Addendum to the paper *Rendering Model Transformations Validity-Preserving* (Correctness Proofs)

Nebras Nassar, Jens Kosiol, Thorsten Arendt, and Gabriele Taentzer

Philipps-Universität Marburg, Marburg, Germany
{nassarn,taentzer}@informatik.uni-marburg.de
kosiolje@mathematik.uni-marburg.de
thorsten.arendt@uni-marburg.de

In this work, we formally state and prove the correctness of the claims we made in our paper *Rendering Model Transformations Validity-Preserving*. The numbering of sections follows this paper. The presentation assumes familiarity with our formal background as, e.g., presented in [1,3].

5 Proofs for Correctness of Suggested Refinements and Optimizations

5.1 Dealing with EMF's Built-In Negative Constraints

Theorem 1. *Given any graph condition c over a graph G, let c' be the graph condition that results by replacing every occurrence of a subcondition $\exists(a : C_1 \hookrightarrow C_2)$ by* `false` *if the graph C_2 contains parallel edges or multiple incoming containment edges to the same node. Then any monomorphism $g : G \to M$ into an EMF-model graph M satisfies c iff it satisfies c'. In particular, if c is a constraint any EMF-model graph M satisfies c iff it satisfies c', i.e., $M \models c \Leftrightarrow M \models c'$.*

Proof. We prove the statement using structural induction.

The statement holds for `true` since `true'` = `true` and $h \models$`true` for every monomorphism h.

Let $c = \exists(a : C_1 \to C_2, d)$ be a condition and $h \models d \Leftrightarrow h \models d'$ for each monomorphism h from C_2.

First, if C_2 does neither contain parallel edges nor multiple incoming containment edges to the same node, then $c' = \exists(a : C_1 \to C_2, d')$. Now, for every monomorphism $g : C_1 \to M$, where M is any EMF-model graph

$$\begin{aligned} g \models c &\Leftrightarrow \exists h : C_2 \to M, \text{ s.t. } h \circ a = g \text{ and } h \models d \\ &\Leftrightarrow \exists h : C_2 \to M, \text{ s.t. } h \circ a = g \text{ and } h \models d' \\ &\Leftrightarrow g \models c' \ . \end{aligned}$$

Secondly, if C_2 contains parallel edges or multiple incoming containment edges to the same node, then $c' = \exists(\texttt{false}, d')$. Therefore, no monomorphism

$g : C_1 \to M$ satisfies c', where M is any graph. But since no EMF-model graph M contains parallel edges or multiple incoming containment edges to the same node, there does not exist any monomorphism $h : C_2 \to M$ into any EMF-model graph M. Hence, there does never exist a monomorphism h s.t. $h \circ a = g$ and $h \models d'$. In summary, $g \not\models c \Leftrightarrow g \not\models c'$ for all monomorphisms $g : C_1 \to M$ for any EMF-model graph M.

The induction step for boolean operators is routine. □

Remark 1. The same kind of argument would also apply for the case of containment cycles. But since containment cycles of arbitrary length cannot be expressed as graph constraints, the correctness of replacing their occurrence by `false` is intuitive but not amendable to formal proof in our chosen framework.

5.3 From Global to Local Checks

We are interested in the possibilities to simplify graph conditions arising from integration of graph constraints into rules. The background is as follows: Given a graph constraint c and a rule r, *OCL2AC* implements the so-called calculation of the *c-guaranteeing rule*, i. e., it computes an additional application condition for r, such that r becomes only applicable at matches where constraint c holds after application of the rule at that match. If an instance is known to be valid, calculation of a rule with weaker application condition, the so-called *c-preserving rule* would suffice. Albeit the application condition of the c-preserving rule is (logically) weaker, it is generally more complex. Thus, we are interested in developing an "overapproximation" for an application condition, which just preserves a constraint, by omitting parts of the application condition arising when calculating the application condition for the c-guaranteeing rule.

1. Simplifying Application Conditions when Rule Actions and Constraint Do Not Overlap. Intuitively, if application of a rule has no effect on elements referred to in a given constraint, application of that rule does not effect the validity of the constraint. We specify and proof this in the following theorem.

Theorem 2. *Let c be a graph constraint and $r = (L \hookleftarrow K \hookrightarrow R)$ be a rule. Let there neither be an intersection of the elements of $L \setminus K$, i.e., the elements the rule deletes, nor of the elements of $R \setminus K$, i.e., the elements the rule creates, with any graph of the constraint c (respecting inheritance).*

Then for all graphs $G \Rightarrow_r H$ where $G \models c$ also $H \models c$.

Proof. The above statement can be proved by reducing it to the well-known notion of sequential independence:

The empty intersections imply that any constant rule $P \hookrightarrow P$, which just checks for the existence of the graph P, where P is some graph from the constraint c, is sequentially and parallelly independent of the application of the rule r. This means that each morphism from P to H can be restricted to a morphism from P to G and if there is no morphism from P to H, there is none from P to G. For details of this argument, compare for example [2].

Because the semantics of graph constraints is given by existence of such morphisms, an easy induction shows that $G \models c \Leftrightarrow H \models c$ in this case. □

2. Not Integrating Positive Constraints into Creating Rules, Not Integrating Negative Constraints into Deleting Rules.

Theorem 3. *Given a positive constraint c, i.e., a constraint of the form $c = \exists C$, and a monotonic rule $r : L \hookrightarrow R$, i.e., a rule which only creates, than for every graph G it holds that $G \models c \Rightarrow H \models c$ for every $G \Rightarrow_r H$.*

Analogously, given a negative constraint c, i.e., a constraint of the form $c = \neg\exists C$, and a rule $r : L \hookleftarrow R$, i.e., a rule which only deletes, than for every graph G it holds that $G \models c \Rightarrow H \models c$ for every $G \Rightarrow_r H$.

Proof. The proofs are completely analogous to the one above: Let H denote a graph which arises by application of rule r to a graph G, i.e., $G \Rightarrow_r H$. A monotonic rule is parallelly independent from the rule which just checks for the existence of the graph C. Hence, every match for the graph C in G extends to a match for graph C in H, i.e., $G \models c \Rightarrow H \models c$.

Dually, checking the existence of a graph C is sequentially independent of applying a rule which only deletes. Hence, every match for C in a graph H can be extended to a match for C in G, i.e., $G \models c \Rightarrow H \models c$. □

3. Simplifying Application Conditions Resulting from Negative Constraints. Negative constraints which are not further nested may be simplified by omitting some parts of the resulting application condition, concretely, when overlapping the constraint with the RHS of a rule, all situations where only elements that are *not* newly created by the rule overlap, may be omitted. Let a negative constraint $c = \neg\exists C$ be given. Let $r = (L \hookleftarrow K \hookrightarrow R)$ be any rule. The initially resulting right application condition rac when calculating the c-guaranteeing rule is of the form $rac = \neg(\bigvee_{i \in I} \exists P_i)$ where I, P_i are dependent from the possibilities to overlap C and R. We construct a subset $J \subseteq I$ and the according application condition $rac' = \neg(\bigvee_{j \in J} \exists P_j)$ by omitting all overlaps where not a newly created element, i.e., an element from $R \setminus K$ is identified with one from C. This simpler right application condition leads to a simpler left application condition ac'. We then show that, given a valid input G, if $G \Rightarrow_r H$, where r is equipped with application condition ac', then $H \models c$.

Theorem 4. *Let a negative constraint $c = \neg\exists C$ and a rule $r = (L \hookleftarrow K \hookrightarrow R)$ be given. Let I denote the set indexing all possible ways to overlap graphs R and C and let P_i denote the corresponding overlappings, i.e., graphs that arise by a pair of jointly surjective, injective morphisms from R and C into P_i. Let $J \subseteq I$ denote the subset of I that arises by only keeping those indices $i \in I$ where P_i contains at least one element that arises by identifying an element from $R \setminus K$ (as elements, not as graph; i.e., an element, newly created by rule application) with one from C. Let $rac_J := \neg(\bigvee_{j \in J} \exists P_j)$. Let ac_J denote the application condition that arises by translating the right application condition rac_J to the LHS of rule r.*

Given a transformation $G \Rightarrow_r H$, where G is valid and r is equipped with the application condition ac_J, then $H \models c$.

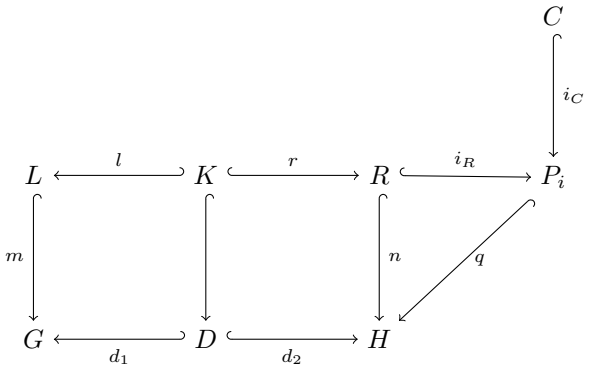
Proof. We prove the above statement by showing that if $H \not\models c$ then already $G \not\models c$. The idea behind the proof is analogous to the ones of the above proofs.

For the following notations, compare the diagram below. Assume, $G \Rightarrow_{r,m} H$, r is equipped with application condition ac_J and $H \not\models C$. By construction, there exists $i \in I$ such that exists a monomorphism $q : P_i \hookrightarrow H$ with $n = q \circ i_R$. Since r is equipped with application condition ac_J, $i \in I \setminus J$ (otherwise rule application would have been prevented by that application condition).

The following equation holds by definition:

$$\begin{aligned} n \not\models \neg\exists(i_R : R \hookrightarrow P_i) &\Leftrightarrow n \models \exists(i_R : R \hookrightarrow P_i) \\ &\Leftrightarrow \exists q : P_i \hookrightarrow H, \text{ s.t. } n = q \circ i_R \ . \end{aligned}$$

Now, like in the proofs above, checking for the existence of $P_i, i \in I \setminus J$ in graph H is independent of applying rule r. This means, for every monomorphism $q : P_i \hookrightarrow H$ the graph $q(i_C(C))$ is in the domain of the partial and injective morphism d_2^{-1}. We define $n' : C \hookrightarrow G$ as $d_1 \circ d_2^{-1} \circ q \circ i_C$. Thus, $G \models \exists C$ and hence $G \not\models c = \neg\exists C$.



□

References

1. Habel, A., Pennemann, K.H.: Correctness of high-level transformation systems relative to nested conditions. Mathematical Structures in Computer Science **19**, 245–296 (2009)
2. Lambers, L.: Certifying rule-based models using graph transformation (2010)
3. Radke, H., Arendt, T., Becker, J.S., Habel, A., Taentzer, G.: Translating essential OCL invariants to nested graph constraints for generating instances of meta-models. Science of Computer Programming **152**, 38–62 (2018)