Modeling Multicasting in Communication Spaces by Reconfigurable High-level Petri Nets*

Claudia Ermel, Enrico Biermann, Hartmut Ehrig, Kathrin Hoffmann, Tony Modica
Technische Universität Berlin
Fakultät IV, Sekr. FR 6-1
Franklinstr. 28/29, 10587 Berlin, Germany
{enrico, ehrig, lieske, hoffmann, modica}@cs.tu-berlin.de

Abstract

In mobile and adaptive communication systems, communicating entities (actors) can transmit content, which is contextually interpreted. Actors may join, move in or leave so-called communication spaces, where the actors' preferences, access rights and roles are respected and define a temporary set of communicating partners and a context of interpretation for communicated data. An adequate modeling technique for communication spaces should take into account the changing communication relations between actors.

Conventional modeling techniques for communication-based systems like Petri nets or UML are restricted to model communication based on a static, immutable network topology. In our research project "Formal modeling and analysis of flexible processes in mobile ad-hoc networks", we have proposed an appropriate integration of Petri nets and Petri net transformation rules, based on graph transformation, leading to a visual formal modeling technique, called reconfigurable Petri nets. *In this paper, we extend this previous work on reconfigurable Petri nets on the one hand by marking-changing Petri net transformations, and on the other hand by a technique parallelizing the application of net transformation rules at several matches at once. Both extensions together allow for a flexible modeling of communication concepts in communication spaces, like e.g. multicasting, where one actor transmits contents to a group of selected actors. We apply our extended technique to model multicasting group communication in the Internet telephone system Skype.*

*This work has been supported by the Integrated Graduate Program on Human-Centric Communication at TU Berlin and by the research project forMA$_1$NET (see http://tfs.cs.tu-berlin.de/formalnet/) of the German Research Council.

1. Introduction

During the last decade, mobile and adaptive communication systems have become more and more important. Communication nowadays is based on Internet platforms like Skype, Facebook, or SecondLife. Mobile Ad-hoc NETworks (MANETs) consist of mobile nodes which communicate with each other via wireless personal digital assistants (PDAs) while the topology of the network constantly changes depending on the availability of communication participants.

In both scenarios, communicating entities (actors) can transmit content (via channels) that is contextually interpreted. Actors may join, move in or leave so-called *communication spaces*, i.e. sub-systems of global communication systems, where the actors' preferences, access rights and roles are respected and define a temporary set of communicating partners and a context of interpretation for communicated data.

An adequate modeling technique for communication spaces should focus on the changing communication relations between the actors. Besides realizing the communication itself, communication spaces adapt to the environment and to changing preferences, and are able to interpret communicated content in different contexts.

Conventional modeling techniques for communication-based systems like UML [16] and actor systems [1] or formal specification techniques like process algebras [15], low-level and high-level Petri nets [18, 13], algebraic specification [10], and different kinds of logic are restricted to model communication based on a static, immutable network topology. Moreover, many of these techniques do not support visual modeling and visual behavior simulation which makes it difficult to validate more complex models.

Therefore, Petri nets as a fundamental, visual and formal model in concurrency, have recently been subject for suitable extensions – in particular to model mobility and

reconfigurations of the net structure based on graph transformation [6] – that might be profitably applied to the study of communication spaces.

In our research project forMA$_l$NET [1], we propose an appropriate integration of graph transformation, Petri nets and processes in high level and higher-order net classes. The integration is based on algebraic Petri net transformations [8], where net transformation rules are applied to adapt the net structure to changing requirements of the system (e.g. mobility), and transition firing models the behavior of the system itself. As underlying Petri nets, we use marked algebraic high-level nets (AHL systems [11]), where tokens are data elements, i.e. algebras over a signature.

In this paper, the formal basis of AHL systems and AHL system morphisms is extended in order to be able to model also a typical communication concept of communication spaces, namely *multicasting*, where one actor transmits content to a group of selected actors. Multicasting is supported by communication platforms which offer conference connections or group chats, like e.g. AppleTalk or Skype. Modeling the communication behavior of multicasting is a challenge for Petri net-based modeling techniques since the number of actors is not known a priori. A well-known but not fully adequate way is to split the transmission to a group into single transmissions to each participating actor. This makes it necessary to code the status of the multicast data for each actor into the net [14]. Furthermore, each actor must be connected to every other actor in order to be able to perform multicasting which leads to a massive explosion of the net structure size in communication nets with multicasting.

In this paper, we introduce an extension of AHL system transformations allowing us to change the structure and / or marking of a variable number of net parts in one step. To this end, we use amalgamation of net transformation rules, which realizes a controlled, parallel rule application at several matches at once. We apply this technique to model multicasting communication in the Internet telephone platform Skype.

The rest of this paper is organized as follows: We start in Section 2 giving an overview of communication spaces and stating requirements on modeling techniques for them. Section 3 reviews the concept of reconfigurable AHL nets and introduces our running example Skype, a platform supporting dynamic communication structures based on the Internet. We model the structure of actor components and establish dynamically communication settings by changing the actors' underlying control nets. In Section 4, our modeling technique is extended by parallel rule application, and we show how this extension is used to model the multicasting communication concept in Skype. We sketch the underlying formal basis of our modeling approach in Section 5, where we focus on the new category **AHLI** of AHL systems with individual tokens (as opposed to the collective token approach in [8]). The extension enables us to change the marking by rule applications, which is necessary for our approach to model multicasting communication. Section 6 ends with the conclusions and an outlook to future work.

[1] Formal modeling and analysis of flexible processes in mobile ad-hoc networks, funded by the German Research Council (DFG) (2006 - 2011), http://tfs.cs.tu-berlin.de/projekte/formalnet/.

2. Communication Spaces

The notion of *Communication Spaces* has been coined in the research area "Modeling and Engineering of Computer Supported Communication Spaces" of the recently founded Innovation Center Human-Centric Communication at Technische Universität Berlin and is an ontology for describing communication systems[2]. In communication spaces, communicating entities (actors) can transmit content (via channels) that are contextually interpreted. Actors may move within different communication spaces, they may join or leave them. Preferences, access rights, and roles are to be respected. Typical examples are Internet-based applications like Skype, Facebook, or SecondLife; and also Mobile Adhoc networks or SmartHomes, in which appliances are connected intelligently to offer increased comfort to their inhabitants.

An adequate modeling approach would have to cover at least three main aspects of communication spaces: data and knowledge of the actors (their *content spaces*), structure of the underlying communication network, i.e. which actors are connected to each other (the *topology*), and communication within and between different communication spaces (*interaction*). According to our experience, no single classic modeling approach is powerful enough to cover all these aspects.

Recently, Petri nets as a fundamental, visual and formal model in concurrency, have been subject for suitable extensions – in particular to model mobility and reconfigurations of the net structure – that might be profitably applied to the study of communication spaces. In our research project forMA$_l$NET , we propose an appropriate integration of graph transformation concepts and high-level Petri nets: algebraic Petri net transformations [8] use net transformation rules to adapt the net structure to changing requirements of the system. As underlying Petri nets, we use marked algebraic high-level nets (AHL systems [17, 11]), where tokens are data elements, i.e. algebras over a suitable signature.

As we will see in the next section's example, such *reconfigurable AHL systems* are powerful enough to cover the main aspects of communication spaces by integrating Petri

[2] see http://www.h-c3.org/ra_en.html#RAE for more information

nets (topology), abstract data types (content spaces), and net transformation (interaction). We introduce the concept of reconfigurable AHL systems informally, while the basic new formal notions are introduced in Section 5, and a formal theory is under development [7].

3. Example: Skype

In this paper, we will concentrate on the Skype platform[3] as running example, which offers many typical features of communication spaces. Skype is a widely used program for Internet telephony, offering easy to use (synchronized) data exchange and conferences. With its contact and privacy management, users can decide who and how other users can contact them. Skype is not open source, there is no (publicly available) formal model, and Skype uses proprietary network protocols. Therefore, we limit ourselves to modeling the observable behavior, i.e. to activities users can perform in their Skype client software and the direct effects of these activities caused in the Skype system. The overall aim is to use such a model as basis for analysis, e.g. to find conflicts and dependencies in system runs which help to detect errors in access right or role policies.

Our model follows these guidelines:

- An AHL system models a whole Skype communication space. Each actor (resp. his Skype client instance) is represented by a distinguished component within the AHL system.
- We strictly distinguish between actor-triggered client behavior and system reactions. An actor's actions are modeled by transitions in the corresponding actor component; an actor can act if at least one of its transitions is enabled. System reactions are modeled by rules that reconfigure the system. In more detail: An actor's action can either alter the actor's configuration directly (like (de)activating the actor, modifying privacy settings etc.) or represent a request to the Skype system to perform a global task (like establishing connections or transmitting data) that may extend/restrict possible actions of the actor. System operations executing such requests are realized by rule applications, which possibly create or remove transitions of an actor component net.
- To keep the intuitive visual representation of AHL systems for communication spaces comprehensible, we assume the system to apply cleaning-up rules on temporary net structures after the activity that they were created for has been completed. Moreover, when simulating a system, the modeler should be able to grasp the system's state very quickly.

[3] Skype is freely available at `http://www.skype.com`

For a quick introduction to reconfigurable AHL systems, we avoid the formal notation and consider a simplified visual representation: Fig. 1 shows an AHL system component, with rectangles as transitions and ovals as typed places that contain tokens (data elements). Arcs with variables connect places and transitions. The AHL system in Fig. 1 shows a part of an actor's component, called its *data unit*. The data unit net contains places *User* of type *SkypeName*, *Store* of type *Data*, *Out* of type *OutData* and *In* of type *InData*. Place *Store* contains a token "*hello*", and place *User* contains a token *Alice*, which represent values of the corresponding data types.

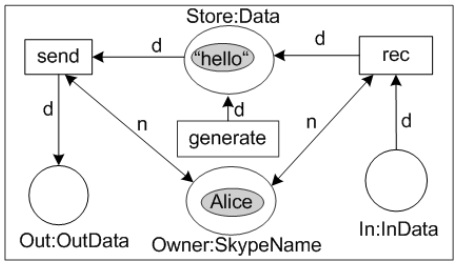

Figure 1. AHL net *Actor's data unit*

The actor is a Skype client identified by its owners's Skype name "Alice". The owning user is indicated by the *SkypeName* token on the actor's place *Owner*. Actors may send data to and receive data from other actors. The type *Data* may represent audible, textual, graphical etc. data, which a data unit can *send*, *rec*eive and *generate* via the correspondent transitions. Place *Store* is in the predomain of transition *send* because there is an arc pointing to the transition from the place. Analogously, place *Out* is in the postdomain of the same transition and *Owner* is in both the pre- and postdomain.

We call a transition enabled if for each variable of its predomain arcs we can find a suitable token on the corresponding place. In Fig. 1, transition *send* is enabled, because we can assign $d = \text{"}hello''$ and $n = Alice$. If a transition is enabled, we can fire it, which means that the predomain tokens are removed and for each variable on the postdomain arcs a token according to the variable assignment is added to the corresponding place.

Apart from having a data unit, each actor has a net component controlling its behavior.

Fig. 2 shows the AHL system modeling the control component for the Skype client of actor "Alice". Each actor control component has the following basic structure:

- The four places typed by *Control*, called Control places, represent the possible states of a Skype client. The data type *Control* consists of the unique value •, currently marking the state *Offline* in the example.

- *SkypeName* places carry identities of Skype actors, e.g. the token *Alice* on the place *User* indicates that Alice is the client's user, and the tokens on *Contacts* represent two Skype actors she has in her contact list. A *SkypeName* token on *CallRequest* would announce a request to the system for connection to the corresponding actor.

- If there is a token on the actor's place *ReadyToTalk*, the client is supposed neither to be offline nor to participate in another call/conference, hence to be able to accept incoming calls.

Possible firing step allow for activating the actor, changing the actor's state to *DoNotDisturb* or *SkypeMe*, deactivating the client by firing *deactivate*, which would delete the token from place *ReadyToTalk*, or announcing a request for a connection to another actor by firing *requestCall*.

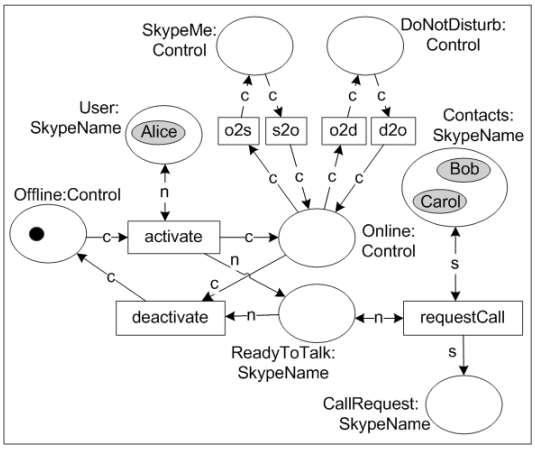

Figure 2. AHL system *Actor's control component*

In the following, we will see how the system reacts to requests by reconfiguring actor control nets to allow for more activities.

In order to model reconfiguration of AHL systems we use the rule-based Petri net transformation approach [6, 8]. An AHL system transformation rule is a span of component-wise injective AHL system morphisms $p = (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ with L, K and R being AHL systems. L and R are called left- and right-hand side of the rule, and $K = L \cap R$ is the interface of L and R (which we sometimes omit).

Example. Imagine that Alice wants to have a direct call to Bob, who is in her Skype contact list. In her control net in Fig. 2, she fires at first the *activate* transition, and afterwards the *requestCall* transition, so the token *Bob* (which of course has to be present on her *Contacts* place for this) is copied to her *CallRequest* place. To allow the system to react to the request we formulate the rule CreateConference depicted in Fig.3.

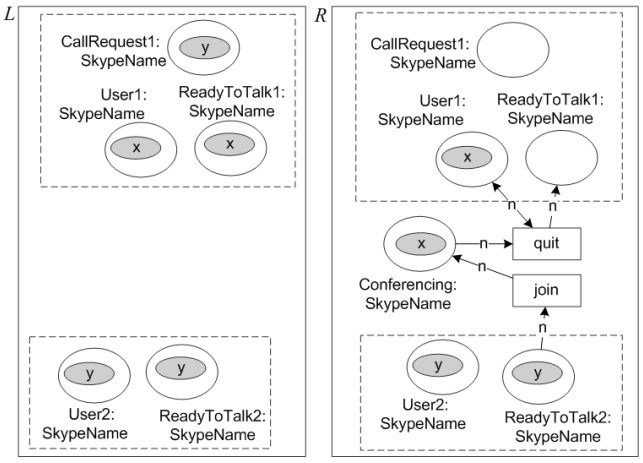

Figure 3. Example rule *CreateConference*

The three upper framed places in the left-hand side L are considered the places of the requesting actor component, whereas the lower ones belong to the responding actor component. Note that we use variables for the tokens in this general rule, so it can match only if we find the same token value (that we assign to the variable y) on the requesting actor's *CallRequest* and on the responding actor's *User* and *ReadyToTalk* places. The rule's interface K looks like L but without the tokens on *CallRequest1* and *ReadyToTalk1*. The rule creates a conferencing structure, moves the requester's *SkypeName* token to the new *Conferencing* place, and deletes his request token on *CallRequest*.

Applying a rule p means to find a pattern (called *match*) of L in the source AHL system N_1 and to replace this matched part by R, thus transforming the source AHL system into the target AHL system of the transformation. Intuitively, the application of rule $p = (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ to N_1 via a match m from L to N_1 deletes the image $m(L)$ from G and replaces it by a copy of the right-hand side $m^*(R)$.

Note that a rule may only be applied if the so-called *gluing condition* is satisfied, i.e. the deletion step must not leave *dangling edges*, and for two objects which are identified by the match, the rule must not preserve one of them and delete the other one. Moreover, we forbid the application of a rule if there are unmatched transitions connected to places that should be deleted.

If rule $p = (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ is applicable to net N_1 (i.e. the gluing conditions is satisfied), we obtain the AHL system transformation step $N_1 \stackrel{p,m}{\Longrightarrow} N_2$ from N_1 to the resulting AHL system N_2.

Example. Let us apply rule *CreateConference* (see Fig. 3), to a net where Alice has requested to talk to Bob, i.e. there

is a token *Bob* on place *CallRequest* of Alice's control net, and a token *Alice* on its place *ReadyToTalk*. In Bob's control net, we have one token *Bob* on the *User* place, and one token *Bob* on place *ReadyToTalk*. Fig. 4 shows the critical regions of Alice's and Bob's control nets before and after application of rule *CreateConference* on the obvious match m with variable assignment $x = Alice$, and $y = Bob$. The gluing condition is satisfied because the match m is injective and the rule deletes only two tokens, leaving no edges dangling. We interpret the newly created transitions as additional behavior of the participating actors. Being the host, Alice is attending the conference directly after rule application; she is unavailable to other calls while her conference is running. Hence, her token on place *ReadyToTalk* is removed. Her only option is to *quit* and terminate the whole conference. Bob may join the conference now if he likes, i.e. he can fire his new *join* transition, which would move his *SkypeName* token to *Conferencing* as well.

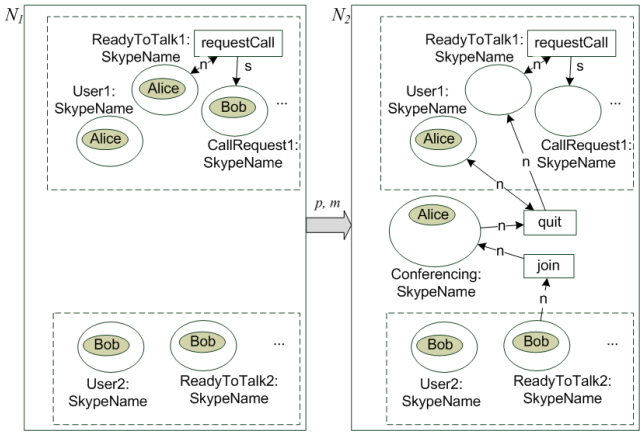

Figure 4. Applying rule *CreateConference*

4. Modeling Multicasting in Skype by Rule Amalgamation

Skype offers communication concepts like telephone conferences or group chats, the underlying principle of which is called *multicasting*. In multicasting communication, one actor transmits contents to a group of previously selected actors. Modeling the communication behavior of multicasting is a challenge for Petri net-based modeling techniques since the number of actors is not known a priori. In this section, we introduce an extension of AHL system transformations allowing us to change the structure and / or marking of a variable number of net parts in one step. To this end, we use amalgamation of net transformation rules, which realizes a controlled, parallel rule application at several matches at once.

The essence of rule amalgamation is that (possibly infinite) sets of rules which have a certain regularity, so-called rule schemes, can be described by a finite set of rules modeling the elementary actions. The application of amalgamated rules is also known as parallel graph transformation [19, 5] and provides a general concept to model parallel actions with the possibility of synchronization.

If actions are not independent of each other, they can be applied in parallel if they can be synchronized by subactions. If two actions contain the deletion or the creation of the same element, this operation can be encapsulated in a separate action which is a common subaction of the original ones. A common subaction is modelled by the application of a subrule of all original rules (called elementary rules).

Rule amalgamation is defined here by an *interaction scheme* $IS = (E, s, SE)$ consisting of a set E of AHL system rules (elementary rules), an AHL system rule s (subrule) and a set of subrule embeddings $se : s \to e$ into the elementary rules for all $e \in E$. (In general, more than one subrule can be defined for rule amalgamation, see [19].)

The application of rules synchronized by a subrule is then performed by gluing the elementary rules at their subrule which leads to the corresponding *amalgamated rule*. The application of such a rule is called *amalgamated net transformation*.

Example. In multicasting, one actor transmits contents to a group of other actors. The common subaction is that contents is sent by the sending actor. This is done only once, at one occurence in the net, independently of the number of receiving actors. The elementary actions in multicasting are the receive actions, involving as many actors as there are in the current conference together with the sender.

Hence, our interaction scheme for multicasting can be defined as follows: $IS_{multi} = (E, s, se)$ where s is the common subrule depicted at the top of Fig. 5. Note that the interface K_s equals R_s which allows marking-change between the *out1*-places from L_s to R_S, although $K_s \to L_s$ and $K_s \to R_s$ are marking-preserving AHL system morphisms. $E = \{e\}$ is the elementary rule in the bottom of Fig. 5. The subrule embedding se consists of the morphisms $L_s \to L_e$, $K_s \to K_e$ and $R_s \to R_e$ which embed the left-hand subrule side, the subrule interface and its right-hand side into the corresponding parts of the elementary rule. These subrule embedding morphisms are indicated by equal numbers of mapped elements.

In addition to the specification how elementary rules should be synchronized, we have to decide where and how often a set of elementary rules should be applied. The basic way to synchronize complex parallel operations (used in this paper) is to require that a rule should be applied at all different matches it has in a given net. This expresses massively parallel execution of actions.

Given an AHL system *N*, an interaction scheme $IS =$

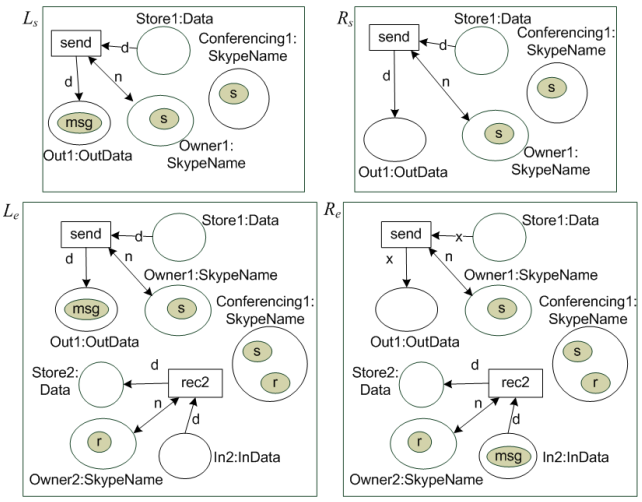

Figure 5. Interaction Scheme for Multicasting

(E, s, se) and a match $m_s : L_s \to N$ for subrule s, an amalgamated AHL system rule $L_a \xleftarrow{K}{}_a\!\!\to R_a$ is constructed by gluing the submatch $m_s : L_s \to N$ of all possible matches $m_e : L_e \to N$ from elementary rules $e \in E$, i.e. we have $m_e \circ se_L = m_s : L_s \to N$ for all $e \in E$.

Example. Fig.6 shows in the bottom the amalgamated rule p_a with L_a and R_a being the left and right-hand rule sides. Rule p_a is constructed over our interaction scheme in Fig. 5, matched to a net N with one sending component and three available receiving components which are joined in the same conference (i.e. they all have their *SkypeName* tokens on place *Conferencing1*). We have shaded the corresponding subnets in similar shades to clarify how the subrule and the elementary rules are merged in the amalgamated rule.

This amalgamated rule should be applied to a net at exactly the matches of its subrule and elementary rules which were used for its construction. The effect of rule application is that the *msg* token from actor 1 (the sender) is removed from its *OutData* place *Out1* and added to all *InData* places of the actors participating in the conference (the receivers).

5. Towards Formal Modeling of Multicasting in Communication Spaces

In this section, we present the first, basic steps towards an underlying formal framework for our modeling technique.

We use AHL system transformation with rule amalgamation. As we have seen in our multicasting example, we needed rules which were able to change the marking on preserved places. Up to now, tokens on places have been treated as indistinguishable entities (called *collective token approach*). As a consequence, morphisms on AHL systems

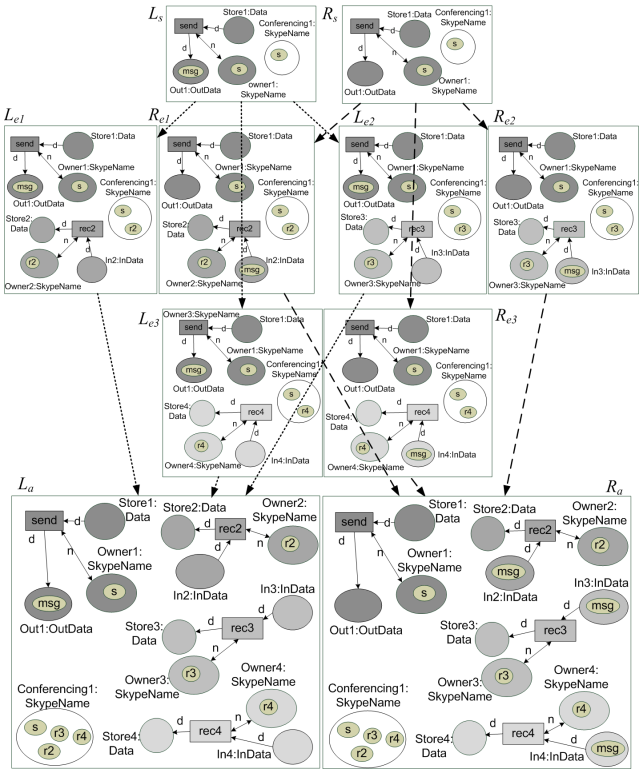

Figure 6. Amalgamated Rule for Multicasting with four participants

have been marking-strict, i.e. they did not allow the change the number of tokens on places that were preserved by the rule [8, 9]. To overcome this limitation, we are currently elaborating a transformation theory of nets with individual tokens, where tokens become distinguishable [7]. A related approach has been followed by Montanari et al. in [4], but there token individuals were needed only to specify process semantics and not yet on the level of nets and net morphisms. Our new notion of AHL systems, called AHLI nets (AHL systems with individual tokens) allows for rule-based changes of markings and hence provide the formal background needed to model multicasting by rule amalgamation as described in Section 4.

In the following, we formalize AHLI net transformation. Here is the full definition of AHLI systems and AHLI system morphisms:

Definition 1 (Algebraic High-Level Net System with Individual Tokens)**.** An algebraic high-level net system with individual tokens (AHLI) is given by

$$N = (\Sigma, P, T, pre, post, cond, type, A, I, m) \quad \text{with}$$

a signature $\Sigma = (S, OP, X)$, sets P of places and T of transitions, pre and post conditions $pre, post : T \to$

$(T_\Sigma(X) \otimes P)^\oplus$, firing conditions for the transitions $cond : T \to \mathcal{P}_{fin}(Eqns(\Sigma; X))$, a function $type : P \to S$, typing the places over the signature sorts, a Σ-algebra A a (possibly infinite) set I of individual tokens, and a marking function $m : I \to A \otimes P$, assigning the individual tokens to data elements on the places.

$T_\Sigma(X)$ is the set of terms over the algebra's signature Σ with variables in X. $T_\Sigma(X) \otimes P$ are the pairs $(term, p)$, such that $term$ is of the sort $type(p)$, similiar for $A \otimes P$. $\mathcal{P}_{fin}(Eqns(\Sigma; X))$ are the finite sets of equations over Σ-terms with variables in X.

Remark. Each AHLI net with individual token marking (I, m) can be interpreted as an AHL system with collective token marking $M = \sum_{(a,p)\in A\otimes P} |m^{-1}(a, p)|(a, p)$ where $|m^{-1}(a, p)| = |\{i \in I | m(i) = (a, p)\}|$ denotes the number of individual tokens with marking (a, p). The main difference is that we can distinguish tokens of the same algebraic value on the same place in AHLI nets.

Definition 2 (AHLI Net System Morphisms)**.** Given two AHL system systems $N_i = (\Sigma, P_i, T_i, pre_i, post_i, cond_i, type_i, A, I_i, m_i)$ for $i \in \{1, 2\}$ with the same signature Σ and algebra A, an AHL system system morphism is given by a triple of functions

$$f = (f_P : P_1 \to P_2, f_T : T_1 \to T_2, f_I : I_1 \to I_2) : N_1 \to N_2$$

such that the following properties hold: $type_2 \circ f_P = type_1$ (f_P preserves place types), $cond_2 \circ f_T = cond_1$ (f_T preserves transition conditions), $pre_2 \circ f_T = (id_{T_\Sigma(X)} \otimes f_P)^\oplus \circ pre_1$, analogously for $post_i$ (f preserves the pre and post conditions of transitions), $m_2 \circ f_I = (id_A \otimes f_P) \circ m_1$ (f preserves the individual tokens' values and locations)[4].

The main idea of AHLI system transformation (in analogy to algebraic graph transformation [6]) is the rule-based modification of AHLI systems where each application of a transformation rule leads to a transformation step, changing the AHLI system.

Definition 3 (AHLI transformation rule)**.** An AHLI transformation rule is a span of component-wise injective AHLI morphisms $p = (L \xleftarrow{l} K \xrightarrow{r} R)$. L and R are called left- and right-hand side of the rule, and $K = L \cap R$ is the interface of L and R.

Definition 4 (AHLI System Transformation)**.** Given an AHLI transformation rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ as defined above and an AHLI system N_1 with a general AHLI morphism[4] $m : L \to N_1$, called match. Rule p is called applicable at match m if the gluing condition is satisfied for l and m. In this case, we obtain an AHLI system N_0 leading to an AHLI system transformation step $N_1 \overset{p,m}{\Longrightarrow} N_2$ from N_1 to the AHLI system N_2, consisting of the following pushout diagrams (1) and (2) in the category **AHLI**. Informally, a pushout in a category **CAT** is a gluing construction of two objects over a specific interface.

$$\begin{array}{ccccc} L & \xleftarrow{l} & K & \xrightarrow{r} & R \\ {\scriptstyle m}\downarrow & (1) & \downarrow{\scriptstyle k} & (2) & \downarrow{\scriptstyle m^*} \\ N_1 & \xleftarrow{} & N_0 & \xrightarrow{} & N_2 \end{array}$$

Combining AHLI systems with transformation rules, we get the required integrated approach for reconfigurable AHLI systems:

Definition 5 (Reconfigurable AHLI Systems with Amalgamation)**.** A reconfigurable AHLI system with amalgamation is a triple $(N, \mathcal{R}, \mathcal{IS})$, where an AHLI-system N is the start configuration of the system, $\mathcal{R}$ is a set of transformation rules, and $\mathcal{IS}$ is a set of interaction schemes for rule amalgamation.

Using the amalgamation concept, we rely on the formalization of interaction schemes as synchronization mechanism, together with the definition of subrules and elementary rules as shown in Section 4. In [3], the amalgamation theorem for graph transformation states that applying an amalgamated rule of elementary rules $p1$ and $p2$ has essentially the same effect as applying first the common part of $p1$ and $p2$ (the subrule) and then the remainders of $p1$ and $p2$. The amalgamation construction and theorem have been shown to be valid also for attributed graph transformation [12]. It remains open to transfer these concepts to AHLI system transformation.

Summarizing, for our multicasting model, we rely on a reconfigurable AHLI system $(N, \mathcal{R}, \mathcal{IS})$ with the following features:

- Actors in this system are modeled as arbitrary (independent) components in N.
- $\mathcal{R}$ and $\mathcal{IS}$ may contain rules and schemes to create, delete, and manipulate these components dynamically.
- $\mathcal{IS}$ contains the interaction scheme for multicasting, thus allowing to construct a suitable amalgamated rule to perform the multicasting of a token *msg* to all participants in a group communication.

6. Conclusion and Future Work

In this paper, we use amalgamated transformation rules for AHLI systems to model dynamic systems capable of multicasting data between components. In contrast to usual

[4]Morphisms can also be formulated for the general case that we have morphisms on the signature and algebra parts, as well.

approaches, we do not realize the distribution of data tokens by (temporary) transitions but by net transformation. For this, we exploit that AHLI rules can change the marking of AHLI nets.

We validated our approach by modeling group communication in Skype, where multicasting in communication spaces is used frequently. Our approach aims to keep the visualization of communication spaces small and understandable. Hence, structures for communication are added to the system only in case when they are needed and are removed after the communication action has been completed. For modeling multicasting, we refrain from adding a massive connective net structure, connecting the sender with each of the receiver components. Instead, we define an interactions scheme allowing to find all receivers with the appropriate access rights in a possibly large system net. A suitable amalgamated rule, which can be constructed automatically, then carries out the multitasking action in one single transformation step.

We have indicated our first ideas towards a formal foundation of our approach, based on the notion of AHLI systems and amalgamation. The basic concepts of the corresponding theory will be treated in our technical report [7].

We have implemented an Eclipse-based tool environment for reconfigurable Petri nets, which currently allows modeling, simulation and analysis of reconfigurable P/T nets[5] [2]. An extension of our tool to AHLI systems is planned for the future.

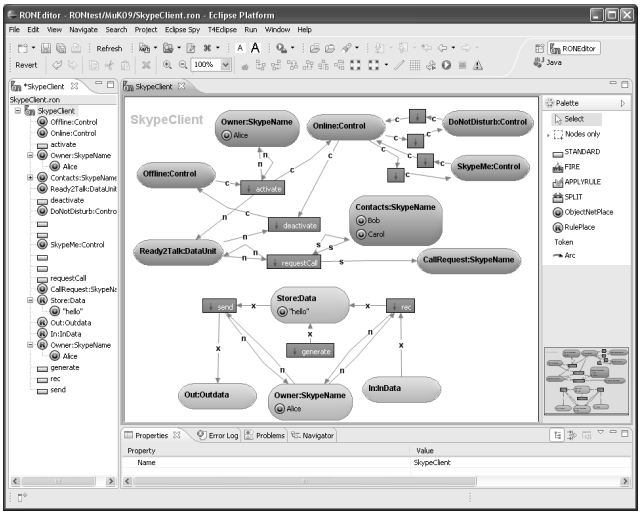

Figure 7. Tool Environment for Reconfigurable Petri Nets

[5] http://www.tfs.cs.tu-berlin.de/roneditor

References

[1] G. Agha. *ACTORS: A Model of Concurrent Computation in Distributed Systems*. PhD thesis, MIT, 1985. Cambridge: MIT Press.

[2] E. Biermann, C. Ermel, F. Hermann, and T. Modica. A Visual Editor for Reconfigurable Object Nets based on the ECLIPSE Graphical Editor Framework. In *Proc. Workshop on Algorithms and Tools for Petri Nets (AWPN)*. Gesellschaft für Informatik, 2007.

[3] P. Böhm, H.-R. Fonio, and A. Habel. Amalgamation of graph transformations: a synchronization mechanism. *Journal of Computer and System Science*, 34:377–408, 1987.

[4] R. Bruni, J. Meseguer, U. Montanari, and V. Sassone. Functorial Semantics for Petri Nets under the Individual Token Philosophy. In *Proc. Conf. on Category Theory and Comp. Science*, volume 29 of *ENTCS*, pages 1–19. Elsevier, 1999.

[5] J. de Lara, C. Ermel, G. Taentzer, and K. Ehrig. Parallel Graph Transformation for Model Simulation applied to Timed Transition Petri Nets. In *Proc. Workshop on Graph Transformation and Visual Modelling Techniques (GTVMT)*, pages 17–29, 2004.

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

[7] H. Ehrig, C. Ermel, K. Hoffmann, T. Modica, and E. Biermann. Low and High-Level Petri Nets with Individual Tokens. Technical report, Technische Universität Berlin, 2009. to appear.

[8] H. Ehrig, K. Hoffmann, J. Padberg, C. Ermel, U. Prange, E. Biermann, and T. Modica. Petri Net Transformations. In *Petri Net Theory and Applications*, pages 1–16. I-Tech Education and Publication, 2008.

[9] H. Ehrig, K. Hoffmann, J. Padberg, U. Prange, and C. Ermel. Independence of net transformations and token firing in reconfigurable place/transition systems. In *Proc. Conference on Application and Theory of Petri Nets and Other Models of Concurrency (ATPN)*, volume 4546 of *LNCS*, pages 104–123. Springer, 2007.

[10] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, volume 6 of *EATCS Monographs on Theor. Comp. Science*. Springer, 1985.

[11] H. Ehrig, J. Padberg, and L. Ribeiro. Algebraic High-Level Nets: Petri Nets Revisited. In *Proc. Workshop on Specification of Abstract Data Types*, volume 785 of *LNCS*, pages 188–206. Springer, 1994.

[12] R. Heckel, J. Müller, G. Taentzer, and A. Wagner. Attributed graph transformations with controlled application of rules. In *Proc. Colloquium on Graph Transformation and its Application in Computer Science*, pages 41–53, 1995.

[13] K. Jensen and G. Rozenberg, editors. *High-Level Petri Nets*. Springer, 1991.

[14] J.-K. Lee and K.-H. Lee. Modeling of the Multicast Transport Protocols using Petri Nets. In *Proc. Conf. on Communications and Networks.*, pages 106–110, 1995.

[15] R. Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press, 1999.

[16] Object Management Group. *Unified Modeling Language: Superstructure – Version 2.1.1*, 2007. formal/07-02-05, `http://www.omg.org/technology/documents/formal/uml.htm`.

[17] U. Prange. Towards algebraic high-level systems as weak adhesive HLR categories. In *Proc. Workshop on Applied and Computational Category Theory (ACCAT)*, volume 203 / 6 of *ENTCS*, pages 67–88. Elsevier, 2008.

[18] W. Reisig. *Petri Nets: An Introduction*, volume 4 of *EATCS Monographs on Theor. Comp. Science*. Springer, 1985.

[19] G. Taentzer. *Parallel and Distributed Graph Transformation: Formal Description and Application to Communication-Based Systems*. PhD thesis, Technische Universität Berlin, 1996. Shaker Verlag.