I = L \cup R$, where the three graphs are multilevel typed over a common typing chain $\mathcal{MM}$. A match of the left-hand side L of the rule in a graph S, at the bottom of a certain hierarchy, multilevel typed over a typing chain $\mathcal{TG}$, is given by a graph homomorphism $\mu : L \to S$ and a flexible typing chain morphism from $\mathcal{MM}$ into $\mathcal{TG}$. The typing chain $\mathcal{MM}$ is local for the rules and is usually different from $\mathcal{TG}$ which is determined by the path from S to the top of the hierarchy (see Fig. 1).

$$\begin{tikzcd}
L \arrow[r, hook, "\lambda"] \arrow[d, "\mu"'] \arrow[dr, phantom, "PO"] & I \arrow[d, "\delta"] \arrow[dr, phantom, "FPBC"] & R \arrow[l, hook', "\rho"'] \arrow[d, "\nu"] \\
S \arrow[r, hook, "\varsigma"] & D & T \arrow[l, hook', "\theta"']
\end{tikzcd}$$

Fig. 2. Rule structure and basic constructions for rule application

To apply these rules we rely on an adaptation of the Sesqui pushout (Sq-PO) approach [7] to cospans. We construct first the pushout and then the final pullback complement (FPBC) of the underlying graph homomorphisms in the category **Graph** as shown in Fig. 2. Based on these traditional constructions we want to build, in a canonical way, type compatible multilevel typings of the result graphs D and T over the typing chain $\mathcal{TG}$. For this to work, we need quite reasonable type compatibility conditions for rules and relatively flexible conditions for matches, formulated as equations and inequations, resp., between composed partial typing morphisms.

We introduce typing chain morphisms, and the corresponding category **Chain** of typing chains and typing chain morphisms, to formalize flexible matching and application of multilevel typed rules. The composition of partial graph homomorphisms is based on pullbacks in the category **Graph**, thus type compatibility conditions can be equivalently expressed by commutativity and pullback conditions in **Graph**. Therefore, we formalize and analyze multilevel typing as well as describe constructions and prove the intended results, in a systematic way, within the category **Chain**. Especially, we show that the first step in a rule application can be described by a pushout in **Chain**. Moreover, the second step is described as a canonical construction in **Chain**, however, it is an open question whether this is a final pullback construction in **Chain** or not.

A preliminary version of typing chains are an implicit constituent of the concept "deep metamodeling stack" introduced in [22] to formalize concepts like parallel linguistic and ontological typing, linguistic extensions, deep instantiation and potencies in deep metamodeling. We revised this earlier version and further developed it to a concept of its own which serves as a foundation of our approach to multilevel typed model transformations in [20,26]. Compared to [20], we present in this paper a radically revised and extended theory of multilevel typed graph transformations. In particular, the theory is now more powerful, since we drop the condition that typing chain morphisms have to be closed (see Definition 5). Moreover, we detail the FPBC step which is missing in [20]. Due to space limitations, we will not present the background results concerning the equivalence between the practice of individual direct typing – which are used in applications and implementations – and our categorical reformulation of this practice by means of typing chains. These equivalence results as well as examples and proofs can be found in [26].

2 Typing Chains and Multilevel Typing of Graphs

Graph denotes the category of (directed multi-) graphs $G = (G^N, G^A, sc^G, tg^G)$ and graph homomorphisms $\phi = (\phi^N,\phi^A) : G \to H$ [12]. We will use the term **element** to refer to both nodes and arrows.

Multilevel typed graphs are graphs typed over a typing chain, i.e., a sequence $[G_n, G_{n-1}, \ldots, G_1, G_0]$ of graphs where the elements in any of the graphs G_i, with $n \geq i \geq 1$, are, on their part, multilevel typed over the sequence $[G_{i-1}, \ldots, G_1, G_0]$. Paths in our MLM hierarchies give rise to typing chains. The indexes i refer to the abstraction levels in a modeling hierarchy where 0 denotes the most abstract top level.

Following well-established approaches in the Graph Transformations field [12], we define typing by means of graph homomorphisms. This enables us to establish and develop our approach by reusing, variating, and extending the wide range of constructions and results achieved in that field. Moreover, this paves the way to generalize the present "paradigmatic" approach, where models are just graphs, to more sophisticated kinds of diagrammatic models, especially those that take advantage of diagrammatic constraints [22,23].

We allow typing to jump over abstraction levels, i.e., an element in graph G_i may have no type in G_{i-1} but only in one (or more) of the graphs $G_{i-2}, \ldots, G_1, G_0$. Two different elements in the same graph may have their types located in different graphs along the typing chain. To formalize this kind of flexible typing, we use partial graph homomorphisms that we introduced already in [22].

Definition 1. *A* ***partial graph homomorphism*** $\varphi : G \mathrel{-\!\!\circ\!\!\rightarrow} H$ *is given by a subgraph* $D(\varphi) \sqsubseteq G$*, called the* ***domain of definition*** *of* φ*, and a graph homomorphism* $\varphi : D(\varphi) \to H$*.*

Note that we use, in abuse of notation, the same name for both the partial and the corresponding total graph homomorphisms. To express transitivity of typing and later also compatibility of typing, we need as well the composition of partial graph homomorphisms as a partial order between partial graph homomorphisms.

Definition 2. *The* ***composition*** $\varphi; \psi : G \mathrel{-\!\!\circ\!\!\rightarrow} K$ *of two partial graph homomorphisms* $\varphi : G \mathrel{-\!\!\circ\!\!\rightarrow} H$ *and* $\psi : H \mathrel{-\!\!\circ\!\!\rightarrow} K$ *is defined as follows:*

- $D(\varphi; \psi) := \varphi^{-1}(D(\psi))$,
- $(\varphi; \psi)^N(\mathsf{e}) := \psi^N(\varphi^N(\mathsf{e}))$ *for all* $\mathsf{e} \in D(\varphi; \psi)^N$ *and* $(\varphi; \psi)^A(\mathsf{f}) := \psi^A(\varphi^A(\mathsf{f}))$ *for all* $\mathsf{f} \in D(\varphi; \psi)^A$*.*

More abstractly, the composition of two partial graph homomorphisms is defined by the following commutative diagram of total graph homomorphisms.

$$\begin{tikzcd}
& & D(\varphi;\psi) \arrow[dl, hook', "\sqsubseteq"'] \arrow[dr, "\varphi_{|\psi}"] \arrow[ddrr, bend left, "\varphi;\psi"] & & \\
& D(\varphi) \arrow[dl, hook', "\sqsubseteq"'] \arrow[dr, "\varphi"] & PB & D(\psi) \arrow[dl, hook', "\sqsubseteq"'] \arrow[dr, "\psi"] & \\
G & & H & & K
\end{tikzcd}$$

Note that $D(\varphi; \psi) = D(\varphi)$ *if* ψ *is total, i.e.,* $H = D(\psi)$*.*

Definition 3. *For any two partial graph homomorphisms* $\varphi, \phi : G \mathrel{-\!\!\circ\!\!\rightarrow} H$ *we have* $\varphi \preceq \phi$ *iff* $D(\varphi) \sqsubseteq D(\phi)$ *and* φ*,* ϕ *coincide on* $D(\varphi)$*.*

Now, we can define typing chains as a foundation for our investigation of multilevel typed graph transformations in the rest of the paper.

Definition 4. *A* ***typing chain*** $\mathcal{G} = (\overline{G}, n, \tau^{\mathcal{G}})$ *is given by a natural number* n*, a sequence* $\overline{G} = [G_n, G_{n-1}, \ldots, G_1, G_0]$ *of graphs of length* $n + 1$ *and a family* $\tau^{\mathcal{G}} = (\tau^{\mathcal{G}}_{j,i} : G_j \mathrel{-\!\!\circ\!\!\rightarrow} G_i \mid n \geq j > i \geq 0)$ *of partial graph homomorphisms, called* ***typing morphisms****, satisfying the following properties:*

- ***Total:*** *All the morphisms* $\tau^{\mathcal{G}}_{j,0} : G_j \to G_0$ *with* $n \geq j \geq 1$ *are total.*

- ***Transitive:*** *For all $n \geq k > j > i \geq 0$ we have $\tau^{\mathcal{G}}_{k,j};\tau^{\mathcal{G}}_{j,i} \preceq \tau^{\mathcal{G}}_{k,i}$.*
- ***Connex:*** *For all $n \geq k > j > i \geq 0$ we have $D(\tau^{\mathcal{G}}_{k,j}) \cap D(\tau^{\mathcal{G}}_{k,i}) \sqsubseteq D(\tau^{\mathcal{G}}_{k,j};\tau^{\mathcal{G}}_{j,i}) = (\tau^{\mathcal{G}}_{k,j})^{-1}(D(\tau^{\mathcal{G}}_{j,i}))$, moreover, $\tau^{\mathcal{G}}_{k,j};\tau^{\mathcal{G}}_{j,i}$ and $\tau^{\mathcal{G}}_{k,i}$ coincide on $D(\tau^{\mathcal{G}}_{k,j}) \cap D(\tau^{\mathcal{G}}_{k,i})$.*

Due to Definitions 2 and 3, transitivity and connexity together mean that $D(\tau^{\mathcal{G}}_{k,j}) \cap D(\tau^{\mathcal{G}}_{k,i}) = D(\tau^{\mathcal{G}}_{k,j};\tau^{\mathcal{G}}_{j,i})$, i.e., we do have a (unique) total graph homomorphism $\tau^{\mathcal{G}}_{k,j|i} : D(\tau^{\mathcal{G}}_{k,j}) \cap D(\tau^{\mathcal{G}}_{k,i}) \to D(\tau^{\mathcal{G}}_{j,i})$ and the following commutative diagram of total graph homomorphisms

$$\begin{tikzcd}
 & D(\tau^{\mathcal{G}}_{k,i}) \arrow[ld, hook', bend right, "\sqsubseteq"'] \arrow[r, "\tau^{\mathcal{G}}_{k,i}"] & G_i \\
G_k \arrow[r, phantom, "PB"] & D(\tau^{\mathcal{G}}_{k,j}) \cap D(\tau^{\mathcal{G}}_{k,i}) \arrow[u, hook, "\sqsubseteq"'] \arrow[d, hook', "\sqsubseteq"] \arrow[r, "\tau^{\mathcal{G}}_{k,j|i}"] \arrow[ru, phantom, "="] & D(\tau^{\mathcal{G}}_{j,i}) \arrow[u, "\tau^{\mathcal{G}}_{j,i}"'] \arrow[d, hook, "\sqsubseteq"] \arrow[ld, phantom, "PB"] \\
 & D(\tau^{\mathcal{G}}_{k,j}) \arrow[lu, hook, bend left, "\sqsubseteq"] \arrow[r, "\tau^{\mathcal{G}}_{k,j}"] & G_j
\end{tikzcd}$$

Remark 1. For any element $\mathtt{e}$ in any graph G_i in a typing chain, with $i > 0$, there exists a unique index $m_{\mathtt{e}}$, with $i > m_{\mathtt{e}} \geq 0$, such that $\mathtt{e}$ is in the domain of the typing morphism $\tau^{\mathcal{G}}_{i,m_{\mathtt{e}}}$ but not in the domain of any typing morphism $\tau^{\mathcal{G}}_{i,j}$ with $i > j > m_{\mathtt{e}}$. We call $\tau^{\mathcal{G}}_{i,m_{\mathtt{e}}}(\mathtt{e})$ the **direct type** of $\mathtt{e}$. For any other index k, with $m_{\mathtt{e}} > k \geq 0$, we call $\tau^{\mathcal{G}}_{i,k}(\mathtt{e})$, if it is defined, a **transitive type** of $\mathtt{e}$.

Example 1. Figure 3 depicts the typing morphisms between the graphs in a simplified sample hierarchy. The direct types for nodes and arrows are indicated with blue and cursive labels, respectively. All typing morphisms in the simple typing chain $\mathcal{TG}$, determined by the sequence [`hammer_plant`, `generic_plant`, `Ecore`] of graphs, are total except the one from `hammer_plant` to `generic_plant`, since the direct type of `has` is located in `Ecore`. We have chosen Ecore as the top-most graph since it provides implementation support through the Eclipse Modeling Framework [24]. This enables our approach to MLM to exploit the best from fixed-level and multi-level concepts [18]. □

To describe later the flexible matching of multilevel typed rules and the result of rule applications, we need a corresponding flexible notion of morphisms between typing chains.

Definition 5. *A **typing chain morphism** $(\phi, f) : \mathcal{G} \to \mathcal{H}$ between two typing chains $\mathcal{G} = (\overline{G}, n, \tau^{\mathcal{G}})$ and $\mathcal{H} = (\overline{H}, m, \tau^{\mathcal{H}})$ with $n \leq m$ is given by*

- *a function $f : [n] \to [m]$, where $[n] = \{0, 1, 2, \ldots, n\}$, such that (1) $f(0) = 0$ and (2) $j > i$ implies $f(j) - f(i) \geq j - i$ for all $i, j \in [n]$, and*
- *a family of total graph homomorphisms $\phi = (\phi_i : G_i \to H_{f(i)} \mid i \in [n])$ such that*

$$\tau^{\mathcal{G}}_{j,i};\phi_i \preceq \phi_j;\tau^{\mathcal{H}}_{f(j),f(i)} \quad \text{for all } n \geq j > i \geq 0, \tag{1}$$

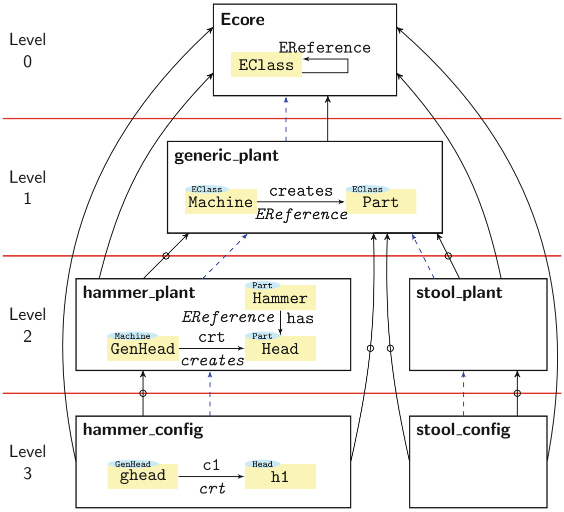

Fig. 3. Multilevel modeling hierarchy with typing morphisms

> *i.e., due to Definitions 2 and 3, we assume for any $n \geq j > i \geq 0$ the existence of a total graph homomorphism $\phi_{j|i}$ that makes the diagram of total graph homomorphisms displayed in Fig. 4 commutative.*

*A typing chain morphism $(\phi, f) : \mathcal{G} \to \mathcal{H}$ is **closed** iff $\tau^{\mathcal{G}}_{j,i}; \phi_i = \phi_j; \tau^{\mathcal{H}}_{f(j),f(i)}$ for all $n \geq j > i \geq 0$, i.e., the right lower square in Fig. 4 is a pullback.*

$$\begin{tikzcd}
G_i \arrow[r, "\phi_i"] & H_{f(i)} \\
G_j \arrow[u, "\tau^G_{j,i}", "\circ" marking] \arrow[r, "\phi_j"] \arrow[ur, phantom, "\preceq"] & H_{f(j)} \arrow[u, "\tau^H_{f(j),f(i)}"', "\circ" marking]
\end{tikzcd}
\qquad
\begin{tikzcd}
G_i \arrow[r, "\phi_i"] & H_{f(i)} \\
D(\tau^G_{j,i}) \arrow[u, "\tau^G_{j,i}"] \arrow[r, "\phi_{j|i}"] \arrow[d, hook, "\sqsubseteq"'] \arrow[ur, phantom, "="] \arrow[dr, phantom, "= (PB)"] & D(\tau^H_{f(j),f(i)}) \arrow[u, "\tau^H_{f(j),f(i)}"'] \arrow[d, hook', "\sqsubseteq"] \\
G_j \arrow[r, "\phi_j"] & H_{f(j)}
\end{tikzcd}$$

Fig. 4. Establishing a morphism between two typing chains, level-wise

Typing morphisms are composed by the composition of commutative squares.

Definition 6. *The **composition** $(\phi, f); (\psi, g) : \mathcal{G} \to \mathcal{K}$ of two typing chain morphisms $(\phi, f) : \mathcal{G} \to \mathcal{H}$, $(\psi, g) : \mathcal{H} \to \mathcal{K}$ between typing chains $\mathcal{G} = (\overline{G}, n, \tau^{\mathcal{G}})$, $\mathcal{H} = (\overline{H}, m, \tau^{\mathcal{H}})$, $\mathcal{K} = (\overline{K}, l, \tau^{\mathcal{K}})$ with $n \leq m \leq l$ is defined by $(\phi, f); (\psi, g) := (\phi; \psi_{\downarrow f}, f; g)$, where $\psi_{\downarrow f} := (\psi_{f(i)} : H_{f(i)} \to K_{g(f(i))} \mid i \in [n])$, and thus $\phi; \psi_{\downarrow f} := (\phi_i; \psi_{f(i)} : G_i \to K_{g(f(i))} \mid i \in [n])$.*

Chain denotes the category of typing chains and typing chain morphisms.

A natural way to define multilevel typing of a graph H over a typing chain $\mathcal{G}$ would be a family $\sigma = (\sigma_i : H \mathrel{-\!\!\circ\!\!\rightarrow} G_i \mid n \geq i \geq 0)$ of partial graph homomorphisms satisfying certain properties. However, as shown in [26], those families are not appropriate to state adequate type compatibility requirements for rules and matches and to construct the results of rule applications. Therefore, we employ the sequence of the domains of definition of the σ_i's as a typing chain and describe multilevel typing by means of typing chain morphisms. The following Lemma describes how any sequence of subgraphs gives rise to a typing chain.

Lemma 1. *Any sequence $\overline{H} = [H_n, H_{n-1}, \ldots, H_1, H_0]$ of subgraphs of a graph H, with $H_0 = H$, can be extended to a typing chain $\mathcal{H} = (\overline{H}, n, \tau^{\mathcal{H}})$ where for all $n \geq j > i \geq 0$ the corresponding* ***typing morphism*** *$\tau^{\mathcal{H}}_{j,i} : H_j \mathrel{-\!\!\circ\!\!\rightarrow} H_i$ is given by $D(\tau^{\mathcal{H}}_{j,i}) := H_j \cap H_i$ and the span of total inclusion graph homomorphisms*

$$\begin{tikzcd} H_j & D(\tau^{\mathcal{H}}_{j,i}) = H_j \cap H_i \arrow[l, hook', "\sqsubseteq"'] \arrow[r, hook, "\tau^{\mathcal{H}}_{j,i}"] & H_i \end{tikzcd} .$$

We call the typing chain $\mathcal{H} = (\overline{H}, n, \tau^{\mathcal{H}})$ an ***inclusion chain on*** *H.*

A **multilevel typing of a graph** H over a typing chain $\mathcal{G} = (\overline{G}, n, \tau^{\mathcal{G}})$ is given by an inclusion chain $\mathcal{H} = (\overline{H}, n, \tau^{\mathcal{H}})$ on H (of the same length as $\mathcal{G}$) and a typing chain morphism $(\sigma^{\mathcal{H}}, id_{[n]}) : \mathcal{H} \to \mathcal{G}$.

3 Multilevel Typed Graph Transformations

Underlying Graph Transformation. To meet the characteristics of our application areas [19–21] we work with cospans $L \xhookrightarrow{\lambda} I \xhookleftarrow{\rho} R$ of inclusion graph homomorphisms, where $I = L \cup R$, as the **underlying graph transformation rule** of a multilevel typed rule. To apply such a rule [7,12,13], we have to find a match $\mu : L \to S$ of L in a graph S at the bottom-most level of an MLM hierarchy. To describe the effect of a rule application, we adapt the Sq-PO approach [7] to our cospan-rules: First, we construct a pushout and, second, a final pullback complement (FPBC) to create the graphs D and T, resp. (see Fig. 2). The details behind choosing cospan rules and Sq-PO, as opposed to span rules and double-pushout (DPO), are out of the scope of this paper. In short, however: (i) cospan rules are more suitable from an implementation point-of-view since they allow for first adding new elements then deleting (some of the) old elements [13], and (ii) having both old and new elements in I allows us to introduce constraints on new elements depending on old constraints involving elements to be deleted [23]. Moreover, we apply the rules using our variant of Sq-PO [7,13] since (i) the pushout complement in DPO, even if it exists, may not be unique, in contrast the FPBC, if it exists, is always unique (up to isomorphism), (ii) FPBC allows faithful deletion in unknown context, i.e., dangling edges may be deleted by applying the rules, however, the co-match ν is always total, i.e., if the match μ identifies elements to be removed with elements to be preserved, the FPBC will not exist and the application will not be allowed.

Multilevel Typed Rule. We augment the cospan rule to a **multilevel typed rule** by chosing a typing chain $\mathcal{MM} = (\overline{MM}, n, \tau^{\mathcal{MM}})$, the typing chain of the rule, together with **coherent** multilevel typings over $\mathcal{MM}$ of L and R, respectively. That is, we choose an inclusion chain $\mathcal{L} = (\overline{L}, n, \tau^{\mathcal{L}})$ on L, an inclusion chain $\mathcal{R} = (\overline{R}, n, \tau^{\mathcal{R}})$ on R and typing chain morphisms $(\sigma^{\mathcal{L}}, id_{[n]}) : \mathcal{L} \to \mathcal{MM}$ with $\sigma^{\mathcal{L}} = (\sigma_i^{\mathcal{L}} : L_i \to MM_i \mid i \in [n])$, $(\sigma^{\mathcal{R}}, id_{[n]}) : \mathcal{R} \to \mathcal{MM}$ with $\sigma^{\mathcal{R}} = (\sigma_i^{\mathcal{R}} : R_i \to MM_i \mid i \in [n])$ (see Fig. 5), such that $L_i \cap R = L \cap R_i = L_i \cap R_i$ and, moreover, $\sigma_i^{\mathcal{L}}$ and $\sigma_i^{\mathcal{R}}$ coincide on the intersection $L_i \cap R_i$ for all $i \in [n]$.

$$\begin{tikzcd}
& \mathcal{MM} & \\
\mathcal{L} \arrow[ur, "{(\sigma^{\mathcal{L}}, id_{[n]})}"] \arrow[r, dotted, "{(\lambda, id_{[n]})}"] & \mathcal{I} \arrow[u, dotted, "{(\sigma^{\mathcal{I}}, id_{[n]})}"'] & \mathcal{R} \arrow[ul, "{(\sigma^{\mathcal{R}}, id_{[n]})}"'] \arrow[l, dotted, "{(\rho, id_{[n]})}"']
\end{tikzcd}$$

Fig. 5. Rule morphisms and their type compatibility

The inclusion chain $\mathcal{I} = (\overline{I}, n, \tau^{\mathcal{I}})$ on the union (pushout) $I = L \cup R$ is simply constructed by level-wise unions (pushouts): $I_i := L_i \cup R_i$ for all $i \in [n]$; thus, we have $I_0 = I$. Since **Graph** is an adhesive category [12], the construction of $\mathcal{I}$ by pushouts and the coherence condition ensure that we get for any $i \in [n]$ two pullbacks as shown in Fig. 6. The existence of these pullbacks implies, according to the following Lemma, that we can reconstruct the inclusion chains $\mathcal{L}$ and $\mathcal{R}$, respectively, as reducts of the inclusion chain $\mathcal{I}$.

Lemma 2. *Let be given two inclusion chains $\mathcal{G} = (\overline{G}, n, \tau^{\mathcal{G}})$ and $\mathcal{H} = (\overline{H}, m, \tau^{\mathcal{H}})$ with $n \leq m$ and a function $f : [n] \to [m]$ such that $f(0) = 0$ and $j > i$ implies $f(j) - f(i) \geq j - i$ for all $i, j \in [n]$. For any family $\phi = (\phi_i : G_i \to H_{f(i)} \mid i \in [n])$ of graph homomorphisms the following two requirements are equivalent:*

1. *For all $n \geq j > 0$ the left-hand square in Fig. 7 is a pullback.*
2. *The pair (ϕ, f) constitutes a closed typing chain morphism $(\phi, f) : \mathcal{G} \to \mathcal{H}$ where for all $n \geq j > i \geq 0$ the right-hand diagram in Fig. 7 consists of two pullbacks.*

Given a closed typing chain morphism $(\phi, f) : \mathcal{G} \to \mathcal{H}$ between inclusion chains, as described in Lemma 2, we call $\mathcal{G}$ the **reduct of $\mathcal{H}$ along $\phi_0 : G_0 \to H_0$ and $f : [n] \to [m]$** while $(\phi, f) : \mathcal{G} \to \mathcal{H}$ is called a **reduct morphism**. Note that the composition of two reduct morphisms is a reduct morphism as well.

Lemma 2 ensures that the families $(\lambda_i : L_i \to I_i \mid i \in [n])$ and $(\rho_i : R_i \to I_i \mid i \in [n])$ of inclusion graph homomorphisms establish reduct morphisms $(\lambda, id_{[n]}) : \mathcal{L} \to \mathcal{I}$ and $(\rho, id_{[n]}) : \mathcal{R} \to \mathcal{I}$, resp., as shown in Fig. 5.

$$\begin{tikzcd}
L_0 = L \arrow[r, hook, "\lambda_0 := \lambda"] & I_0 = L_0 \cup R_0 & R_0 = R \arrow[l, hook', "\rho_0 := \rho"'] \\
L_i \arrow[u, hook, "\tau^{\mathcal{L}}_{i,0}"'] \arrow[ur, phantom, "PB"] \arrow[r, hook, "\lambda_i"] \arrow[dr, "\sigma^{\mathcal{L}}_i"'] \arrow[dr, phantom, bend left, "="] & I_i = L_i \cup R_i \arrow[u, hook, "\tau^{\mathcal{I}}_{i,0}"'] \arrow[ur, phantom, "PB"] \arrow[d, "\sigma^{\mathcal{I}}_i"] & R_i \arrow[l, hook', "\rho_i"'] \arrow[u, hook, "\tau^{\mathcal{R}}_{i,0}"'] \arrow[dl, "\sigma^{\mathcal{R}}_i"] \arrow[dl, phantom, bend right, "="] \\
& MM_i &
\end{tikzcd}$$

Fig. 6. Type compatibility of rule morphisms level-wise

$$\begin{tikzcd}
G_0 \arrow[r, "\phi_0"] & H_0 \\
G_j \arrow[u, hook, "\tau^{\mathcal{G}}_{j,0}"] \arrow[ur, phantom, "PB"] \arrow[r, "\phi_j"] & H_{f(j)} \arrow[u, hook, "\tau^{\mathcal{H}}_{f(j),0}"']
\end{tikzcd}
\qquad
\begin{tikzcd}
G_i \arrow[r, "\phi_i"] & H_{f(i]} \\
G_j \cap G_i \arrow[u, hook, "\tau^{\mathcal{G}}_{j,i}"] \arrow[ur, phantom, "PB"] \arrow[r, "\phi_{j|i}"] \arrow[d, hook, "\sqsubseteq"'] \arrow[dr, phantom, "PB"] & H_{f(j)} \cap H_{f(i)} \arrow[u, hook, "\tau^{\mathcal{H}}_{f(j),f(i)}"'] \arrow[d, hook, "\sqsubseteq"] \\
G_j \arrow[r, "\phi_j"] & H_{f(j)}
\end{tikzcd}$$

Fig. 7. Reduct of inclusion chains

Finally, we have to construct a typing chain morphism $(\sigma^{\mathcal{I}}, id_{[n]}) : \mathcal{I} \to \mathcal{MM}$ making the diagram in Fig. 5 commute: For all $i \in [n]$, we constructed the union (pushout) $I_i := L_i \cup R_i$. Moreover, $\sigma^{\mathcal{L}}_i$ and $\sigma^{\mathcal{R}}_i$ coincide on $L_i \cap R_i$, by coherence assumption, thus we get a unique $\sigma^{\mathcal{I}}_i : I_i \to MM_i$ such that (see Fig. 6)

$$\sigma^{\mathcal{L}}_i = \lambda_i ; \sigma^{\mathcal{I}}_i \quad \text{and} \quad \sigma^{\mathcal{R}}_i = \rho_i ; \sigma^{\mathcal{I}}_i \tag{2}$$

Since **Graph** is adhesive, Lemma 2 ensures that the family $\sigma^{\mathcal{I}} = (\sigma^{\mathcal{I}}_i : I_i \to MM_i \mid i \in [n])$ of graph homomorphisms establishes indeed a typing chain morphism $(\sigma^{\mathcal{I}}, id_{[n]}) : \mathcal{I} \to \mathcal{MM}$ while the Eq. 2 ensure that the diagram in Fig. 5 commutes indeed.

Example 2. Figure 8 shows a multilevel typed rule *CreatePart* from a case study [20]. This rule can be used to specify the behaviour of machines that create parts, by matching an existing type of machine that generates a certain type of parts, and in the instance at the bottom, generating such a part. META defines a typing chain $\mathcal{MM}$ of depth 3. It declares the graph ($\texttt{M1} \xrightarrow{\texttt{cr}} \texttt{P1}$) that becomes MM_2. The declaration of the direct types `Machine`, `creates`, `Part` for the elements in MM_2 declares, implicitly, a graph $MM_1 :=$ ($\texttt{Machine} \xrightarrow{\texttt{creates}} \texttt{Part}$) that is in turn, implicitly, typed over $MM_0 := \mathtt{ECore}$. All the morphisms in $\tau^{\mathcal{MM}}$ are total and uniquely determined thus we have, especially, $\tau^{\mathcal{MM}}_{2,0} = \tau^{\mathcal{MM}}_{2,1};\tau^{\mathcal{MM}}_{1,0}$.

FROM and TO declare as well the left-hand side $L := (\,\mathtt{m1}\,)$ and the right-hand-side $R := (\,\mathtt{m1} \xrightarrow{\mathtt{c}} \mathtt{p1}\,)$, resp., of the rule and the direct types of the elements in L and R. These direct types are all located in MM_2 thus $L_2 = L$ and $R_2 = R$ where the direct types define nothing but the typing morphisms $\sigma^{\mathcal{L}}_2 : L_2 \to MM_2$ and $\sigma^{\mathcal{R}}_2 : R_2 \to MM_2$, resp. The other typing morphisms are obtained by "transitive closure", i.e., $\sigma^{\mathcal{L}}_1 := \sigma^{\mathcal{L}}_2;\tau^{\mathcal{MM}}_{2,1}$, $\sigma^{\mathcal{L}}_0 := \sigma^{\mathcal{L}}_2;\tau^{\mathcal{MM}}_{2,0}$ and $\sigma^{\mathcal{R}}_1 := \sigma^{\mathcal{R}}_2;\tau^{\mathcal{MM}}_{2,1}$, $\sigma^{\mathcal{R}}_0 := \sigma^{\mathcal{R}}_2;\tau^{\mathcal{MM}}_{2,0}$, thus we have $L = L_0 = L_1 = L_2$ and $R = R_0 = R_1 = R_2$.

For the "plain variant" of the rule *CreatePart* (in Fig. 15), $\mathcal{MM}$ consists only of the graphs $MM_1 = (\,\mathtt{M1} \xrightarrow{\mathtt{cr}} \mathtt{P1}\,)$, $MM_0 = \mathtt{ECore}$ and the trivial $\tau^{\mathcal{MM}}_{1,0}$.

Multilevel Typed Match. In the multilevel typed setting all the graphs S, D, T are multilevel typed over a common typing chain $\mathcal{TG} = (\overline{TG}, m, \tau^{\mathcal{TG}})$, with $n \leq m$, that is determined by the path from S to the top of the current MLM hierarchy (see Fig. 1).

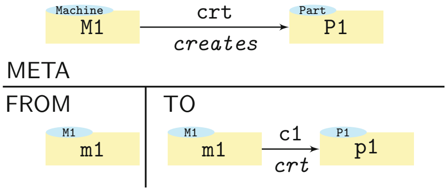

Fig. 8. *CreatePart*: a sample rule

A **match** of the multilevel typed rule into a graph S with a given multilevel typing over $\mathcal{TG}$, i.e., an inclusion chain $\mathcal{S} = (\overline{S}, m, \tau^{\mathcal{S}})$ with $S_0 = S$ and a typing chain morphism $(\sigma^{\mathcal{S}}, id_{[m]}) : \mathcal{S} \to \mathcal{TG}$, is given by a graph homomorphism $\mu : L \to S$ and a typing chain morphism $(\beta, f) : \mathcal{MM} \to \mathcal{TG}$ such that the following two conditions are satisfied:

- **Reduct:** $\mathcal{L}$ is the reduct of $\mathcal{S}$ along $\mu : L \to S$ and $f : [n] \to [m]$, i.e., $\mu_0 := \mu : L_0 = L \longrightarrow S_0 = S$ extends uniquely (by pullbacks) to a reduction morphism $(\mu, f) : \mathcal{L} \to \mathcal{S}$ with $\mu = (\mu_i : L_i \to S_{f(i)} \mid i \in [n])$ (see Fig. 9).
- **Type compatibility:** $(\sigma^{\mathcal{L}}, id_{[n]});(\beta, f) = (\mu, f);(\sigma^{\mathcal{S}}, id_{[m]})$, i.e., we require

$$\sigma^{\mathcal{L}}_i;\beta_i = \mu_i;\sigma^{\mathcal{S}}_{f(i)} \text{ for all } n \geq i > 0. \tag{3}$$

$$\begin{tikzcd}
L_0 = L \arrow[r, "\mu_0 := \mu"] & S_0 = S \\
L_i \arrow[u, hook, "\tau^{\mathcal{L}}_{i,0}"] \arrow[r, "\mu_i"] \arrow[d, "\sigma^{\mathcal{L}}_i"'] \arrow[ur, phantom, "PB"] & S_{f(i)} \arrow[u, hook, "\tau^{\mathcal{S}}_{f(i),f(0)}"'] \arrow[d, "\sigma^{\mathcal{S}}_{f(i)}"] \\
MM_i \arrow[r, "\beta_i"] \arrow[ur, phantom, "="] & TG_{f(i)}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{MM} \arrow[r, "{(\beta, f)}"] & \mathcal{TG} \\
\mathcal{L} \arrow[u, "{(\sigma^{\mathcal{L}}, id_{[n]})}"] \arrow[r, "{(\mu, f)}"] \arrow[ur, phantom, "="] & \mathcal{S} \arrow[u, "{(\sigma^{\mathcal{S}}, id_{[m]})}"']
\end{tikzcd}$$

Fig. 9. Conditions for multilevel typEd Match

Application of a Multilevel Typed Rule – Objectives. The basic idea is to construct for a given application of a graph transformation rule, as shown in Fig. 2, a unique type compatible multilevel typing of the result graphs D and T. The parameters of this construction are typing chains $\mathcal{MM}$, $\mathcal{TG}$; a coherent multilevel typing of the graph transformation rule over $\mathcal{MM}$; a multilevel typing of the graph S over $\mathcal{TG}$ and a typing chain morphism $(\beta, f) : \mathcal{MM} \to \mathcal{TG}$ extending the given match $\mu : L \to S$ of graphs to a multilevel typed match satisfying the two respective conditions for multilevel typed matches.

Example 3 (Multilevel Typed Match). To achieve precision in rule application the elements `Machine`, `creates`, `Part` in the original rule *CreatePart* are constants required to match syntactically with elements in the hierarchy. In such a way, MM_1 = ($\texttt{Machine} \xrightarrow{\texttt{creates}} \texttt{Part}$)has to match with `generic_plant` while MM_2 = ($\texttt{M1} \xrightarrow{\texttt{cr}} \texttt{P1}$) could match with `hammer_plant` or `stool_plant`. We will observe later that for the plain version of the rule *CreatePart* in Fig. 15 we could match MM_1 = ($\texttt{M1} \xrightarrow{\texttt{cr}} \texttt{P1}$) either with TG_2 = `hammer_plant` or TG_1 = `generic_plant` in the hierarchy in Fig. 3, where the second match would lead to undesired results (see Example 4).

Pushout step. As shown later, the pushout of the span $S \xleftarrow{\mu} L \xhookrightarrow{\lambda} I$ in **Graph** extends, in a canonical way, to a pushout of the span

$$\mathcal{S} \xleftarrow{(\mu, f)} \mathcal{L} \xhookrightarrow{(\lambda, id_{[n]})} \mathcal{I}$$

of reduct morphisms in **Chain** such that the result typing chain $\mathcal{D} = (\overline{D}, m, \tau^{\mathcal{D}})$ is an inclusion chain and the typing chain morphisms $(\varsigma, id_{[m]}) : \mathcal{S} \hookrightarrow \mathcal{D}$ and $(\delta, f) : \mathcal{I} \to \mathcal{D}$ become reduct morphisms (see the bottom in Fig. 10).

$$\begin{tikzcd}
& \mathcal{MM} \arrow[rr, "{(\beta, f)}"] & & \mathcal{TG} \\
\mathcal{L} \arrow[ur, "{(\sigma^{\mathcal{L}}, id_{[n]})}"] \arrow[r, hook, "{(\lambda, id_{[n]})}"] \arrow[dr, "{(\mu, f)}"'] & \mathcal{I} \arrow[u, "{(\sigma^{\mathcal{I}}, id_{[n]})}"] \arrow[drr, "{(\delta, f)}"] & & \\
& \mathcal{S} \arrow[uurr, "{(\sigma^{\mathcal{S}}, id_{[m]})}"] \arrow[rr, hook, "{(\varsigma, id_{[m]})}"] & & \mathcal{D} \arrow[uu, dotted, "{(\sigma^{\mathcal{D}}, id_{[m]})}"']
\end{tikzcd}$$

Fig. 10. Pushout step

We get also a type compatible typing chain morphism from $\mathcal{D}$ into $\mathcal{TG}$: The back triangle in Fig. 10 commutes due to the type compatibility of the rule (see Fig. 5). The roof square commutes since the match is type compatible (see Fig. 9). This gives us $(\mu, f); (\sigma^{\mathcal{S}}, id_{[m]}) = (\lambda, id_{[n]}); (\sigma^{\mathcal{I}}, id_{[n]}); (\beta, f)$, thus the universal property of the pushout bottom square provides a unique chain morphism $(\sigma^{\mathcal{D}}, id_{[m]}) : \mathcal{D} \to \mathcal{TG}$ such that both type compatibility conditions $(\varsigma, id_{[m]}); (\sigma^{\mathcal{D}}, id_{[m]}) = (\sigma^{\mathcal{S}}, id_{[m]})$ and $(\delta, f); (\sigma^{\mathcal{D}}, id_{[m]}) = (\sigma^{\mathcal{I}}, id_{[n]}); (\beta, f)$ are satisfied.

Pullback Complement Step. As shown later, the final pullback complement $D \xleftarrow{\theta} T \xhookleftarrow{\nu} R$ in **Graph** extends, in a canonical way, to a sequence of reduct morphisms $\mathcal{D} \xhookleftarrow{(\theta, id_{[n]})} \mathcal{T} \xleftarrow{(\nu, f)} \mathcal{R}$ in **Chain** such that the bottom square in Fig. 11 commutes.

$$\begin{tikzcd}
\mathcal{MM} \arrow[rr, "{(\beta, f)}"] & & \mathcal{TG} & & \\
\mathcal{I} \arrow[u, "{(\sigma^{\mathcal{I}}, id_{[n]})}"] \arrow[drr, "{(\delta, f)}"'] & & & \mathcal{R} \arrow[lll, "{(\rho, id_{[n]})}"] \arrow[ulll, "{(\sigma^{\mathcal{R}}, id_{[n]})}"'] \arrow[dr, dotted, "{(\nu, f)}"] & \\
& & \mathcal{D} \arrow[uu, "{(\sigma^{\mathcal{D}}, id_{[m]})}"'] & & \mathcal{T} \arrow[ll, dotted, "{(\theta, id_{[m]})}"] \arrow[uull, dotted, "{(\sigma^{\mathcal{T}}, id_{[m]})}"']
\end{tikzcd}$$

Fig. 11. Pullback complement step

Pushout of Reduct Morphisms – Two Steps. We discuss the intended pushout of the span

$$\mathcal{S} \xleftarrow{(\mu, f)} \mathcal{L} \xhookrightarrow{(\lambda, id_{[n]})} \mathcal{I}$$

of reduct morphisms in **Chain**. The reduct morphism $(\lambda, id_{[n]})$ is surjective w.r.t. levels, thus the pushout inclusion chain $\mathcal{D}$ should have the same length as $\mathcal{S}$. The rule provides, however, only information how to extend the subgraphs of $S_0 = S$ at the levels $f([n]) \subseteq [m]$. For the subgraphs in $\mathcal{S}$ at levels in $[m] \setminus f([n])$ the rule does not impose anything thus we let the subgraphs at those levels untouched. In terms of typing chain morphisms, this means that we factorize the reduct morphism (μ, f) into two reduct morphisms and that we will construct the resulting inclusion chain $\mathcal{D}$ in two pushout steps (see Fig. 12) where $\mathcal{S}_{\downarrow f} := (\overline{S}_{\downarrow f}, n, \tau^{\mathcal{S}}_{\downarrow f})$ with $\overline{S}_{\downarrow f} := [S_{f(n)}, S_{f(n-1)}, \ldots, S_{f(1)}, S_{f(0)=0}]$ and $\tau^{\mathcal{S}}_{\downarrow f} := (\tau^{\mathcal{S}}_{f(j),f(i)} : S_{f(j)} \mathrel{\circ\!\!\!\longrightarrow} S_{f(i)} \mid n \geq j > i \geq 0)$ Note, that $\overline{S}_{\downarrow f} := [S_{f(n)}, \ldots, S_{f(0)}]$ is just a shorthand for the defining statement: $(\overline{S}_{\downarrow f})_i := S_{f(i)}$ for all $n \geq i \geq 0$.

$$\begin{tikzcd}
\mathcal{L} \arrow[r, "(\mu, id_{[n]})"] \arrow[rr, bend left, "(\mu, f)"] \arrow[d, hook, "(\lambda, id_{[n]})"'] \arrow[dr, phantom, "(1)"] & \mathcal{S}_{\downarrow f} \arrow[r, "(\overline{id}^{\mathcal{S}}_{\downarrow f}, f)"] \arrow[d, hook, "(\varsigma_{\downarrow f}, id_{[n]})"] \arrow[dr, phantom, "(2)"] & \mathcal{S} \arrow[d, hook, "(\varsigma, id_{[m]})"] \\
\mathcal{I} \arrow[r, "(\delta, id_{[n]})"'] \arrow[rr, bend right, "(\delta, f)"'] & \mathcal{D}_{\downarrow f} \arrow[r, "(\overline{id}^{\mathcal{D}}_{\downarrow f}, f)"'] & \mathcal{D}
\end{tikzcd}$$

Fig. 12. Two pushout steps to construct the inclusion chain $\mathcal{D}$

The reduct morphism $(\overline{id}^{\mathcal{S}}_{\downarrow f}, f) : \mathcal{S}_{\downarrow f} \to \mathcal{S}$ is a level-wise identity and just embeds an inclusion chain of length $n+1$ into an inclusion chain of length $m+1$, i.e., $\overline{id}^{\mathcal{S}}_{\downarrow f} = (id_{f(i)} : S_{f(i)} \to S_{f(i)} \mid i \in [n])$. In the pushout step (1) we will construct a pushout of inclusion chains of equal length and obtain a chain $\mathcal{D}_{\downarrow f} := (\overline{D}_{\downarrow f}, n, \tau^{\mathcal{D}}_{\downarrow f})$ with $\overline{D}_{\downarrow f} = [D_{f(n)}, D_{f(n-1)}, \ldots, D_{f(1)}, D_{f(0)=0}]$ and $\tau^{\mathcal{D}}_{\downarrow f} = (\tau^{\mathcal{D}}_{f(j),f(i)} : D_{f(j)} \mathrel{\circ\!\!\!\longrightarrow} D_{f(i)} \mid n \geq j > i \geq 0)$.

In the pushout step (2) we will fill the gaps in $\mathcal{D}_{\downarrow f}$ with the corresponding untouched graphs from the original inclusion chain $\mathcal{S}$.

Pushouts of Graphs for Inclusion Graph Homomorphisms. Our constructions and proofs rely on the standard construction of pushouts in **Graph** for a span of an inclusion graph homomorphism $\phi : G \hookrightarrow H$ and an arbitrary graph homomorphism $\psi : G \to K$ where we assume that H and K are disjoint. The pushout P is given by $P^N := K^N \cup H^N \setminus G^N$, $P^A := K^A \cup H^A \setminus G^A$ and $sc^P(e) := sc^K(e)$, if $e \in K^A$, and $sc^P(e) := \psi^A(sc^H(e))$, if $e \in H^A \setminus G^A$. tg^P is defined analogously. $\phi^* : K \hookrightarrow P$ is an inclusion graph homomorphism by construction and $\psi^* : H \to P$ is defined for $X \in \{A, N\}$ by $\psi^{*X}(v) := \psi^X(v)$, if $v \in G^X$ and $\psi^{*X}(v) := v$, if $v \in H^X \setminus G^X$.

The pair $G \setminus H := (H^N \setminus G^N, H^A \setminus G^A)$ of subsets of nodes and arrows of H is, in general, not establishing a subgraph of H. We will nevertheless use the notation $P = K + H \setminus G$ to indicate that P is constructed as described above. ψ^* can be described then as a sum of two parallel pairs of mappings

$$\psi^* = \psi + id_{H \setminus G} := (\psi^N + id_{H^N \setminus G^N}, \psi^A + id_{H^A \setminus G^A}) \tag{4}$$

Pushout for Inclusion Chains with Equal Depth. We consider now the span

$$\begin{tikzcd}
\mathcal{S}_{\downarrow f} & \mathcal{L} \arrow[l, "{(\mu, id_{[n]})}"'] \arrow[r, hook, "{(\lambda, id_{[n]})}"] & \mathcal{I}
\end{tikzcd}$$

of reduct morphisms in **Chain** (see Fig. 12). For each level $i \in [n]$ we construct the corresponding pushout of graph homomorphisms. This ensures, especially,

$$\lambda_i ; \delta_i = \mu_i ; \varsigma_{f(i)} \quad \text{for all} \quad i \in [n]. \tag{5}$$

$$\begin{tikzcd}
L_0 = L \arrow[rr, hook, "\lambda_0 = \lambda"] \arrow[dr, "\mu_0"] & & I_0 = I \arrow[dr, "\delta_0"] & \\
& S_0 = S \arrow[rr, hook, "\varsigma_0 = \varsigma"'] \arrow[ur, phantom, "PO"] & & D_0 = D = S + I \setminus L \\
L_i \arrow[uu, hook, "\tau^{\mathcal{L}}_{i,0}"] \arrow[rr, hook, "\lambda_i"] \arrow[dr, "\mu_i"] & & I_i \arrow[uu, hook, "\tau^{\mathcal{I}}_{i,0}"] \arrow[dr, "\delta_i"] & \\
& S_{f(i)} \arrow[uu, hook, "\tau^{\mathcal{S}}_{f(i),0}"] \arrow[rr, hook, "\varsigma_{f(i)}"'] \arrow[ur, phantom, "PO"] & & D_{f(i)} = S_{f(i)} + I_i \setminus L_i \arrow[uu, dotted, "\tau^{\mathcal{D}}_{f(i),0}(= \tau^{\mathcal{S}}_{f(i),0} + \tau^{\mathcal{I}}_{i,0\downarrow I_i \setminus L_i})"']
\end{tikzcd}$$

Fig. 13. Level-wise pushout construction

We look at an arbitrary level $n \geq i \geq 1$ together with the base level 0 (see Fig. 13). We get a cube where the top and bottom square are pushouts by construction. In addition, the left and back square are pullbacks since $(\mu, id_{[n]})$ and $(\lambda, id_{[n]})$, respectively, are reduct morphisms. We get a unique graph homomorphism $\tau^{\mathcal{D}}_{f(i),0} : D_{f(i)} \to D$ that makes the cube commute. By the uniqueness of mediating morphisms and the fact that the top pushout square has the Van Kampen property (see [12,25]), we can conclude that the front and the right square are pullbacks as well. That the back square is a pullback means nothing but $L_i = L \cap I_i$. This entails $I_i \setminus L_i \subseteq I \setminus L$ thus $\tau^{\mathcal{D}}_{f(i),0}$ turns out to be the sum of the two inclusions $\tau^{\mathcal{S}}_{f(i),0} : S_{f(i)} \hookrightarrow S$ and $\tau^{\mathcal{I}}_{i,0\downarrow I_i \setminus L_i} : I_i \setminus L_i \hookrightarrow I \setminus L$ and is therefore an inclusion itself.

The sequence $[D_{f(n)}, D_{f(n-1)}, \ldots, D_{f(1)}, D_0]$ of subgraphs of $D = D_0$ defines the intended inclusion chain $\mathcal{D}_{\downarrow f}$. Since the front and right squares in Fig. 13 are pullbacks, Lemma 2 ensures that the family $\varsigma_{\downarrow f} = (\varsigma_{f(i)} : S_{f(i)} \hookrightarrow D_{f(i)} \mid i \in [n])$ of inclusion graph homomorphisms constitutes a reduct morphism $(\varsigma_{\downarrow f}, id_{[n]}) : \mathcal{S}_{\downarrow f} \rightarrow \mathcal{D}_{\downarrow f}$ while the family $\delta = (\delta_i : I_i \rightarrow D_{f(i)} \mid i \in [n])$ of graph homomorphisms constitutes a reduct morphism $(\delta, id_{[n]}) : \mathcal{I} \rightarrow \mathcal{D}_{\downarrow f}$. Finally, Eq. 5 ensures that the resulting square (1) of reduct morphisms in Fig. 12 commutes. The proof that we have constructed a pushout in **Chain** is given in [26].

Remark 2 (Only one pushout). $\varsigma_{f(i)}$ and δ_i are jointly surjective for all $n \geq i \geq 1$ thus we can describe $D_{f(i)}$ as the union $D_{f(i)} = \varsigma(S_{f(i)}) \cup \delta(I_i)$. Hence in practice, there is no need for an explicit construction of pushouts at all the levels $n \geq i \geq 1$; these are all constructed implicitly by the pushout construction at level 0.

Pushout by Chain Extension. To obtain an inclusion chain $\mathcal{D}$ of length $m+1$, we fill the gaps in $\mathcal{D}_{\downarrow f}$ by corresponding subgraphs of S: $D_a := D_a$ if $a \in f([n])$ and $D_a := S_a$ if $a \in [m] \setminus f([n])$ and obtain the intended inclusion chain $\mathcal{D} = (\overline{D}, m, \tau^{\mathcal{D}})$. The family $\overline{id}^{\mathcal{D}}_{\downarrow f} = (id_{D_{f(i)}} : D_{f(i)} \rightarrow D_{f(i)} \mid i \in [n])$ of identities defines trivially a reduct morphism $(\overline{Id}^{\mathcal{D}}_{\downarrow f}, f) : \mathcal{D}_{\downarrow f} \rightarrow \mathcal{D}$. One can show that the family $\varsigma = (\varsigma_a : S_a \rightarrow D_a \mid a \in [m])$ of graph homomorphisms defined by

$$\varsigma_a := \begin{cases} \varsigma_a : S_a \hookrightarrow D_a & \text{if } a \in f([n]) \\ id_{S_a} : S_a \rightarrow D_a = S_a & \text{if } a \in [m] \setminus f([n]) \end{cases}$$

establishes a reduct morphism $(\varsigma, id_{[m]}) : \mathcal{S} \rightarrow \mathcal{D}$. The two reduct morphisms $(\overline{id}^{\mathcal{D}}_{\downarrow f}, f) : \mathcal{D}_{\downarrow f} \rightarrow \mathcal{D}$ and $(\varsigma, id_{[m]}) : \mathcal{S} \rightarrow \mathcal{D}$ establish square (2) in Fig. 12 that commutes trivially. In [26] it is shown that square (2) is also a pushout in **Chain**.

Pullback Complement. We construct the reduct of $\mathcal{D} = (\overline{D}, m, \tau^{\mathcal{D}})$ along $\theta : T \hookrightarrow D$ and $id_{[m]}$ by level-wise intersection (pullback) for all $n \geq i \geq 1$ (see the pullback square below). Due to Lemma 2, we obtain, in such a way, an inclusion chain $\mathcal{T} = (\overline{T}, m, \tau^{\mathcal{T}})$ together with a reduct morphism $(\theta, id_{[m]}) : \mathcal{T} \rightarrow \mathcal{D}$. The multilevel typing of $\mathcal{T}$ is simply borrowed from $\mathcal{D}$, that is, we define (see Fig. 11)

$$(\sigma^{\mathcal{T}}, id_{[m]}) := (\theta, id_{[m]}); (\sigma^{\mathcal{D}}, id_{[m]}) \tag{6}$$

and this gives us trivially the intended type compatibility of $(\theta, id_{[m]})$. The typing chain morphism $(\nu, f) : \mathcal{R} \rightarrow \mathcal{T}$ with $\nu = (\nu_i : R_i \rightarrow T_{f(i)} \mid i \in [n])$ such that

$$(\rho, id_{[n]}); (\delta, f) = (\nu, f); (\theta, id_{[m]}) \tag{7}$$

is simply given by pullback composition and decomposition in **Graph**: For each $n \geq i \geq 1$ we consider the following incomplete cube on the right-hand side:

$$\begin{tikzcd}
D_0 = D \arrow[d, phantom, ""] & & T_0 = T \arrow[ll, hook', "\theta_0 = \theta"'] \\
& PB & \\
D_i \arrow[uu, hook, "\tau^{\mathcal{D}}_{i,0}"] & & T_i \arrow[ll, hook', "\theta_i"'] \arrow[uu, hook, "\tau^{\mathcal{T}}_{i,0}"']
\end{tikzcd}$$

$$\begin{tikzcd}
I_0 = I \arrow[dr, "\delta_0 = \delta"] & & R_0 = R \arrow[ll, hook', "\rho_0 = \rho"'] \arrow[dr, "\nu_0 = \nu"] & \\
& D_0 = D & & T_0 = T \arrow[ll, hook', "\theta_0 = \theta"'] \\
I_i \arrow[uu, hook, "\tau^{\mathcal{I}}_{i,0}"] \arrow[dr, "\delta_i"'] & & R_i \arrow[ll, hook', "\rho_i"'] \arrow[uu, hook, "\tau^{\mathcal{R}}_{i,0}"'] \arrow[dr, dotted, "\nu_i"] & \\
& D_{f(i)} \arrow[uu, hook, "\tau^{\mathcal{D}}_{f(i),0}"'] & & T_{f(i)} \arrow[ll, hook', "\theta_{f(i)}"'] \arrow[uu, hook, "\tau^{\mathcal{T}}_{f(i),0}"']
\end{tikzcd}$$

The back square, the left square as well as the front square are pullbacks since $(\rho, id_{[n]})$, (δ, f) and $(\theta, id_{[m]})$, respectively, are reduct morphisms. The top square is constructed as a pullback complement. The diagonal square from $\tau^{\mathcal{R}}_{i,0}$ to $\tau^{\mathcal{D}}_{f(i),0}$ is a pullback due to the composition of the back pullback and the left pullback. The decomposition of this diagonal pullback w.r.t. the front pullback gives us $\nu_i : R_i \to T_i$ making the cube, and especially the bottom square, commute and making the right square to a pullback as well.

According to Lemma 2 the family $\nu = (\nu_i : R_i \to T_{f(i)} \mid i \in [n])$ of graph homomorphisms defines a reduct morphism $(\nu, f) : \mathcal{R} \to \mathcal{T}$ where condition 7 is simply satisfied by construction. Finally, (ν, f) is also type compatible since conditions 6 and 7 ensure that the roof square in Fig. 11 commutes.

Example 4. To present a non-trivial rule application for our example, we discuss the undesired application of the plain version of rule *CreatePart* (see Fig. 14), mentioned in Example 3, for a state of the hammer configuration with only one node `ghead`, as shown in `hammer_config_0` in Fig. 15. So, we have $f : [1] \to [2]$, with $f(0) = 0, f(1) = 1$, and the "undesired match" of $MM_1 = (\ \texttt{M1} \xrightarrow{\texttt{cr}} \texttt{P1}\ )$ with $TG_1 = \texttt{generic\_plant} = (\ \texttt{Machine} \xrightarrow{\texttt{creates}} \texttt{Part}\ )$ together with the trivial match of the left-hand side $L = (\,\texttt{m1}\,)$ of the rule with $\texttt{hammer\_config\_0} = (\,\texttt{ghead}\,)$. The resulting inclusion chains $\mathcal{S}$, $\mathcal{L}$, $\mathcal{R}$ and two reduct morphisms between them are depicted in Fig. 14. Note, that the ellipse and cursive labels indicate here the corresponding typing chain morphisms $(\sigma^{\mathcal{S}}, id_{[2]})$, $(\sigma^{\mathcal{L}}, id_{[1]})$ and $(\sigma^{\mathcal{R}}, id_{[1]})$, respectively.

For the two levels in $f([1]) = \{0, 1\} \subset [2]$ we construct the pushouts D_0 and D_1 while D_2 is just taken as S_2. The lowest level in $\mathcal{D}$, where the new elements `p1` and `c` appear, is level 1 thus the constructed direct types of `p1` and `c` are `Part` and `creates`, resp., as shown in `hammer_config_1` in Fig. 15.

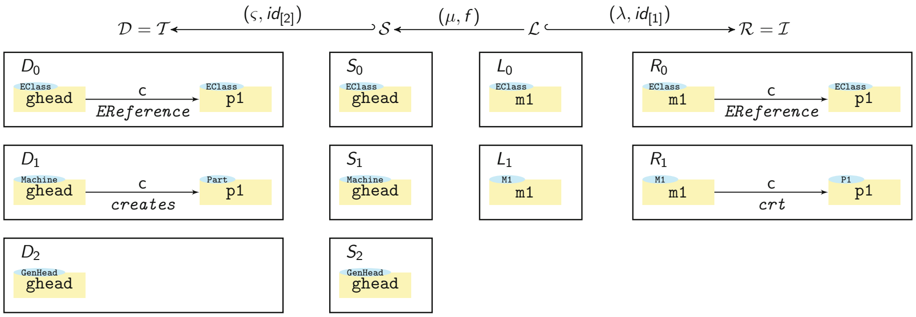

Fig. 14. Inclusion chains for the plain version of *CreatePart*

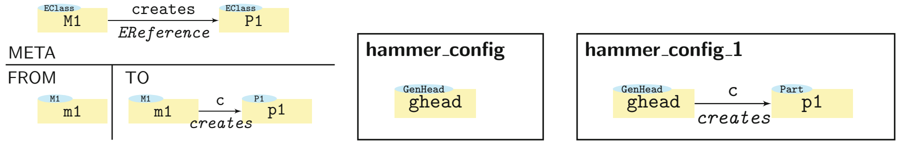

Fig. 15. Plain version of *CreatePart* and its application

4 Conclusions, Related and Future Work

Conclusion. Multilevel modeling offers more flexibility on top of traditional modeling techniques by supporting an unlimited number of abstraction levels. Our approach to multilevel modeling enhances reusability of concepts and their behaviour by allowing the definition of flexible transformation rules which are applicable to different hierarchies with a variable number of levels. In this paper, we have presented a formalization of these flexible and reusable transformation rules based on graph transformations. We represent multilevel models by multilevel typed graphs whose manipulation and transformation are carried out by multilevel typed graph transformation rules. These rules are cospans of three graphs and two inclusion graph homomorphisms where the three graphs are multilevel typed over a common typing chain. As these rules are represented as cospans, their application is carried out by a pushout and a final pullback complement construction for the underlying graphs in the category **Graph**. We have identified type compatibility conditions, for rules and their matches, which are crucial for rule applications. Moreover, we have shown that typed graph transformations can be generalized to multilevel typed graph transformations improving preciseness, flexibility and reusability of transformation rules.

Related work. The theory and practise of graph transformations are well-studied, and the concept of model transformations applied to MLM is not novel. Earlier works in the area have worked in the extension of pre-existing model transformation languages to be able to manipulate multilevel models and model hierarchies. In [3], the authors adapt ATL [15] to manipulate multilevel models built with the Melanee tool [2]. In a similar manner, [11] proposes the adaptation of ETL [16] and other languages from the Epsilon family [14] for the application of model transformation rules into multilevel hierarchies created with MetaDepth [8]. These works, however, tackle the problem from the practical point of view. That is, how to reuse mature off-the-shelf tools for model transformation in the context of MLM, via the manipulation of a "flattened" representation of the hierarchy to emulate multilevel transformations. Our approach, on the contrary, has been developed from scratch with a multilevel setting in mind, and we believe it can be further extended to tackle all scenarios considered by other approaches. Therefore, to the best of our knowledge, there are no formal treatments of multilevel typed graph transformations in the literature except for our previous works [19,20,26] (see Sect. 4 in [26]). Hence, we consider our approach the first approximation to formally address the challenges which come with multilevel modeling and multilevel model transformations.

Common for our work and [9] is that the concepts of typing chains, multilevel typed graphs and multilevel models originate in [22]. However, [9] presents partial morphisms as spans of total morphisms and does not use the composition of those spans explicitly. Wrt. typing chains, a multilevel model in [9] is a sequence of graphs $[G_n, G_{n-1}, \ldots, G_1, G_0]$ together with the subfamily $(\tau^{\mathcal{G}}_{i+1,i} : G_{i+1} \mathrel{-\!\!\circ\!\!\!\rightarrow} G_i \mid n \geq i \geq 0)$ of typing morphisms.

Future work. Although it is trivial to see that the bottom square in the cube for the pullback complement step becomes a pullback for all $n \geq i \geq 1$, we leave it for future work to prove that we indeed have constructed a final pullback complement in **Chain**. A utilization of our theory to deal with coupled transformations [21] in the setting of multilevel typed modelling is also desirable. Furthermore, it would be interesting to investigate the category **Chain** for its own; e.g., study its monomorphisms and epimorphisms, possible factorization systems, and the conditions for existence of general pushouts and pullbacks.

References

1. Almeida, J.P.A., Frank, U., Kühne, T.: Multi-level modelling (Dagstuhl Seminar 17492). Dagstuhl Reports **7**(12), 18–49 (2018). https://doi.org/10.4230/DagRep.7.12.18
2. Atkinson, C., Gerbig, R.: Flexible deep modeling with melanee. In: Stefanie Betz and Ulrich Reimer, editors, Modellierung 2016, 2.-4. März 2016, Karlsruhe - Workshopband, Modellierung 2016, Bonn, vol. 255, pp. 117–122 (2016). Gesellschaft für Informatik
3. Atkinson, C., Gerbig, R., Tunjic, C.V.: Enhancing classic transformation languages to support multi-level modeling. Software Syst. Model. **14**(2), 645–666 (2015). https://doi.org/10.1007/s10270-013-0384-y
4. Atkinson, C., Kühne, T.: The essence of multilevel metamodeling. In: Gogolla, M., Kobryn, C. (eds.) UML 2001. LNCS, vol. 2185, pp. 19–33. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45441-1_3

5. Atkinson, C., Kühne, T.: Rearchitecting the UML infrastructure. ACM Trans. Model. Comput. Simul. **12**(4), 290–321 (2002). https://doi.org/10.1145/643120.643123
6. Atkinson, C., Kühne, T.: Reducing accidental complexity in domain models. Software Syst. Model. **7**(3), 345–359 (2008). https://doi.org/10.1007/s10270-007-0061-0
7. Corradini, A., Heindel, T., Hermann, F., König, B.: Sesqui-pushout rewriting. In: Corradini, A., Ehrig, H., Montanari, U., Ribeiro, L., Rozenberg, G. (eds.) ICGT 2006. LNCS, vol. 4178, pp. 30–45. Springer, Heidelberg (2006). https://doi.org/10.1007/11841883_4
8. de Lara, J., Guerra, E.: Deep meta-modelling with MetaDepth. In: Vitek, J. (ed.) TOOLS 2010. LNCS, vol. 6141, pp. 1–20. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13953-6_1
9. de Lara, J., Guerra, E.: Multi-level model product lines. FASE 2020. LNCS, vol. 12076, pp. 161–181. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45234-6_8
10. de Lara, J., Guerra, E., Cuadrado, J.S.: When and how to use multilevel modelling. ACM Trans. Softw. Eng. Methodol. **24**(2), 12:1–12:46 (2014). https://doi.org/10.1145/2685615
11. de Lara, J., Guerra, E., Cuadrado, J.S.: Model-driven engineering with domain-specific meta-modelling languages. Software Syst. Model. **14**(1), 429–459 (2015). https://doi.org/10.1007/s10270-013-0367-z
12. Implementation of typed attributed graph transformation by AGG. Fundamentals of Algebraic Graph Transformation. MTCSAES, pp. 305–323. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-31188-2_15
13. Ehrig, H., Hermann, F., Prange, U.: Cospan DPO approach: an alternative for DPO graph transformations. Bulletin EATCS **98**, 139–149 (2009)
14. The Eclipse Foundation. Epsilon (2012). http://www.eclipse.org/epsilon/
15. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: a model transformation tool. Sci. Comput. Program. **72**(1–2), 31–39 (2008). https://doi.org/10.1016/j.scico.2007.08.002
16. Kolovos, D.S., Paige, R.F., Polack, F.A.C.: The epsilon transformation language. In: Vallecillo, A., Gray, J., Pierantonio, A. (eds.) ICMT 2008. LNCS, vol. 5063, pp. 46–60. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69927-9_4
17. Macías, F.: Multilevel modelling and domain-specific languages. Ph.D. thesis, Western Norway University of Applied Sciences and University of Oslo (2019)
18. Macías, F., Rutle, A., Stolz, V.: MultEcore: combining the best of fixed-level and multilevel metamodelling. In: MULTI, CEUR Workshop Proceedings, vol. 1722. CEUR-WS.org (2016)
19. Macías, F., Rutle, A., Stolz, V., Rodriguez-Echeverria, R., Wolter, U.: An approach to flexible multilevel modelling. Enterp. Model. Inf. Syst. Archit. **13**, 10:1–10:35 (2018). https://doi.org/10.18417/emisa.13.10
20. Macías, F., Wolter, U., Rutle, A., Durán, F., Rodriguez-Echeverria, R.: Multilevel coupled model transformations for precise and reusable definition of model behaviour. J. Log. Algebr. Meth. Program. **106**, 167–195 (2019). https://doi.org/10.1016/j.jlamp.2018.12.005
21. Mantz, F., Taentzer, G., Lamo, Y., Wolter, U.: Co-evolving meta-models and their instance models: a formal approach based on graph transformation. Sci. Comput. Program. **104**, 2–43 (2015). https://doi.org/10.1016/j.scico.2015.01.002

22. Rossini, A., de Lara, J., Guerra, E., Rutle, A., Wolter, U.: A formalisation of deep metamodelling. Formal Aspects Comput. **26**(6), 1115–1152 (2014). https://doi.org/10.1007/s00165-014-0307-x
23. Rutle, A., Rossini, A., Lamo, Y., Wolter, U.: A formal approach to the specification and transformation of constraints in MDE. J. Log. Algebr. Program. **81**(4), 422–457 (2012). https://doi.org/10.1016/j.jlap.2012.03.006
24. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework, 2nd edn. Addison-Wesley Professional, Boston (2008)
25. Wolter, U., König, H.: Fibred amalgamation, descent data, and van kampen squares in topoi. Appl. Categor. Struct. **23**(3), 447–486 (2013). https://doi.org/10.1007/s10485-013-9339-2
26. Wolter, U., Macías, F., Rutle, A.: The category of typing Chains as a foundation of multilevel typed model transformations. Technical Report 2019–417, University of Bergen, Department of Informatics, November 2019