Semantical Correctness of Simulation-to-Animation Model and Rule Transformation: Long Version

Claudia Ermel[1], Hartmut Ehrig[1] and Karsten Ehrig[2]

[1]Technische Universität Berlin, Germany
{lieske,ehrig}@cs.tu-berlin.de

[2]University of Leicester, UK
{karsten}@mcs.le.ac.uk

Forschungsberichte des Fachbereichs Informatik

Bericht-Nr. 2006/??, ISSN 1436-9915

Abstract

In the framework of graph transformation, simulation rules are well-known to define the operational behavior of visual models. Moreover, it has been shown already how to construct animation rules in a domain specific layout from simulation rules. An important requirement of this construction is the semantical correctness which has not yet been considered. In this paper we give a precise definition for *simulation-to-animation* ($S2A$) model and rule transformations. Our main results show under which conditions semantical correctness can be obtained in the cases without and with negative application conditions for rules. The results are applied to show the semantical correctness of the $S2A$ transformation of a Radio Clock model.

Keywords: graph transformation, model and rule transformation, semantical correctness, simulation, animation

Contents

1 Introduction

In recent years, visual models represented by graphs have become very popular in model-based software development, as the wide-spread use of UML and Petri nets proves. For the definition of an operational semantics for visual models, the transformation of graphs plays a similar central role as term rewriting in the traditional case of textual models. The area of graph transformation provides a rule-based setting to express the semantics of visual models (see e.g. [Roz97]). The objective of *simulation rules* is their application to the states of a visual model, deriving subsequent model states, thus characterizing system evolution. A *simulation scenario*, i.e. a sequence of such simulation steps can be visualized by showing the states before and after each simulation rule application as graphs.

For validation purposes, simulation may be extended to a domain specific view, called *animation view* [EB04, EE05b, EHKZ05], which allows to define scenario visualizations in the layout of the application domain. The animation view is defined by extending the alphabet of the original visual modeling language by symbols representing entities from the application domain. The simulation rules for a specific visual model are translated to the animation view by performing a *simulation-to-animation model and rule transformation* ($S2A$ transformation), realizing a consistent mapping from simulation steps to animation steps which can be visualized in the animation view layout. $S2A$ transformation is defined by a set of $S2A$ graph transformation rules, and an additional formal construction allowing to apply $S2A$ rules to simulation rules, resulting in a new set of graph transformation rules, called *animation rules*.

Comparable theoretical research in the area of applying graph transformation rules to rules has been done by Parisi-Presicce [PP96]. His approach has provided the basis of our definition of $S2A$ transformations which additionally allows to transform not only the rule interfaces, and which also treats negative application conditions (NACs), both for the transforming rules and for the transformed rules.

An important requirement is the *semantical correctness* of the $S2A$ transformation in the sense that the behavior of the original model is preserved in the animation view. Up to now, the semantical correctness of the $S2A$ transformation has not been considered. In this paper, we give a formal definition for $S2A$ transformations and show under which conditions semantical correctness can be obtained. In our approach, an $S2A$ transformation generates one animation rule for each simulation rule. Hence, our notion of semantical correctness implies that each animation step (obtained by applying an animation rule) corresponds to a simulation step of the original model. Please note that there are more general definitions for the semantical correctness of model transformations which establish a correspondence between one simulation step in the source model and a sequence of simulation steps in the target model. For our case of $S2A$ transformation it is sufficient to relate single simulation and animation steps. Intermediate animation states providing smooth state transitions are possible nonetheless: They are defined by enriching an animation rule by animation operations to specify continuous changes of object properties such as size, color or position. Since animation operations leave the states before and after a rule application unchanged, they do not influence the semantical correctness of the $S2A$ transformation. Our approach has been implemented in the generic visual modeling environment GenGED [Gen]. The implementation includes an animation editor, where animation operations can be defined visually, and animation scenarios can be exported to to the SVG format [WWW03].

There exist related tool-oriented approaches, where different visual representations are used to visualize a model's behavior. One example is the *reactive animation* approach by Harel [HEC03]. Here, the reactive system behavior is specified with tools like Rhapsody [Rha05] using UML, or the Play-Engine [HM03] using Life-Sequence-Charts, an extension of State-charts. The animated representation of the system behavior is implemented by linking these tools to pure animation tools like Flash or Director from Macromedia [Mac04]. Hence, the mapping from simulation to animation views happens at the implementation level and is not specified formally. Furthermore, different Petri net tools also offer support for customized Petri net animations (e.g. the SimPEP tool [Gra99] to animate transition firings of low-level Petri nets). In general, approaches to enhance the front end of CASE tools for simulating/animating the behavior of models are restricted to one specific modeling language. In our approach we integrate animation views at model level with graph transformation representations for different visual modeling languages based on a formal specification. This provides the model designer with more flexibility, as the modeling language to be enhanced by animation features, can be freely chosen.

The paper is organized as follows: Section 2 presents the basic concepts of simulation and animation, illustrated by our case study in Section 3. In Section 4, the main result on semantical correctness of $S2A$ transformation is given in the case without NACs. Extensions to cope with NACs are discussed. Explicit proofs for the case with NACs are given in Section 5, and the semantical correctness of the Radio Clock case study is presented in Section 6. Section 7 contains a summary as well as an overview on ongoing work.

2 Basic Concepts of Simulation and Animation

We use typed algebraic graph transformation systems (TGTS) in the double-pushout-approach (DPO) [EEPT06] which have proven to be an adequate formalism for visual language (VL) modeling. A VL is modeled by a type graph capturing the definition of the underlying visual alphabet, i.e. the symbols and relations which are available. Sentences or diagrams of the VL are given by graphs typed over the alphabet type graph. We distinguish the abstract and the concrete syntax in alphabets and models, where the concrete syntax includes the abstract symbols and relations, and additionally defines their layout. Formally, a VL can be considered as a subclass of graphs typed over a type graph TG in the category $\mathbf{Graphs_{TG}}$.

For behavioral diagrams like Statecharts, an operational semantics can be given by a set of simulation rules P_S, using the abstract syntax of the modeling VL. A simulation rule $p \in P_S$ is a graph transformation rule, consisting of a triple of graphs $p = (L, I, R)$, called left-hand side, interface and right-hand side, and two injective morphisms $L \leftarrow I \rightarrow R$. Applying the rule p to a graph G means to find a match of L in G and to replace the occurrence $m(L)$ of L in G by R leading to the target graph G' of the graph transformation step. In the DPO approach, the deletion of $m(L)$ and the addition of R are described by two pushouts (a DPO) in the category $\mathbf{Graphs_{TG}}$ of typed graphs. A rule p may be extended by a set of *negative application conditions (NACs)* [EEPT06], describing situations in which the rule should not be applied to G. Formally, the match $L \xrightarrow{m} G$ satisfies a NAC N with the injective NAC morphism $L \xrightarrow{n} N$, if there does not exist an injective graph morphism $N \xrightarrow{x} G$ with $x \circ n = m$. A sequence $G_0 \Rightarrow G_1 \Rightarrow ... \Rightarrow G_n$ of graph transformation steps is called *transformation* and denoted as $G_0 \stackrel{*}{\Rightarrow} G_n$. A transformation $G_0 \stackrel{*}{\Rightarrow} G_n$, where the rules of a rule set P are applied as long as possible (i.e. as long as matches can be found which satisfy the respective NACs), is denoted by $G_0 \stackrel{P\ !}{\Longrightarrow} G_n$.

We define a model's simulation language VL_S, typed over the simulation alphabet TG_S, as a sublanguage of the modeling language VL, such that all diagrams $G_S \in VL_S$ represent different states of the model during simulation. Based on VL_S, the operational semantics of a model is given by a *simulation specification*.

Definition 2.1 (*Simulation Specification*)
Given a visual language VL_S typed over TG_S, i.e. $VL_S \subseteq \mathbf{Graphs_{TG_S}}$, a *simulation specification* $SimSpec_{VL_S} = (VL_S, P_S)$ over VL_S is given by a TGTS (TG_S, P_S) s.t. VL_S is closed under simulation steps, i.e.

$$G_S \in VL_S \text{ and } G_S \Rightarrow H_S \text{ via } p_S \in P_S \text{ implies } H_S \in VL_S.$$

The rules $p_S \in P_S$ are called simulation rules. $\triangle$

In order to transform a simulation specification to an animation view, we define an $S2A$ transformation $S2A = (S2AM, S2AR)$ consisting of a simulation-to-animation model transformation $S2AM$, and a corresponding rule transformation $S2AR$. The $S2AM$ transformation applies $S2A$ transformation rules from a rule set Q to each $G_S \in VL_S$ as long as possible, adding symbols from the application domain to the model state graphs. The resulting set of graphs comprises the animation language VL_A.

Definition 2.2 (*$S2AM$-Transformation*)
Given a simulation specification $SimSpec_{VL_S} = (VL_S, P_S)$ with VL_S typed over TG_S and a type graph TG_A, called animation type graph, with $TG_S \subseteq TG_A$, a *simulation-to-animation model transformation*, short $S2AM$-transformation,

$$S2AM : VL_S \rightarrow VL_A$$

is given by $S2AM = (VL_S, TG_A, Q)$ where (TG_A, Q) is a TGTS with non-deleting rules $q \in Q$, and $S2AM$-transformations $G_S \stackrel{Q\ !}{\Longrightarrow} G_A$ with $G_S \in VL_S$. The *animation language* VL_A is defined by $VL_A = \{G_A |\ \exists\ G_S \in VL_S\ \&\ G_S \stackrel{Q\ !}{\Longrightarrow} G_A\}$. This means $G_S \stackrel{Q\ !}{\Longrightarrow} G_A$ implies $G_S \in VL_S$ and $G_A \in VL_A$, where each intermediate step $G_i \stackrel{q_i}{\Longrightarrow} G_{i+1}$ is called *$S2AM$-step*. $\triangle$

Our aim is not only to transform model states but to obtain a complete animation specification, including animation rules, from the simulation specification. Hence, we define a construction allowing us to apply the $S2A$ transformation rules from Q also to the simulation rules. The following definition extends the construction for rewriting rules by rules given by Parisi-Presicce in [PP96], where a rule q is only applicable to another rule p if it is applicable to the interface graph of p. This means, q cannot be applied if p deletes or generates objects which q needs. In this paper, we want to add animation symbols to simulation rules even if the $S2A$ transformation rule is *not* applicable to the interface of the simulation rule. Hence, we distinguish three cases in Def. 2.3. Case (1) corresponds to the notion of rule rewriting in [PP96], adapted to non-deleting $S2A$ transformation rules. In Case (2), the $S2A$ transformation rule q is not applicable to the interface, but only to the left-hand side of a rule p (p deletes something that is needed by q), and in Case (3), q is only applicable to the right-hand side of p (p generates something that q needs).

Definition 2.3 (*Transformation of Rules by Non-Deleting Rules*)
Given a non-deleting rule $q = (L_q \rightarrow R_q)$ and a rule $p_1 = (L_1 \stackrel{l_1}{\leftarrow} I_1 \stackrel{r_1}{\rightarrow} R_1)$ then q is appicable to p_1 leading to a *rule transformation step* $p_1 \stackrel{q}{\Longrightarrow} p_2$, if the precondition of one of the following three cases is satisfied and $p_2 = (L_2 \stackrel{l_2}{\leftarrow} I_2 \stackrel{r_2}{\rightarrow} R_2)$ is defined according to the corresponding construction

- *Case (1)*
  Precondition (1): There is a match $h : L_q \rightarrow I_1$.
  Construction (1): Let I_2, L_2, and R_2 be defined as pushout objects in the following squares leading to injective morphisms l_2 and r_2

$$\begin{tikzcd}
& L_q \arrow[rr, "q"] \arrow[d, "h"'] & & R_q \arrow[d] \\
& I_1 \arrow[rr, "q_I"] \arrow[dl, "l_1"'] \arrow[dd, "r_1"] & & I_2 \arrow[dl, "l_2"'] \arrow[dd, "r_2"] \\
L_1 \arrow[rr, "q_L"] & & L_2 & \\
& R_1 \arrow[rr, "q_R"] & & R_2
\end{tikzcd}$$

- *Case (2)*
  Precondition (2): There is no match $h : L_q \rightarrow I_1$, but a match $h' : L_q \rightarrow L_1$.
  Construction (2): Let L_2 be defined as pushout in the following diagram and define $I_2 = I_1$, $R_2 = R_1$, $r_2 = r_1$, and $l_2 = q' \circ l_1$

$$\begin{tikzcd}
L_q \arrow[r, "q"] \arrow[d, "h'"'] & R_q \arrow[d] \\
L_1 \arrow[r, "q_L"] & L_2
\end{tikzcd}$$

- *Case (3)*
  Precondition (3): There are no matches $h : L_q \rightarrow I_1$ and $h' : L_q \rightarrow L_1$, but there is a match $h'' : L_q \rightarrow R_1$.
  Construction (3): Let R_2 be defined as pushout in the following diagram and define $L_2 = L_1$, $I_2 = I_1$, $l_2 = l_1$, and $r_2 = q' \circ r_1$

$$\begin{tikzcd}
L_q \arrow[r, "q"] \arrow[d, "h''"'] & R_q \arrow[d] \\
R_1 \arrow[r, "q_R"] & R_2
\end{tikzcd}$$

△

The transformation of rules defined above allows now to define an $S2AR$ transformation of rules, leading to an $S2A$ transformation $S2A = (S2AM, S2AR)$ from the simulation specification $SimSpec_{VL_S}$ to the animation specification $AnimSpec_{VL_A}$.

Definition 2.4 (*$S2AR$-Transformation*)

Given a simulation specification $SimSpec_{VL_S} = (VL_S, P_S)$ and an $S2AM$-transformation $S2AM = (VL_S, TG_A, Q)$ then a *simulation-to-animation rule transformation*, short *$S2AR$-trafo*,

$$S2AR : P_S \to P_A,$$

is given by $S2AR = (P_S, TG_A, Q)$ and $S2AR$ *transformation sequence* $p_S \overset{Q\,!}{\Longrightarrow} p_A$ with $p_S \in P_S$, where rule transformation steps $p_1 \overset{q}{\Longrightarrow} p_2$ with $q \in Q$ (see Def. 2.3) are applied as long as possible.
The *animation rules* P_A are defined by $P_A = \{p_A | \, \exists \, p_S \in P_S \wedge \, p_S \overset{Q\,!}{\Longrightarrow} p_A \, \}$.

This means $p_S \overset{Q\,!}{\Longrightarrow} p_A$ implies $p_S \in P_S$ and $p_A \in P_A$, where each intermediate step $p_i \overset{q_i}{\Longrightarrow} p_{i+1}$ is called $S2AR$-*step*.

△

Definition 2.5 (*Animation Specification and $S2A$ Transformation*)
Given a simulation specification $SimSpec_{VL_S} = (VL_S, P_S)$, an $S2AM$ transformation $S2AM : VL_S \to VL_A$ and an $S2AR$ transformation $S2AR : P_S \to P_A$, then

1. $AnimSpec_{VL_A} = (VL_A, P_A)$ is called *animation specification*, and each transformation step $G_A \overset{p_A}{\Longrightarrow} H_A$ with $G_A, H_A \in VL_A$ and $p_A \in P_A$ is called *animation step*.
2. $S2A : SimSpec_{VL_S} \to AnimSpec_{VL_A}$, defined by $S2A = (S2AM, S2AR)$ is called *simulation-to-animation model and rule transformation*, short $S2A$ transformation.

△

3 Case Study: Radio Clock

In this section, we illustrate the main concepts of Section 2 by the well-known Radio Clock case study from Harel [Har87]. The behavior of a radio clock is modeled by the nested Statechart shown in Fig. 1 (a). The radio clock display can show alternatively the time, the date or allows to set the alarm time. The changes between the modes are modeled by transitions labeled with the event Mode. The nested state Alarm allows to change to modes for setting the hours and the minutes (transition Select) of the alarm time. A Set event increments the number of hours or minutes which are currently displayed.

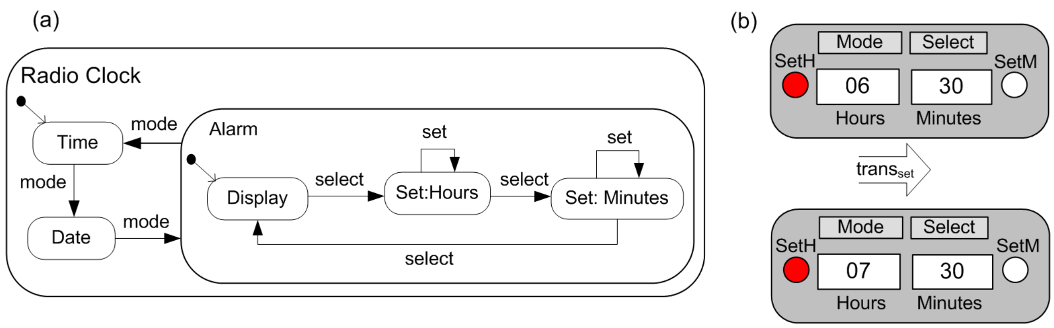

Figure 1: Radio Clock Statechart (a), and Animation View Snapshots (b)

A domain-specific animation view of the Radio Clock is illustrated in Fig. 1 (b). The two snapshots from a possible simulation run of the Statechart in Fig. 1 (a) correspond to the active state Set:Hours before and after the set event has been processed. The animation view shows directly the current display of the clock and indicates by a red light that in the current state the hours may be set. Furthermore, buttons are shown either to proceed to the state where the minutes may be set (button Select), or to switch back to the Time display (button Mode).

The abstract syntax graph of the Radio Clock Statechart is the given by the graph G_I in Fig. 2.

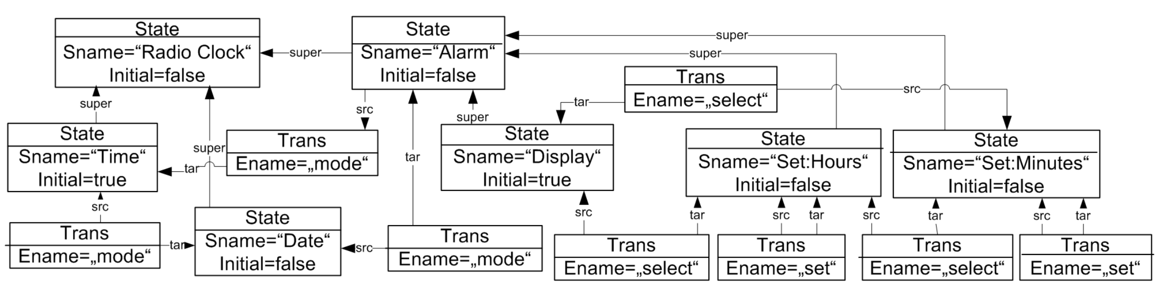

Figure 2: Abstract Syntax Graph G_I of the Radio Clock Statechart

The set of model-specific simulation rules P_S to be applied to G_I is shown in Fig. 3. In the first rule layer, the initialization of an event queue is realized by the rules initial(h,m,e) and addEvent(e) which generate the object, set its current pointer to the top level state "Radio Clock" and fill its event queue. In this way, the events that should be processed during a simulation run, can be defined in the beginning of the simulation. Alternatively, events also may be inserted at the end of the queue while a simulation is running. Furthermore, the object node holds values for the initial alarm time given by the rule parameters of rule initial. The second layer contains all remaining rules and realizes the actual simulation, processing the events in the queue. For each superstate there is a down rule which moves the current pointer from the superstate to its initial substate. Analogously, for each substate there is an up rule moving the current pointer from the substate to its superstate. For each transition there is a trans rule moving the current pointer from the source state of the transition to its target state, if the next event in the queue is the triggering event of the transition. For the transitions named "set", the value of hours or the minutes of the current alarm time are incremented by the respective rule (i.e. in case the hours variable has the value 23, the incr method would replace it by 00, and the minutes variable 59 would be replaced by 00; in all other cases, the incr method corresponds to "+ 1").

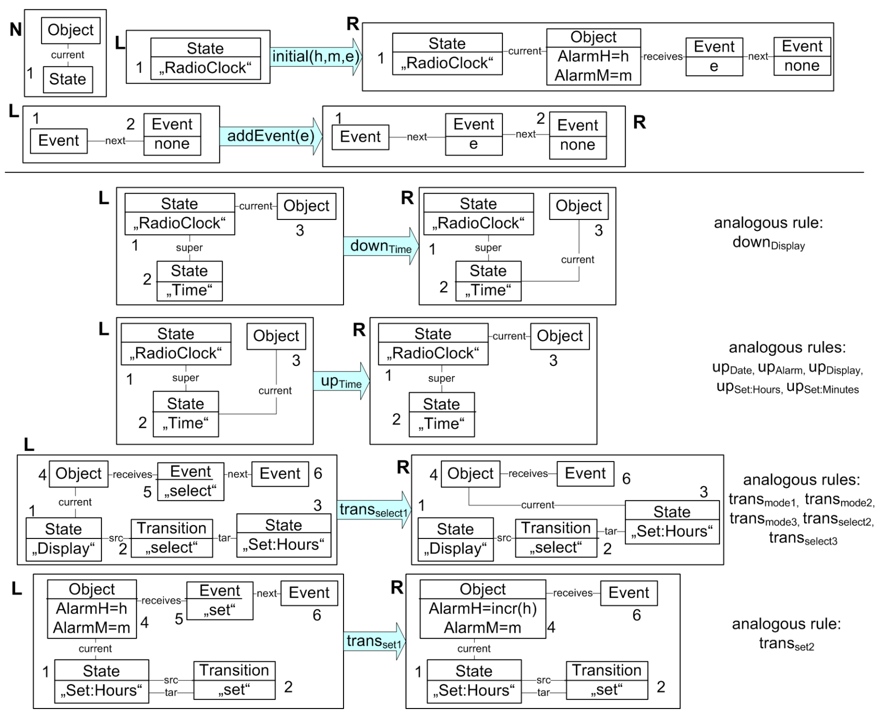

Figure 3: Simulation Rules for the Radio Clock

The simulation specification $SimSpec_{VL_S} = (VL_S, P_S)$ consists of the simulation language VL_S typed over TG_S, where TG_S is the simulation alphabet depicted in the left-hand side of Fig. 4, P_S is the set of simulation rules shown in Fig. 3, and VL_S consists of all graphs that can occur in any Radio Clock simulation scenario: $VL_S = \{G_S | \exists G_I \overset{P_S *}{\Longrightarrow} G_S\}$, where G_I is the initial graph shown in Fig. 2.

Fig. 4 shows the animation view type graph TG_A, which is a disjoint union of the simulation alphabet for Statecharts TG_S, shown in the left part of Fig. 4, and the new visualization alphabet TG_V shown in the right part of Fig. 4 which models the elements for a domain-specific visualization of the radio clock behavior. (Since we do not need the concrete syntax of the Statecharts to define the animation view, we do not depict it in Fig. 4.)

The three modes of the clock are visualized by five different displays: a date display, show-

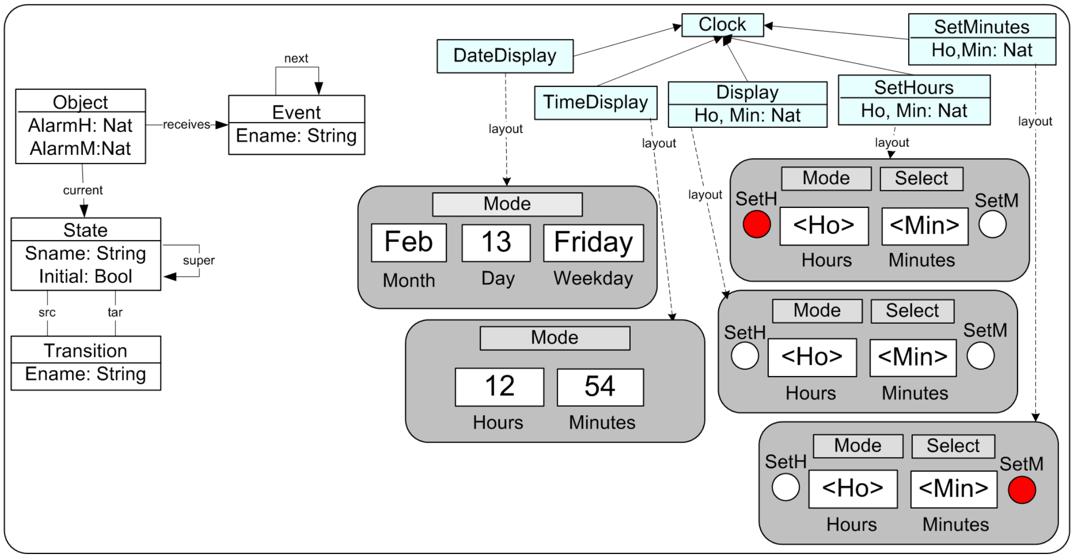

Figure 4: Simulation and Animation Alphabet for the Radio Clock

ing the date (month / day / weekday), a time display showing the time (hours / minutes), and three alarm displays showing the time for the alarm to ring, but differing in the states of two lights which indicate the states Display (both lights off), Set:Hours (light SetH on), and Set:Minutes (light SetM on).

The $S2A$ transformation rules Q, shown in Fig. 5, add corresponding visualization elements to the simulation rules and to the initial radio clock graph, depending on the state the current pointer is pointing at. We visualize only basic states which do not have any substates. Superstates (i.e. the states Radio Clock and Alarm are not visualized in the animation view, as they are considered as transient, abstract states which are active on the way of the current pointer up and down the state hierarchy between two basic states, but which have no concrete layout themselves.

The $S2A$ rule clock initializes the animation view part by adding a Clock symbol to all (rule) graphs it is applied to. This rule belongs to $S2A$ rule layer 0. For simplicity, we do not visualize the real lapse of time, and show just constants for time and date of the clock. To the Clock symbol, all generated animation view elements are linked. $S2A$ rules time and date generate the time and date displays the corresponding active state named “Time” or “Date”, respectively. $S2A$ rules display, setH and setM generate the different alarm displays, where the numbers of hours and minutes to be shown in the respective display positions are the current values of the corresponding Object attributes.

All Radio Clock $S2A$ transformation rules are typed over TG_A, and have a negative application condition NAC which equals its RHS denoted by R and N in Fig. 5. Moreover, all rules within the same rule layer are parallel independent, as none of them generates elements which are forbidden by the NACs of the other rules in the layer.

The Radio Clock $S2AM$ transformation $S2AM : VL_S \rightarrow VL_A$ is given by $S2AM = (VL_S, TG_A, Q)$ with animation language $VL_A = \{G_A | \exists G_S \in VL_S : G_S \overset{Q\ !}{\Longrightarrow} G_A\}$.

We consider a sample $S2A$ transformation sequence which transforms the simulation rule

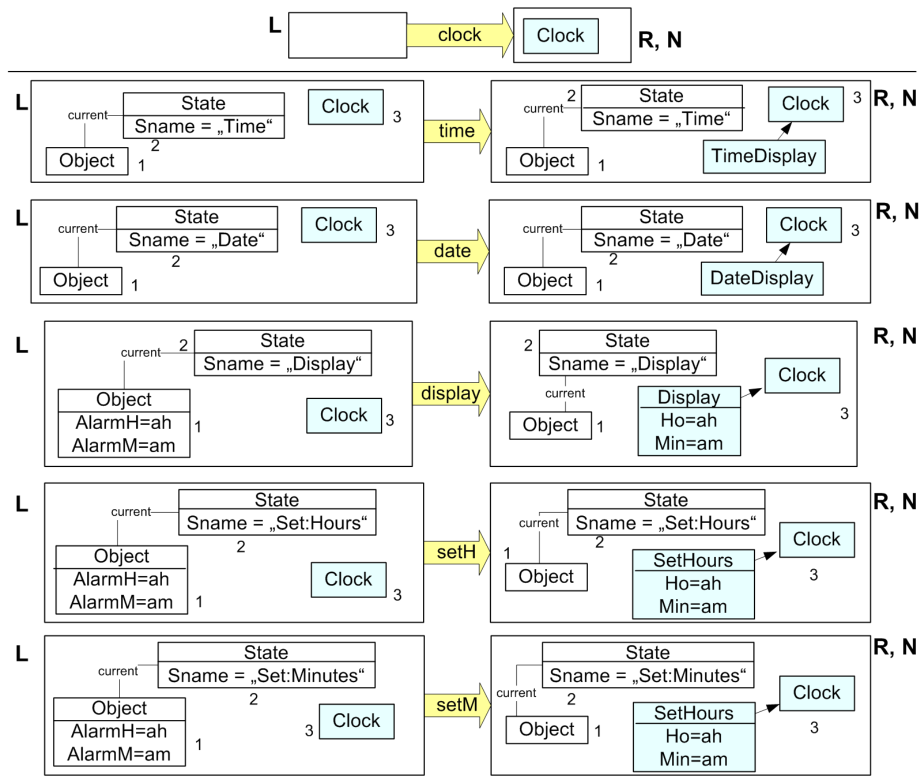

Figure 5: $S2A$ Rules for the Radio Clock

up$_{Time}$ in Fig. 3 to the animation rule S2A(up$_{Time}$) in Fig.7 using $S2A$ rules in Fig.5. In the first $S2A$ transformation step, only the $S2A$ rule clock is applicable. It is applied according to Case (1) of Def. 2.3 to all three rule graphs of rule up$_{Time}$, which results in an intermediate rule up$'_{Time}$ in Fig.6.

Secondly, rule time from $S2A$ rule layer 2 is applicable to rule up$'_{Time}$, according to Case (2), as there is a match to the LHS of rule up$'_{Time}$. The application adds a symbol of type TimeDisplay to the LHS graph, and links it to the Clock symbol. This transformation step is depicted in Fig. 6, resulting in the animation rule $S2A(up_{Time}) = (L'' \leftarrow I'' \rightarrow R'')$, since no more $S2A$ rules can be applied to this rule.

The Radio Clock $S2AR$ transformation $S2AR : P_S \rightarrow P_A$ is given by $S2AR = (P_S, TG_A, Q)$ with animation rules $P_A = \{p_A | \exists p_S \in P_S :\ p_S \overset{Q\ !}{\Longrightarrow} p_A\ \}$. The Radio Clock animation specification $AnimSpec_{VL_A}$ based on the $S2A$ transformation $S2A = (S2AM, S2AR)$ is given by $AnimSpec_{VL_A} = (VL_A, P_A)$, where VL_A is the animation language obtained by the Radio Clock $S2AM$ transformation, and P_A are the animation rules obtained by the Radio Clock $S2AR$ transformation of the simulation rules P_S. Fig. 7 shows some of the animation rules which we obtain by $S2A$ transformation applying the $S2A$ rules in Fig. 5 to the simulation rules in Fig. 3.

Fig. 8 shows an animation scenario in the concrete notation of the animation view, where the animation rules are applied beginning with the start graph $S2AM(G_I)$.

The first state of the scenario in Fig. 8 is obtained by applying animation rules from the first

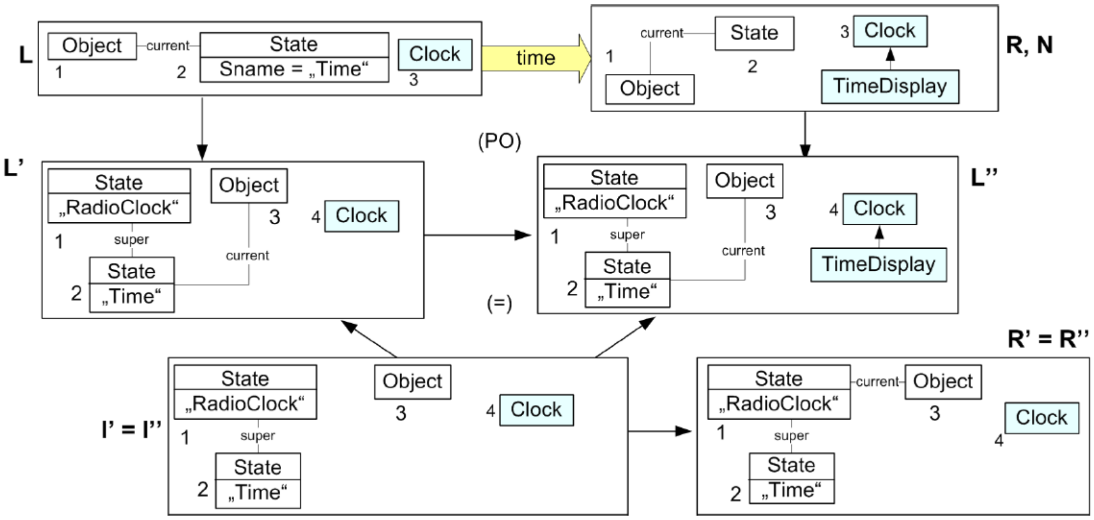

Figure 6: Application of $S2A$ Rule $time = (L \rightarrow R)$ to Rule $up'_{Time} = (L' \leftarrow I' \rightarrow R')$ resulting in Animation Rule $S2A(up_{Time}) = (L'' \leftarrow I'' \rightarrow R'')$

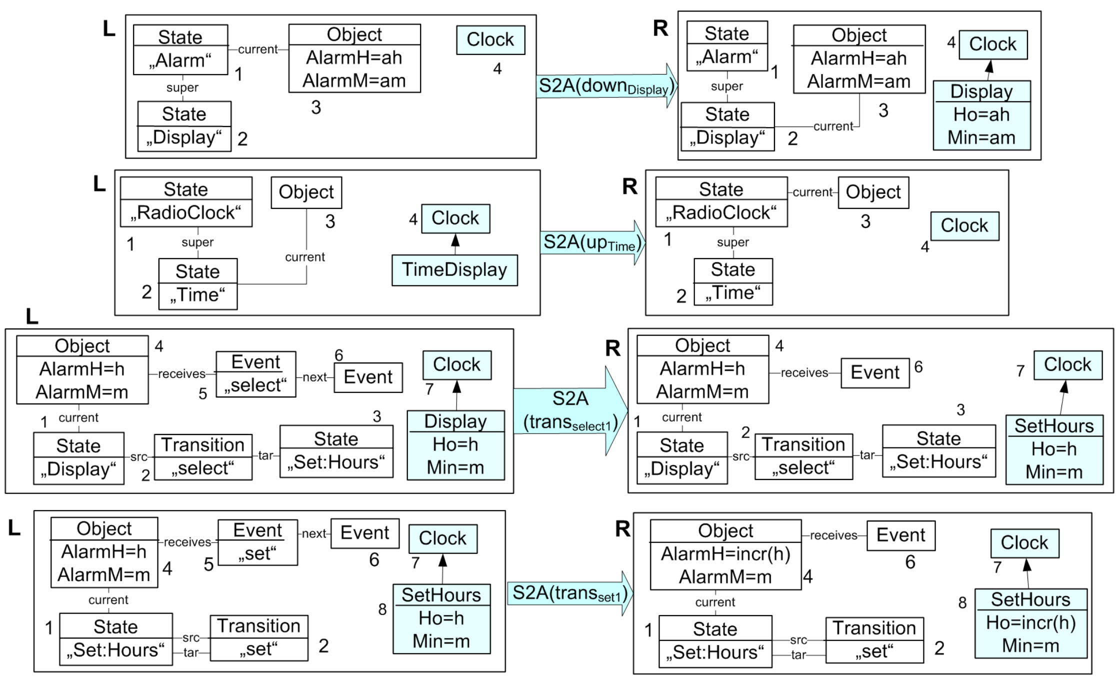

Figure 7: Animation Rules for the Radio Clock

rule layer for setting the alarm time and initializing the event queue with the events mode, mode, select, set, mode. The subsequent animation steps result from applying animation rules from the second rule layer for event processing or for moving up and down the state hierarchy.

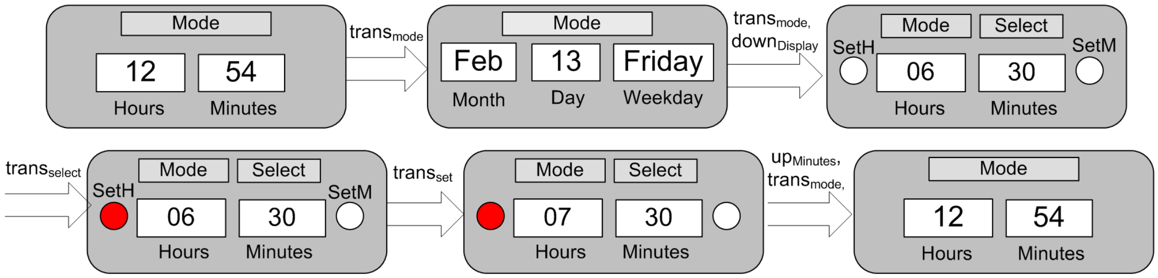

Figure 8: Animation Scenario of the Radio Clock Model

4 Semantical Correctness of $S2A$ Transformations

In this section, we continue the general theory of Section 2 and study semantical correctness of $S2A$-transformations. In our case, semantical correctness of an $S2A$-transformation means that for each simulation step $G_S \stackrel{p_S}{\Longrightarrow} H_S$ there is a corresponding animation step $G_A \stackrel{p_A}{\Longrightarrow} H_A$ where G_A (resp. H_A) are obtained by $S2A$ model transformation from G_S (resp. H_S), and p_A by $S2A$ rule transformation from p_S. Note that this is a special case of semantical correctness defined in [EE05a], where instead of a single step $G_A \stackrel{p_A}{\Longrightarrow} H_A$ more general sequences $G_A \stackrel{*}{\Longrightarrow} H_A$ and $H_S \stackrel{*}{\Longrightarrow} H_A$ are allowed.

Definition 4.1 (*Semantical Correctness of $S2A$ Transformations*)
An $S2A$-transformation $S2A : SimSpec_{VL_S} \to AnimSpec_{VL_A}$ given by $S2A = (S2AM : VL_S \to VL_A, S2AR : P_S \to P_A)$ is called *semantically correct*, if for each simulation step $G_S \stackrel{p_S}{\Longrightarrow} H_S$ with

$G_S \in VL_S$ and each $S2AR$-transformation sequence $p_S \stackrel{Q\ !}{\Longrightarrow} p_A$ (see Def. 2.4) we have

1. $S2AM$-transformation sequences $G_S \stackrel{Q\ !}{\Longrightarrow} G_A$ and $H_S \stackrel{Q\ !}{\Longrightarrow} H_A$, and
2. an animation step $G_A \stackrel{p_A}{\Longrightarrow} H_A$

$$\begin{tikzcd}
G_S \arrow[rr, Rightarrow, "Q\ !"] \arrow[dd, Rightarrow] & & G_A \arrow[dd, Rightarrow] \\
& p_S \arrow[r, Rightarrow, "Q\ !"] & p_A \\
H_S \arrow[rr, Rightarrow, "Q\ !"] & & H_A
\end{tikzcd}$$

△

Before we prove semantical correctness in Theorem 4.4, we first show local semantical correctness in Theorem 4.2 where only one $S2AM$-step (resp. $S2AR$-step) is considered.

Theorem 4.2 (*Local Semantical Correctness of $S2A$-Transformations*)
Given an $S2A$-transformation $S2A : SimSpec_{VL_S} \to AnimSpecVL_A$ with $S2A = (S2AM : VL_S \to VL_A, S2AR : P_S \to P_A)$ and an $S2AR$-transformation sequence $p_S \stackrel{Q\ !}{\Longrightarrow} p_A$ with intermediate $S2AR$-step $p_i \stackrel{q}{\Longrightarrow} p_{i+1}$ with $q \in Q$. Then for each graph transformation step $G_i \stackrel{p_i}{\Longrightarrow} H_i$ with $G_i, H_i \in \mathbf{Graphs_{TG_A}}$ we have

1. Graph transformation steps $G_i \overset{q_i}{\Longrightarrow} G_{i+1}$ in Cases (1) and (2), $G_i \overset{id}{\Longrightarrow} G_{i+1}$ in Case (3), $H_i \overset{q}{\Longrightarrow} H_{i+1}$ in Cases (1) and (3), and $H_i \overset{id}{\Longrightarrow} H_{i+1}$ in Case (2) of Def. 2.3.

2. Graph transformation step $G_{i+1} \overset{p_{i+1}}{\Longrightarrow} H_{i+1}$ with G_{i+1}, $H_{i+1} \in$ $\mathbf{Graphs_{TG_A}}$

$$\begin{tikzcd}
G_i \arrow[rr, Rightarrow, "q\ /\ id"] \arrow[dd, Rightarrow] & & G_{i+1} \arrow[dd, Rightarrow] \\
& p_i \arrow[r, Rightarrow, "q"] & p_{i+1} \\
H_i \arrow[rr, Rightarrow, "q\ /\ id"] & & H_{i+1}
\end{tikzcd}$$

△

Proof: We consider the respective pushout diagrams for $p_i \overset{q}{\Longrightarrow} p_{i+1}$ according to the three rule transformation cases in Def. 2.3, and show by pushout composition/decomposition that in each case we obtain the commuting double cube below where the two back squares comprise the given DPO for the transformation step $G_i \overset{p_i}{\Longrightarrow} H_i$, and in the front squares we get the required DPO for the transformation step $G_{i+1} \overset{p_{i+1}}{\Longrightarrow} H_{i+1}$.

In Case (1) of Def. 2.3, we obtain the top squares as pushouts and then construct G_{i+1}, C_{i+1}, H_{i+1} as pushouts in the diagonal squares, leading to unique induced morphisms $C_{i+1} \to G_{i+1}$ and $C_{i+1} \to H_{i+1}$ s.t. the double cube commutes. By pushout composition/decomposition also the front and the bottom squares are pushouts. Furthermore, we obtain pushouts for the transformation steps $G_i \overset{q}{\Longrightarrow} G_{i+1}$ and $H_i \overset{q}{\Longrightarrow} H_{i+1}$ by composing pushout (PO_I) below with the respective pushouts from the double cube.

Cases (2) and (3) are handled similarly, with the difference that some morphisms in the respective double cubes are identities.

$$\begin{tikzcd}
& L_i \arrow[dl] \arrow[dd, "m_i"] & & I_i \arrow[ll, "l_i"'] \arrow[rr, "r_i"] \arrow[dl] \arrow[dd] & & R_i \arrow[dl] \arrow[dd] \\
L_{i+1} \arrow[dd, "m_{i+1}"'] & & I_{i+1} \arrow[ll] \arrow[rr] \arrow[dd] & & R_{i+1} \arrow[dd] & \\
& G_i \arrow[dl] & & C_i \arrow[ll] \arrow[rr] \arrow[dl] & & H_i \arrow[dl] \\
G_{i+1} & & C_{i+1} \arrow[ll] \arrow[rr] & & H_{i+1} &
\end{tikzcd}
\qquad
\begin{tikzcd}
L_q \arrow[r, "q"] \arrow[d, "h"'] \arrow[dr, phantom, "(PO_I)"] & R_q \arrow[d] \\
I_i \arrow[r, "q_{i+1}"] & I_{i+1}
\end{tikzcd}$$

□

The following notions are used for proving the main Theorem 4.4.

Definition 4.3 (*Termination of $S2AM$ and Rule Compatibility of $S2A$*)
An $S2AM$ transformation $S2AM : VL_S \to VL_A$ is *terminating* if each transformation $G_S \overset{Q\ *}{\Longrightarrow} G_n$ can be extended to $G_S \overset{Q\ *}{\Longrightarrow} G_n \overset{*}{\Longrightarrow} G_m$ such that no $q \in Q$ is applicable to G_m anymore.

An $S2A$-transformation $S2A = (S2AM : VL_S \to VL_A, S2AR : P_S \to P_A)$ with $S2AM = (VL_S, TG_A, Q)$ is called *rule compatible*, if for all $p_A \in P_A$ and $q \in Q$ we have that p_A and q are parallel and sequential independent.
More precisely for each $G \overset{p_A}{\Longrightarrow} H$ with $G_S \overset{Q\ *}{\Longrightarrow} G$ and $H_S \overset{Q\ *}{\Longrightarrow} H$ for some G_S, $H_S \in VL_S$ and each $G \overset{q}{\Longrightarrow} G'$ (resp. $H \overset{q}{\Longrightarrow} H'$) we have parallel (resp. sequential) independence of $G \overset{p_A}{\Longrightarrow} H$ and $G \overset{q}{\Longrightarrow} G'$ (resp. $H \overset{q}{\Longrightarrow} H'$). △

Theorem 4.4 (*Semantical Correctness of* $S2A$)
Each $S2A$ transformation $S2A = (S2AM, S2AR)$ is semantically correct, provided that $S2A$ is rule compatible, and $S2AM$ is terminating. $\triangle$

Proof: Given $S2A = (S2AM : VL_S \to VL_A, S2AR : P_S \to P_A)$ with terminating $S2AM = (VL_S, TG_A, Q)$, a simulation step $G_S \overset{p_S}{\Longrightarrow} H_S$ with $G_S \in VL_S$, and an $S2AR$ transformation sequence $p_S \overset{Q\ !}{\Longrightarrow} p_A$ with
$p_S = p_0 \overset{q_0}{\Longrightarrow} p_1 \overset{q_1}{\Longrightarrow} \cdot\cdot \overset{q_{n-1}}{\Longrightarrow} p_n = p_A$ with $n \geq 1$, then we can apply the Local Semantical Correctness Theorem 4.2 for $i = 0, .., n-1$, leading to the following diagram

$$
\begin{tikzcd}
G_S = G_0 \arrow[r, Rightarrow, "q_0"] \arrow[dd, Rightarrow] & G_1 \arrow[r, Rightarrow, "q_1"] \arrow[dd, Rightarrow, "p_1"] & G_2 \arrow[r, Rightarrow, "q_2"] \arrow[dd, Rightarrow, "p_2"] & \cdots \arrow[r, Rightarrow] & G_{n-1} \arrow[r, Rightarrow, "q_{n-1}"] \arrow[dd, Rightarrow, "p_{n-1}"] & G_n \arrow[dd, Rightarrow] \\
p_S=p_0 \arrow[rrrrr, Rightarrow, "Q!"] & & & & & p_n=p_A \\
H_S = H_0 \arrow[r, Rightarrow, "q_0"'] & H_1 \arrow[r, Rightarrow, "q_1"'] & H_2 \arrow[r, Rightarrow, "q_2"'] & \cdots \arrow[r, Rightarrow] & H_{n-1} \arrow[r, Rightarrow, "q_{n-1}"'] & H_n
\end{tikzcd}
$$

which includes the case $n = 0$ with $G_S = G_0, H_S = H_0$ and $p_S = p_0 = p_A$, where no $q \in Q$ can be applied to $p_S = p_0 = p_A$. If no $q \in Q$ can be applied to G_n and H_n anymore, we are ready, because the top sequence is $G_S \overset{Q\ !}{\Longrightarrow} G_n = G_A$, and the bottom sequence is $H_S \overset{Q\ !}{\Longrightarrow} H_n = H_A$.

Now assume that we have $q_n \in Q$ which is applicable to G_n leading to $G_n \overset{q_n}{\Longrightarrow} G_{n+1}$. Then, rule compatibility implies parallel independence with $G_A \overset{p_A}{\Longrightarrow} H_A$, and the Local Church Rosser Theorem [EEPT06] leads to square (n):

$$
\begin{tikzcd}
G_n \arrow[r, Rightarrow, "q_n"] \arrow[d, Rightarrow, "p_A"] \arrow[dr, phantom, "(n)"] & G_{n+1} \arrow[r, Rightarrow] \arrow[d, Rightarrow, "p_A"] & \cdots \arrow[r, Rightarrow] & G_{m-1} \arrow[r, Rightarrow, "q_{m-1}"'] \arrow[d, Rightarrow, "p_A"] & G_m = G_A \arrow[d, Rightarrow, "p_A"] \\
H_n \arrow[r, Rightarrow, "q_n"] & H_{n+1} \arrow[r, Rightarrow] & \cdots \arrow[r, Rightarrow] & H_{m-1} \arrow[r, Rightarrow, "q_{m-1}"] & H_m = H_A
\end{tikzcd}
$$

This procedure can be repeated as long as rules $q_i \in Q$ are applicable to G_i for $i \geq n$. Since the $S2AM$ transformation is terminating, we have some $m > n$ such that no $q \in Q$ is applicable to G_m anymore, leading to a sequence $G_S = G_0 \overset{Q\ !}{\Longrightarrow} G_m = G_A$.

Now assume that there is some $q \in Q$ which is still applicable to H_m leading to $H_m \overset{q}{\Longrightarrow} H_{m+1}$. Now rule compatibility implies sequential independence of $G_m \overset{p_A}{\Longrightarrow} H_m \overset{q}{\Longrightarrow} H_{m+1}$. In this case, the Local Church Rosser Theorem would lead to a sequence $G_m \overset{q}{\Longrightarrow} G_{m+1} \overset{p_A}{\Longrightarrow} H_{m+1}$ which contradicts the fact that no $q \in Q$ is applicable to G_m anymore. This implies that also $H_0 \overset{Q\ *}{\Longrightarrow} H_n \overset{Q\ *}{\Longrightarrow} H_m$ is terminating, leading to the required sequence $H_S = H_0 \overset{Q\ !}{\Longrightarrow} H_m = H_A$. $\square$

5 Extension by Negative Application Conditions

In Sections 2 and 4 we have only considered the basic case of rules without negative application conditions (NACs). In this section, we extend the general theory by considering all rules and transformation of rules with NACs.

Definition 5.1 (*Transformation of Rules with NACs*)

Given a non-deleting rule $q = L_q \to R_q$ with $NAC_q = (L_q \xrightarrow{n} N_q)$ and a rule $p_1 = (L_1 \xleftarrow{l_1} I_1 \xrightarrow{r_1} R_1)$ with $NAC_{1i} = (L_1 \xrightarrow{n_{1i}} N_{1i})(i = 1, .., n)$ then q is appicable to p_1 leading to a *rule transformation step with NACs* $p_1 \overset{q}{\Longrightarrow} p_2$, if the precondition of one of the following four cases is satisfied and $p_2 = (L_2 \xleftarrow{l_2} I_2 \xrightarrow{r_2} R_2)$ with $NAC_{2i} = (L_2 \xrightarrow{n_{2i}} N_{2i})(i = 1, .., n)$ is defined according to the corresponding construction:

- *Case (1)*
  Precondition (1): There is a match $h : L_q \to I_1$ such that the matches h, $l_1 \circ h$, $r_1 \circ h$, and $n_{1i} \circ l_1 \circ h$ satisfy NAC_q for $i = 1, .., n$.
  Construction (1): As before (without $NACs$), where now (N_{2i}, n_{2i}) is defined by the following pushout

$$\begin{tikzcd} L_1 \arrow[r, "q_L"] \arrow[d, "n_{1i}"'] & L_2 \arrow[d, "n_{2i}"'] \\ N_{1i} \arrow[r, "q_{N_i}"] & N_{2i} \end{tikzcd}$$

- *Case (2)*
  Precondition (2): Precondition (1) is not satisfied, but there is a match $h' : L_q \to L_1$ such that h' and $n_{1i} \circ h'$ satisfy NAC_q for $i = 1, .., n$.
  Construction (2): As before, where now (N_{2i}, n_{2i}) is defined by pushout

$$\begin{tikzcd} L_1 \arrow[r, "q_L"] \arrow[d, "h"'] & L_2 \arrow[d] \\ N_{1i} \arrow[r, "q_{N_i}"] & N_{2i} \end{tikzcd}$$

- *Case (3)*
  Precondition (3): Preconditions (1)–(2) are not satisfied, but there is a match $h'' : L_q \to R_1$ which satisfies NAC_q.
  Construction (3): As before, where now $(N_{2i}, n_{2i}) = (N_{1i}, n_{1i})$ for $i = 1, .., n$.

- *Case (4)*
  Precondition (4): Preconditions (1)–(3) are not satisfied, but there are matches $h_i''' : L_q \to N_{1i}$ which satisfy NAC_q for $i = 1, .., n$.
  Construction (4): Let (N_{2i}, n_{2i}) be defined by PO

$$\begin{tikzcd} L_q \arrow[r, "q"] \arrow[d, "h_i'''"'] & R_q \arrow[d] \\ N_{1i} \arrow[r, "q_{N_i}"] & N_{2i} \end{tikzcd}$$

and $n_{2i} = q_{N_i} \circ n_{1i}$ $(i = 1, .., n)$. Moreover let $p_2 = p_1$.

$\triangle$

Now we are able to extend the concepts of simulation and animation specifications and $S2A$ transformations in Section 2 including semantical correctness in Section 4 to the case with NACs.

Definition 5.2 (*Animation Specification and $S2AM$-Transformation with NACs*)

Given a simulation specification $SimSpec_{VL_S} = (VL_S, P_S)$ with $NACs$ for P_S, an $S2AM$-transformation $S2AM : VL_S \to VL_A$ given by $S2AM = (VL_S, TG_A, Q)$ with $NACs$ for Q and a corresponding $S2AR$-transformation $S2AR : P_S \to P_A$ based on transformation of rules with NACs (see Definition 5.1), then

1. $AnimSpec_{VL_A} = (VL_A, P_A)$ is called *animation specification with NACs*
2. $S2A : SimSpec_{VL_S} \to AnimSpec_{VL_A}$, defined by $S2A = (S2AM, S2AR)$ is called *$S2A$-transformation with NACs*.

$\triangle$

Definition 5.3 (*Semantical Correctness of $S2A$-Transformations with NACs*)

An $S2A$-transformation $S2A : SimSpec_{VL_S} \to AnimSpecVL_A$ with NACs given by $S2A = (S2AM : VL_S \to VL_A, S2AR : P_S \to P_A)$ and NACs for P_S and P_A is called *semantically correct*, if for each simulation step $G_S \overset{p_S}{\Longrightarrow} H_S$ with $G_S \in VL_S$ and each $S2AR$-transformation sequence $p_S \overset{Q\ !}{\Longrightarrow} p_A$ with NACs for Q we have

1. $S2AM$-transformation sequences $G_S \overset{Q\ !}{\Longrightarrow} G_A$ and $H_S \overset{Q\ !}{\Longrightarrow} H_A$, and an
2. Animation step $G_A \overset{p_A}{\Longrightarrow} H_A$

$$\begin{tikzcd} G_S \arrow[rr, Rightarrow, "Q\ !"] \arrow[dd, Rightarrow] & & G_A \arrow[dd, Rightarrow] \\ p_S \arrow[rr, Rightarrow, "Q\ !"] & & p_A \\ H_S \arrow[rr, Rightarrow, "Q\ !"] & & H_A \end{tikzcd}$$

△

In order to show semantical correctness of $S2A$ transformations with NACs in Theorem 5.6, we need local semantical correctness which requires NAC-compatibility of $S2A$ in the following sense:

Definition 5.4 (*NAC-Compatibility of $S2A$*)

An $S2A$-transformation $S2A = (S2AM : VL_S \rightarrow VL_A, S2AR : P_S \rightarrow P_A)$ with $NACs$ and $S2AM = (VL_S, TG_A, Q)$ is called *NAC-compatible*, if the following conditions hold for all $q \in Q$ and $G_i \overset{p_i}{\Longrightarrow} H_i$ derivable with $NACs$ from some $G_S \overset{p_S}{\Longrightarrow} H_S$ by $S2A$:

1. (NAC-compatibility of $S2AM$)
   If q is applicable to p_i with NAC_q, then each match of q in G_i (resp. H_i) satisfies NAC_q.
2. (NAC-compatibility of $S2AR$)
   If $p_i \overset{q}{\Longrightarrow} p_{i+1}$ satisfies NAC_q, and $G_i \overset{p_i}{\Longrightarrow} H_i$ satisfies $NAC(p_i)$ then $G_{i+1} \overset{p_{i+1}}{\Longrightarrow} H_{i+1}$ satisfies $NAC(p_{i+1})$.

△

In the following proposition we state that each $S2AR$ transformation $S2AR : P_S \rightarrow P_A$ with $S2AR = (P_S, TG_A, Q)$ is NAC-compatible provided that we have a suitable layered graph transformation system as in our case study. Thus, given a concrete $S2A$ transformation, it suffices to show only NAC-compatibility of $S2AM$, where general criteria are still missing.

Proposition 5.5 (*NAC-compatibility of $S2AR$*)
Each $S2AR$-transformation $S2AR : P_S \rightarrow P_A$ with $S2AR = (P_S, TG_A, Q)$ is *NAC-compatible* in the sense of Def. 5.4, 2. △

Proof Sketch: We know that p_i satisfies NAC_i. This means, there does not exist an injective graph morphism $x : N_i \rightarrow G_i$ with $x \circ n_i = m_i$. We must show that then there does not exist an injective graph morphism $x' : N_{i+1} \rightarrow G_{i+1}$ with $x' \circ n_{i+1} = m_{i+1}$.

We assume that there exists such an injective graph morphism $x' : N_{i+1} \rightarrow G_{i+1}$ with $x' \circ n_{i+1} = m_{i+1}$. Then we have the situation depicted in the diagram below. If we can show that now we get an injective graph morphism $x : N_i \rightarrow G_i$ with $x \circ n_i = m_i$, we have a contradiction to the precondition.

$$\begin{tikzcd}
 & N_{i+1} \arrow[dr, "x'"] & L_{i+1} \arrow[l, "n_A"'] \arrow[d, "m_A"] & \\
N_i \arrow[ur, Rightarrow, "Q", "!"'] \arrow[dr, dotted, "x"'] & L_i \arrow[l, "n_S"'] \arrow[ur, Rightarrow, "!", "Q"'] \arrow[d, "m_S"] & G_{i+1} & \\
 & G_i \arrow[ur, Rightarrow, "!", "Q"'] & &
\end{tikzcd}$$

We can show the existence of such an injective morphism $x : N_i \rightarrow G_i$ for all four cases for the transformation of rules with NACs given in Def. 5.1. In the complete proof (see [Erm06]), we use pushout composition/decomposition properties, and the characteristics of a right adjoint functor $f^{<}_{TG_S} : TG_A \rightarrow TG_S$ which models the restriction from graphs typed over the animation alphabet TG_A to graphs typed over the simulation alphabet TG_S, i.e. $f^{<}_{TG_S}(G_A) = G_A|_{TG_S}$. Moreover, we use the fact that $S2A$ rules are *type-increasing* by definition, which means that we have n rule layers and type graph inclusions $TG_S = TG_0 \subseteq TG_1.. \subseteq TG_n = TG_A$ such that for each $S2A$ rule $L_q \xrightarrow{q} R_q$ belonging to layer i, L_q is typed over TG_i, R_q is typed over TG_{i+1}, and $R_q|_{TG_i} = L_q$.

□

Similar to Theorem 4.4, we also need rule compatibility where Def. 4.3 has to be extended to the case with NACs. This means that in addition to parallel and sequential independence in the case without NACs, we have to require that the induced matches satisfy the corresponding NACs.

Theorem 5.6 (*Semantical Correctness of $S2A$-Transformations with NACs*)

Each $S2A$-transformation $S2A = (S2AM, S2AR)$ is semantically correct including $NACs$, provided that $S2AM$ is terminating and $S2A$ is rule compatible and NAC-compatible (see Def. 5.4). △

Proof: Local semantical correctness in Theorem 4.2 can be extended to local semantical correctness with NACs using NAC-compatibility of $S2A$. This allows to extend also Theorem 4.4 to the case with NACs, where now rule compatibility (parallel and sequential independences) and termination have to be required with NACs. Termination of $S2AM$ holds in general, as it has been shown in [Erm06]. □

6 Semantical Correctness of the Radio Clock Case Study

In this section we show the semantical correctness of our case study.

To ensure the semantical correctness $S2A$ transformation of the Radio Clock $S2A = (S2AM, S2AR)$, we have to check transformation NAC-compatibility of the $S2A$-transformation. Since NAC-compatibility of $S2AR$ holds in general see Proposition 5.5), it suffices to show NAC-compatibility of $S2AM$ and the rule compatibility of $S2A$.

Proposition 6.1 (*NAC-Compatibility of Radio Clock $S2AM$-Transformation*)
The Radio Clock $S2AM$ transformation is NAC-compatible according to Def. 5.4, 1. △

Proof: We have to show that for all $p_i \overset{q}{\Longrightarrow} p_{i+1}$ with $q = (L_q \xrightarrow{q} R_q)$ and $NAC_q = (L_q \xrightarrow{q} R_q)$ such that the match from q to p_i satisfies NAC_q, the following $S2AM$ steps also satisfy NAC_q according to the rule transformation cases below:

Case (1): $G_i \stackrel{q}{\Longrightarrow} G_{i+1}$ and $H_i \stackrel{q}{\Longrightarrow} H_{i+1}$,
Case (2): $G_i \stackrel{q}{\Longrightarrow} G_{i+1}$,
Case (3): $H_i \stackrel{q}{\Longrightarrow} H_{i+1}$.

We show for all $q \in Q$ that for a match $L_q \to X$ there is no NAC-morphism $(R_q - L_q) \xrightarrow{x} X$. Due the property of all $q \in Q$ being type-increasing, and due to $TG_V \cap TG_S = \emptyset$, only in this case NAC_q is satisfied for this match.

The only $S2A$ rule which can be applied to any rule p_i according to Case (1) is rule clock. As rule clock belongs to rule layer 1, all rules p_i it can be applied to, are the original simulation rules, and do not contain symbols typed over TG_V. Hence, a step involving the application of clock to a rule p_i is always NAC_q-compatible, since

- the match $L_q \xrightarrow{h} I_{p_i} \xrightarrow{l_{p_i}} L_{p_i} \xrightarrow{m_{p_i}} G_i$ satisfies NAC_q as G_i does not contain TG_V-typed elements, and hence there is no NAC-morphism $(R_q - L_q) \xrightarrow{x} G_i$;

- the match $L_q \xrightarrow{h} I_{p_i} \xrightarrow{r_{p_i}} R_{p_i} \xrightarrow{m^*_{p_i}} H_i$ satisfies NAC_q as H_i does not contain TG_V-typed elements, and hence there is no NAC-morphism $(R_q - L_q) \xrightarrow{x} H_i$;

All subsequent $S2A$ transformation steps are either according to Case (2) or to Case (3). Note that the right-hand sides of all $S2A$ rules do not overlap in their generated elements, i.e. in $(R_q - L_q)$.

Let us consider a step according to Case (2), first: We assume that q is applicable to p_i, i.e. there is a match $L_q \xrightarrow{h} L_i$ satisfying NAC_q. Now, if NAC_q is not satisfied for the match $L_q \xrightarrow{h} L_i \xrightarrow{m} G_i$, then this means that q must have been applied before to another rule p_j according to Case (2) with $p_j \stackrel{q}{\Longrightarrow} p_{j+1} \Longrightarrow \cdots \Longrightarrow p_i$ with $j < i$, since q is the only $S2A$ rule which could add the elements $(R_q - L_q)$ to G_i. But in this case, we have a NAC-morphism $(R_q - L_q) \to L_{j+1} \to L_i$ which is a contradiction to our assumption that NAC_q is satisfied for the match $L_q \to L_i$. Hence, NAC_q must be satisfied for the match $L_q \xrightarrow{h} L_i \xrightarrow{m} G_i$.

Analogously, we can argue for the Case (3) steps: We assume that q is applicable to p_i, i.e. there is a match $L_q \xrightarrow{h} R_i$ satisfying NAC_q. Now, if NAC_q is not satisfied for the match $L_q \xrightarrow{h} R_i \xrightarrow{m^*} H_i$, then this means that q must have been applied before to another rule p_j according to Case (3) with $p_j \stackrel{q}{\Longrightarrow} p_{j+1} \Longrightarrow \cdots \Longrightarrow p_i$ with $j < i$, since q is the only $S2A$ rule which could add the elements $(R_q - L_q)$ to H_i. But in this case, we have a NAC-morphism $(R_q - L_q) \to R_{j+1} \to R_i$ which is a contradiction to our assumption that NAC_q is satisfied for the match $L_q \to R_i$. Hence, NAC_q must be satisfied for the match $L_q \xrightarrow{h} R_i \xrightarrow{m^*} H_i$. $\square$

Proposition 6.2 (*Rule Compatibility of the Radio Clock $S2A$ Transformation*)
The Radio Clock $S2A$ transformation is rule compatible in the sense of Def. 4.3, i.e. all p_A and all q are parallel and sequential independent.

$\triangle$

Proof: If p_A is applicable to a graph G, then there is a match $L_A \xrightarrow{m} G$. Therefore, symbols of at least those types from TG_V that are contained in L_A have also to be contained in G. So, in the sequence $G_S \overset{Q*}{\Longrightarrow} G$ there have been applied at least those rules $q \in Q$ which have also been applied in $p_S \overset{Q\ !}{\Longrightarrow} p_A$ according to Case (1) or (2) (i.e. applied to some L_i, $i = 0, .., n$). All those rules q are not applicable anymore to L_A because of their NACs NAC_q. Neither are they applicable to G, due to NAC-compatibility.

So we have to consider only those overlappings L_A/L_q where $h(L_q)$ is not completely included in $m(L_A)$. As the LHS of $S2A$ rule clock is empty, we do not have to consider this rule at all. Moreover, there exists exactly one instance of type Clock in each graph G, in all L_q and in all L_A. Hence, all pairs L_A/L_q overlap in the Clock symbol. This is uncritical, as the Clock symbol is always preserved by all rules. Furthermore, there exists always only one State symbol with a certain name. So all pairs L_A/L_q which both contain a State symbol with the same name, overlap in this node. Again, this is uncritical, as State symbols are always preserved by all rules. The next symbol and link L_A/L_q could overlap at, is a symbol of type Object and a link of type current. The Object symbol is the last node apart from the Clock and State nodes in the LHSs of the $S2A$ transformation rules. As we have argued above, L_A and L_q overlap already in the Clock and State nodes. If they would overlap also in the Object node and the current link, they would overlap completely, i.e. $h(L_q)$ would be included in $m(L_A)$. This cannot be the case as shown above. Hence, L_A and L_q do not overlap in the Object node. As there is exactly one Object node and one current link in any graph G, we can conclude that there are no pairs L_A/L_q which do not overlap completely, and in these cases NAC_q forbids the application of q.

Hence, all pairs p_A/q are parallel independent.

Due to the Local Church Rosser Theorem, we know that if $G \overset{p_A}{\Longrightarrow} H$ and $G \overset{q}{\Longrightarrow} G'$ are parallel independent (which was shown above), then $G \overset{p_A}{\Longrightarrow} H$ and $H \overset{q}{\Longrightarrow} H'$ are sequential independent. $\square$

Theorem 6.3 (*Semantical Correctness of the Radio Clock $S2A$ Transformation*)
The $S2A$ transformation $S2A = (S2AM, S2AR)$ based on the $S2A$ transformation system (TG_A, Q) for the Radio Clock model, where the $S2A$ transformation rules Q are shown in Fig. 5, is semantically correct. $\triangle$

Proof: Termination has been shown to be fulfilled in [Erm06] for general $S2A$ transformation systems with suitable rule layers by applying the termination criteria given in [EEdL+05]. Moreover, the Radio Clock $S2A$ transformation is rule-compatible (see Proposition 6.2) and NAC-compatible, where NAC-compatibility of $S2AM$ is shown explicitly (see Proposition 6.1), and NAC-compatibility of $S2AR$ has been shown for general $S2A$ transformation systems with suitable rule layers in Proposition 5.5.

Altogether this implies semantical correctness due to Theorem 5.6. $\square$

7 Conclusion and Ongoing Work

In this paper we have given a precise definition for simulation-to-animation ($S2A$) model and rule transformations. The main results show under which conditions an $S2A$ transformation $S2A : SimSpec_{VL_S} \rightarrow AnimSpec_{VL_A}$ between a simulation and an animation specification is semantically correct in the cases without and with negative application conditions. The results have been used to show semantical correctness of our radio clock case study.

For simplicity, the theory has been presented in the DPO-approach for typed graphs, but it can also be extended to typed attributed graphs, where injective graph morphisms are replaced by suitable classes M and M' of typed attributed graph morphisms for rules and negative application conditions, respectively (see [EEPT06]).

Moreover, it is interesting to analyse not only semantical correctness of $S2A : SimSpec_{VL_S} \rightarrow AnimSpec_{VL_A}$, but to construct also a backward model and rule transformation $A2S : AnimSpec_{VL_A} \rightarrow SimSpec_{VL_S}$, essentially given by restriction of all graphs and rules to the type graph TG_S. Semantical correctness of $A2S$ means that for each animation step $G_A \stackrel{p_A}{\Longrightarrow} H_A$ there is also a corresponding simulation step $G_S \stackrel{p_S}{\Longrightarrow} H_S$ using the restrictions G_S, H_S and p_S of G_A, H_A and p_A, respectively. Finally, we can consider semantical equivalence of $SimSpec_{VL_S}$ and $AnimSpec_{VL_A}$, which requires existence and semantical correctness of $S2A$ and $A2S$, such that both are inverse to each other, i.e.

$$A2S \circ S2A = Id \text{ and } S2A \circ A2S = Id.$$

References

[EB04] C. Ermel and R. Bardohl. Scenario Animation for Visual Behavior Models: A Generic Approach. *Software and System Modeling: Special Section on Graph Transformations and Visual Modeling Techniques*, 3(2):164–177, 2004.

[EE05a] H. Ehrig and K. Ehrig. Overview of Formal Concepts for Model Transformations based on Typed Attributed Graph Transformation. In *Proc. International Workshop on Graph and Model Transformation (GraMoT'05)*, ENTCS, Tallinn, Estonia, September 2005. Elsevier Science.

[EE05b] C. Ermel and K. Ehrig. View transformation in visual environments applied to Petri nets. In G. Rozenberg, H. Ehrig, and J. Padberg, editors, *Proc. Workshop on Petri Nets and Graph Transformation (PNGT), Satellite Event of ICGT'04*, volume 127(2) of *ENTCS*, pages 61–86. Elsevier Science, 2005.

[EEdL+05] H. Ehrig, K. Ehrig, J. de Lara, G. Taentzer, D. Varró, and S. Varró-Gyapay. Termination criteria for model transformation. In M. Wermelinger and T. Margaria-Steffen, editors, *Proc. Fundamental Approaches to Software Engineering (FASE)*, volume 2984 of *LNCS*, pages 214–228. Springer, 2005.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs in Theoretical Computer Science. Springer, 2006.

[EHKZ05] C. Ermel, K. Hölscher, S. Kuske, and P. Ziemann. Animated Simulation of Integrated UML Behavioral Models based on Graph Transformation. In M. Erwig and A. Schürr, editors, *Proc. IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*, Dallas, Texas, USA, September 2005. IEEE Computer Society.

[Erm06] C. Ermel. Simulation and Animation of Visual Languages based on Typed Algebraic Graph Transformation. PhD Thesis, TU Berlin, submitted, 2006.

[Gen] GenGED Homepage. http://tfs.cs.tu-berlin.de/genged.

[Gra99] Bernd Grahlmann. The State of PEP. In M. Haeberer A. editor, *Proceedings of AMAST'98 (Algebraic Methodology and Software Technology)*, volume 1548 of *Lecture Notes in Computer Science*. Springer-Verlag, January 1999.

[Har87] D. Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[HEC03] D. Harel, S. Efroni, and I.R. Cohen. Reactive Animation. In F.S. de Boer, M.M. Bonsangue, and S. Graf et al., editors, *Proc. First Int. Symposium on Formal Methods for Components and Objects (FMCO'02)*, volume 2852 of *LNCS*, pages 136–153. Springer, 2003.

[HM03] D. Harel and R. Marelly. *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer, 2003.

[Mac04] Macromedia, Inc. *Macromedia Flash MX 2004 and Director MX 2004*, 2004. http://www.macromedia.com/software/.

[PP96] F. Parisi-Presicce. Transformation of Graph Grammars. In *5th Int. Workshop on Graph Grammars and their Application to Computer Science*, volume 1073 of *LNCS*. Springer, 1996.

[Rha05] I-Logix. *Rhapsody: Model-Driven Development with UML 2.0, SysML and Beyond*, 2005. http://www.ilogix.com/rhapsody/rhapsody.cfm.

[Roz97] G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. World Scientific, 1997.

[WWW03] WWW Consortium (W3C). *Scalable Vector Graphics (SVG) 1.1 Specification.*, 2003. http://www.w3.org/TR/svg11/.