Simulating Algebraic High-Level Nets by Parallel Attributed Graph Transformation: Long Version

Claudia Ermel, Gabriele Taentzer, and Roswitha Bardohl

Technische Universität Berlin, Germany
Intern. Center for Computer Science, Schloss Dagstuhl, Germany
`lieske|gabi@cs.tu-berlin.de, rosi@dagstuhl.de`

Abstract. The "classical" approach to represent Petri nets by graph transformation systems is to translate each transition of a specific Petri net to a graph rule (behavior rule). This translation depends on a concrete model and may yield large graph transformation systems as the number of rules depends directly on the number of transitions in the net. Hence, the aim of this paper is to define the behavior of Algebraic High-Level nets, a high-level Petri net variant, by a parallel, typed, attributed graph transformation system. Such a general parallel transformation system for AHL nets replaces the translation of transitions of specific AHL nets. After reviewing the formal definitions of AHL nets and parallel attributed graph transformation, we formalize the classical translation from AHL nets to graph transformation systems and prove the correctness of the translation. The translation approach then is contrasted to a definition for AHL net behavior based on parallel graph transformation. We show that the resulting amalgamated rules correspond to the behavior rules from the classical translation approach.

1 Introduction

Visual modeling languages (like the Unified Modeling Language (UML), Petri nets, Statecharts, and many more) play a central role for software and system modeling. Visual models are used for system design, simulation, validation, and code generation. Apart from developing visual models, the simulation of a model on the basis of a formal specification is an important issue for testing and validating the system behavior. The simulation of Petri nets, for example, is realized by playing the token game: a transition can fire if it is enabled, a firing step removes tokens from the transition's predomain places and adds tokens to its postdomain places.

Petri net behavior can be defined as graph transformation system where each transition is translated to a graph rule modeling the corresponding change of the marking (deleting and/or adding tokens) in a firing step [15, 3]. This "classical" way to define Petri net behavior by graph transformation assumes a specific Petri net before compiling its transitions into graph rules (*compiler* approach).

Yet, for related visual behavior modeling languages, it is often possible to define a general graph transformation system which is independent of a specific model and can be used to interpret arbitrary models of a visual language (*interpreter* approach). An example is a graph transformation system for describing the behavior of a Statechart variant given in [2]. In general, the interpreter approach is much more flexible and scalable than the compiler approach. As it is independent of a concrete model, the graph transformation system defined for the interpreter approach is fixed once for the complete visual language, i.e. the number of behavior rules is finite and does not grow with the size of the model (scalability). In contrast, using the compiler approach, each specific model must be translated to get the model-specific graph transformation system.

Unfortunately, it is difficult to give a general graph transformation system to simulate Petri nets as there may be arbitrary many places connected to a transition, leading to an arbitrary number of behavior rules. Hence, parallel graph transformation concepts have been used to simulate the behavior of Condition-Event nets in [20] and of *Timed Transition Petri Nets* in [4].

Parallel graph transformation was introduced by Ehrig and Kreowski in [6], later generalized to parallel high-level replacement systems [11] by Ehrig and Taentzer, further elaborated and applied to communication-based systems in [20]. The essence of parallel graph transformation 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. For instance, when modeling the firing of a Petri net transition, the elementary actions would be the removal of a token from a place in the transition's predomain and the addition of a token to a postdomain place. For the description of such rule schemes the concept of amalgamating rules at subrules is used which is based on synchronization mechanisms for rules developed first in [5].

The aim of this paper is to present a formal interpreter approach to define the behavior of high-level Petri nets. A specific, well-defined variant of high-level nets are Algebraic High-Level nets, AHL nets for short, introduced by Ehrig, Padberg and Ribeiro in [18]. We present an interpreter approach for the behavior of AHL nets based on parallel attributed graph transformation. Thus, a general graph transformation system for simulating AHL nets replaces the translation of transitions of specific AHL nets. The resulting parallel behavior specification is formally proven to be semantically equivalent to the corresponding compiler approach translating each specific AHL net to a corresponding attributed graph transformation system. This compiler approach for AHL nets has been presented in [1] and is reviewed in a slightly modified form in this paper.

In Section 2, the formal definitions of AHL nets and their behavior are reviewed, using the well-known *Dining Philosophers* as running example. Section 3 presents the concepts of sequential (classical) and parallel attributed graph transformation. The concepts are the basis in Section 4 to formalize the translation from AHL nets to sequential graph transformation systems according to the compiler approach. We prove the semantical compatibility of an AHL net and its translation to a graph transformation system, i.e. we show that a firing sequence in the net corresponds to a graph transformation sequence in the translated graph transformation system. The compiler approach is contrasted by the interpreter approach based on parallel attributed graph transformation, in Section 5. An interaction scheme is presented specifying the elementary actions when simulating an AHL net. From this scheme, amalgamated rules are defined for AHL nets, and it is proven that these rules correspond semantically to the behavior rules of the sequential graph transformation system given in Section 4. The conclusion (Section 6) gives an outlook on how the simulating graph transformation systems for AHL nets are used in the visual language environment GenGED for simulating and animating the behavior of AHL nets. A shorter version of this paper (without proofs) has been published in [13].

2 Algebraic High-Level Nets

An AHL net is a combination of a place/transition net [19] and an algebraic datatype specification *SPEC* describing operations used as arc inscriptions. Tokens are elements of a corresponding *SPEC*-algebra [8, 7]. In this section, we review the definition of AHL nets and their behavior as given in [18], and present our running example, the well-known *Dining Philosophers*.

In contrast to other variants of AHL nets [14, 16] we do not label places with sorts. The pre- and postdomain of a transition is given by a multiset of pairs of terms and places, i. e. as elements of a commutative monoid.

Definition 1 (Algebraic High-Level Net).
An algebraic high-level net $N = (SPEC, P, T, pre, post, cond, A)$ *consists of an algebraic specification* $SPEC = (S, OP, E; X)$ *with equations* E *and additional variables* X *over the signature* (S, OP)*, sets* P *and* T *of places and transitions respectively, pre- and postdomain functions* $pre, post : T \to (T_{OP}(X) \times P)^{\oplus}$ *assigning to each transition* $t \in T$ *the pre- and postdomains* $pre(t)$ *and* $post(t)$*, respectively, a firing condition function* $cond : T \to \mathcal{P}_{fin}(EQNS(S, OP, X))$ *assigning to each transition* $t \in T$ *a finite set* $cond(t)$ *of equations over the signature (S,OP) with variables* X*, and an* (S, OP, E)*-algebra* A*.*

Remarks

- $T_{OP}(X)$ is the set of terms with variables X over the signature (S, OP), and $M^{\oplus}$ is the free commutative monoid over a set M. Thus, $T_{OP}(X) \times P = \{(term, p) | term \in T_{OP}(X), p \in P\}$.
- The predomain function $pre(t)$ (and similar postdomain function $post(t)$) have the form $pre(t) = \sum_{i=1}^{n}(term_i, p_i)$ with $(n \geq 0)$, $p_i \in P, term_i \in T_{OP}(X)$. This means that $\{p_1, ...p_n\}$ is the predomain of t with arc-inscription $term_i$ for the arc from p_i to t if all $p_1, ..., p_n$ differ (unary case) and arc-inscription $term_{i1} \oplus ... \oplus term_{ik}$ for $p_{i1} = ... = p_{ik}$ (multi case). Note that in our sample AHL net (see Example 1) we have the multi case, but as drawing convention we draw separate arcs, each inscribed by one term only. Hence,

we allow to draw more than one arc in one direction between a place and a transition.
– AHL nets together with AHL net morphisms build a category **AHLnet** [18].

Definition 2 (Marking and Firing Behavior of AHL Nets).
Let $N = (SPEC, P, T, pre, post, cond, A)$ be an AHL net according to Def. 1.

– *A marking m is an element $m \in M^{\oplus}$ with $M = A \times P = \{(a, p) | a \in \bigcup_{s \in S} A_s, p \in P\}$*
– *Enabling and firing of transitions is defined as follows: For any $t \in T$ let $Var(t)$ be the set of local variables occurring in $pre(t), post(t)$ and $cond(t)$. An assignment $asg_A : Var(t) \to A$ is called consistent wrt. $t \in T$ if the equations $cond(t)$ are satisfied in A under asg_A. Transition t is enabled under a consistent assignment $asg_A : Var(t) \to A$ and a marking $m \in (A \times P)^{\oplus}$, if $pre_A(t, asg_A) \leq m$. The marking $pre_A(t, asg_A)$ – analogously $post_A(t, asg_A)$ – is defined for $pre(t) = \sum_{i=1}^{n}(term_i, p_i)$ by $pre_A(t, asg_A) = \sum_{i=1}^{n}(\overline{asg}_A(term_i), p_i)$, where $\overline{asg}_A : T_{OP}(Var(t)) \to A$ is the extended evaluation of terms under assignment asg_A. The successor marking m' is defined in the case of t being enabled by $m' = m \ominus pre_A(t, asg_A) \oplus post_A(t, asg_A)$ and gives raise to a* firing step *$m[t, asg_A\rangle m'$.*

Example 1 (The Dining Philosophers as AHL Net).
As example we show the AHL net for *The Dining Philosophers* in Fig. 1 (see [19, 18] for the corresponding place/transition net). We identify the five philosophers as well as their chopsticks by numbers. Fig. 1 (a) shows the initial situation where all philosophers are thinking and all chopsticks are lying on the table. Fig. 1 (b) shows the AHL net with the corresponding initial marking. For this marking, the transition take is enabled as a thinking philosopher and his left and right hand side chopsticks are available. The firing of transition take with the variable binding $p = 2$, for example, removes token 2 from place thinking and adds it to place eating, whereas tokens 2 and 3 are removed from place table, as the chopstick computing operation (p mod 5) +1 is evaluated to 3.

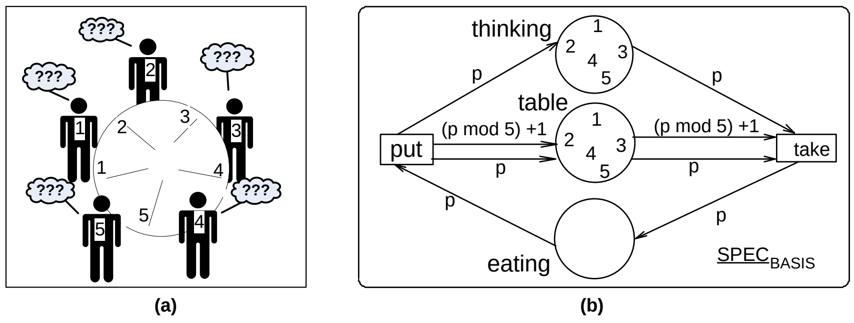

Fig. 1. The *Dining Philosophers* (a) modeled as AHL Net (b)

As datatype specification we take a basic specification for all AHL nets $SPEC_{BASIS}$ consisting of the union of specifications NAT for natural numbers, $BOOL$ for boolean operations, and $STRING$ for strings. The tokens on all places are elements of a corresponding $SPEC_{BASIS}$-algebra, i.e. natural numbers in our example. The arcs are inscribed each by one variable or term from $T_{OP}(X)$ denoting computation operations to be executed on token values.

3 Parallel Attributed Graph Transformation

3.1 Attributed graph transformation

In the following, we present attributed graph structures as defined in [9]. For graph transformations in the category of attributed graph structures and homomorphisms with a distinguished class M of morphisms, the Church-Rosser, Parallelism and Concurrency Theorem have been shown in [9].

Definition 3 (Attributed Graph Structure Signatures). *A graph structure signature $GSIG = (S_G, OP_G)$ is an algebraic signature with unary operations $op : s \to s'$ in OP_G only. An attributed graph structure signature $ASSIG = (GSIG, DSIG)$ consists of a graph structure signature $GSIG$ and a data signature $DSIG = (S_D, OP_D)$ with attribute value sorts $S'_D \subseteq S_D$ such that $S'_D = S_D \cap S_G$ and $OP_D \cap OP_G = \varnothing$.*
$ASSIG$ is called well-structured *if for each $op : s \to s'$ in OP_G we have $s \notin S_D$.*

$ASSIG$-algebras and $ASSIG$-homomorphisms build up a category [9] which is denoted by **ASSIG-Alg**. In the following, we call $ASSIG$-algebras *attributed graphs* and $ASSIG$-homomorphisms *attributed graph morphisms.*

As an example for an attributed graph structure signature we define the signature ASSIG_{AHL} for AHL nets. AHL nets are considered as ASSIG_{AHL}-algebras.

Definition 4 (Attributed Graph Structure Signature for AHL Nets).

The attributed graph structure signature for AHL nets (shown visually in Fig. 2) is given by $\mathsf{ASSIG}_{AHL} = (GSIG_{AHL}, DSIG_{AHL})$. In Fig. 2, the sorts of $GSIG_{AHL}$ are represented as nodes. The operations are the arcs between the sort nodes (the op-links between graph sorts) and from sort nodes to data nodes, (the attr-links between graph sorts and attribute sorts). The $DSIG$ part (data signature) consists of the attribute value sorts of the basic specification, i.e. $String, Nat$ and $Bool$ and their usual operations.

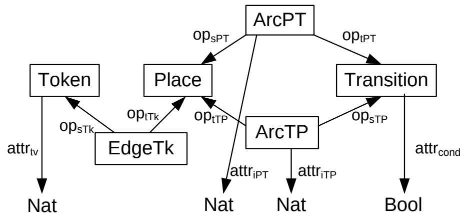

Fig. 2. Abstract Syntax Graph visualizing the ASSIG for AHL Nets

The attribute values are used for the arc inscriptions, tokens and transition firing conditions.

Next, we define the double-pushout approach to graph transformation on the basis of category **ASSIG-Alg**.

Proposition 1 (Pushouts of $ASSIG$-Homomorphisms). *Let M be a distinguished class of all homomorphisms f which is defined by $f \in M$ if f_{GSIG} is injective and $f_{DSIG} = id_{DSIG}$ for f in* **ASSIG-Alg**. *Given $f : A \to B \in M$ and $a : A \to C$ then there exists their pushout in* **ASSIG-Alg**.

Proof: See [9].

Category **ASSIG-Alg** and class M are fixed throughout this section.

Definition 5 (Typed Attributed Graph Transformation System). *A typed attributed graph transformation system $GTS = (S, P)$ based on (***ASSIG-Alg***, M) consists of an $ASSIG$-algebra S, called start graph and a set P of rules, where*

1. *a rule $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ of $ASSIG$-algebras L, I and R attributed over the term algebra $T_{DSIG}(X)$ with variable set X of variables $(X_s)_{s \in S_{DSIG}}$, called left-hand side L, interface I and right-hand side R, and homomorphisms $l, r \in M$, i.e. l and r are injective and identities on the data type $T_{DSIG}(X)$,*
2. *a direct transformation $G \overset{p,m}{\Longrightarrow} H$ via a rule p and a homomorphism $m : L \to G$, called* match, *is given by the diagram to the right, called* double-pushout diagram, *where (1) and (2) are pushouts in* **ASSIG-Alg** *(the triple (m, i, m^*) is called* rule embedding*),*

$$\begin{tikzcd}
L \arrow[d, "m"'] \arrow[dr, phantom, "(1)"] & I \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "i"] \arrow[dr, phantom, "(2)"] & R \arrow[d, "m^*"] \\
G & D \arrow[l, "g"] \arrow[r, "h"'] & H
\end{tikzcd}$$

3. *a typed attributed graph transformation, short transformation, is a sequence $G_0 \Rightarrow G_1 \Rightarrow ... \Rightarrow G_n$ of direct transformations, written $G_0 \overset{*}{\Rightarrow} G_n$,*
4. *the language $L(GTS)$ is defined by $L(GTS) = \{G \mid S \overset{*}{\Rightarrow} G\}$.*

Now we add the concept of attribute conditions.

Definition 6 (Attribute Condition). *Given a rule p attributed over the term algebra $T_{DSIG}(X)$, an attribute condition C consists of a set of equations $(a = b)$ over $T_{DSIG}(X)$. An $ASSIG$-morphism $m : L \to G$ satisfies an attribute condition C, if $m_{DSIG}(a) = m_{DSIG}(b)$ for all $(a = b) \in C$.*

Definition 7 (Conditional Rule and Transformation). *Let $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ be a rule attributed over the term algebra $T_{DSIG}(X)$, and C an attribute condition over $T_{OP}(X)$. Then, $\hat{p} = (p, C, X)$ is a conditional rule. The direct conditional transformation $G \overset{\hat{p},m}{\Longrightarrow} H$ is given by the direct transformation $G \overset{p,m}{\Longrightarrow} H$ if m satisfies C.*

A transformation sequence as well as a graph transformation system and its language based on conditional rules are defined as in Def. 5.

3.2 Parallel Graph Transformation

Parallel graph transformation in the double-pushout approach has been introduced in [20] on the basis of labeled graphs. Here, we extend the concepts to attributed graphs and rules with attribute conditions. The main idea of parallel graph transformation is to apply a number of rules in one parallel step. Their matches are allowed to overlap and can even be conflicting in the general case. Common subactions are described by subrules. Therefore, the notion of subrule embedding is basic to the whole approach.

Definition 8 (Subrule Embedding).
Given a conditional rule $\hat{p} = ((L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R), A, Y)$, *a conditional rule* $\hat{s} = ((L_s \overset{l_s}{\leftarrow} I \overset{r_s}{\rightarrow} R_s), A_s, X)$ *is called* subrule *of* $\hat{p}$ *if* $X \subseteq Y$ *and there are injective morphisms* $e : L_s \to L$, $f : I_s \to I$ *and* $g : R_s \to R$ *in* M *such that* $e \circ l_s = l \circ f$ *and* $g \circ r_s = r \circ f$, *i.e. the diagram to the right commutes.*

$$\begin{tikzcd}
L_s \arrow[d, "e"'] & I_s \arrow[l, "l_s"'] \arrow[r, "r_s"] \arrow[d, "f"] & R_s \arrow[d, "g"] \\
L & I \arrow[l, "l"] \arrow[r, "r"'] & R
\arrow[from=1-1, to=2-2, phantom, "="]
\arrow[from=1-2, to=2-3, phantom, "="]
\end{tikzcd}$$

The triple $t = (e, f, g)$ *from* $\hat{s}$ *to* $\hat{p}$ *(short* $t : \hat{s} \to \hat{p}$*) is called* subrule embedding. *In this context,* $\hat{p}$ *is called* extending rule. *Subrule embedding* t *is called* quasi-identical, *if* e, f, *and* g *are isomorphisms. In this case,* $\hat{s}$ *is called isomorphic to* $\hat{p}$. *Two subrule embeddings* $t_1 : \hat{s}_1 \to \hat{p}_1$ *and* $t_2 : \hat{s}_2 \to \hat{p}_2$ *are called* isomorphic, *if there are quasi-identical subrule embeddings from* $\hat{s}_1$ *to* $\hat{s}_2$ *and from* $\hat{p}_1$ *to* $\hat{p}_2$ *such that they commute with* t_1 *and* t_2.

All conditional rules and their subrule embeddings build up a category which we call $\mathbf{Rule_{ASSIG-Alg}}$. Three rule functors are defined to extract the LHS embeddings, the embeddings of interfaces and the RHS embeddings.

Definition 9 (Rule Functors). *The forgetful functors* V_L, V_I, V_R : $\mathbf{Rule_{ASSIG-Alg}} \to \mathbf{ASSIG-Alg}$, *called* rule functors, *are defined in the obvious way, e.g.* $V_L(\hat{p}) = V_L((L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R), C, Y) = L$.

To apply a set of rules in parallel in a synchronized way, we have to decide how and how often the rules can be applied to a host graph G. One possibility is to allow a rule to be applied at all different matches it has in G. This would result in a massively parallel application of rules which is not always wanted. To restrict the degree of parallelism, two control features are introduced: the *interaction scheme* and the *covering construction*. The interaction scheme is a set of subrule embeddings and restricts the synchronization possibilities of rule applications. The covering construction restricts the matching possibilities for the rules of the interaction scheme. One special covering construction, called *local*, allows to match a subrule s exactly once to a part $m(s)$ of G, and to match all rules extending s as often as possible to the surroundings of $m(s)$. In this way, a kernel action can be described in a variable context. Another important covering construction, called *fully synchronized* forbids conflicting rule matches, i.e. two rule matches of rules extending the same subrule s have to overlap completely at a match of their common subrule.

Formally, a covering is described by an instance interaction scheme and a set of matches. The instance interaction scheme contains the concrete number of instances of each rule in the scheme, depending on how many matches into G have been found for each rule of the interaction scheme. Thus, an interaction scheme can be seen as type information for instance interaction schemes.

Definition 10 (Interaction Scheme). *An* interaction scheme IS *consists of a set of subrule embeddings such that the following conditions hold:*

1. *for each two subrule embeddings $t_1 : \hat{s}_1 \to \hat{p}_1$ and $t_2 : \hat{s}_2 \to \hat{p}_2$ we have $\hat{s}_1 \neq \hat{s}_2$ or $\hat{p}_1 \neq \hat{p}_2$,*
2. *for each two subrule embeddings $t_1 : \hat{s} \to \hat{p}_1$ and $t_2 : \hat{s} \to \hat{p}_2$ in IS with $\hat{s} = (p_s, C_s, X)$, $\hat{p}_1 = (p_1, C_1, Y_1)$ and $\hat{p}_2 = (p_2, C_2, Y_2)$ we have $Y_1 \cap Y_2 = X$.*

IS is called local interaction scheme*, if there is one subrule $\hat{s}$ being the source of at least one subrule embedding to each extending rule.*

Definition 11 (Instance Interaction Scheme). *Given an interaction scheme IS, an interaction scheme IIS is an* instance interaction scheme *of IS, if there is a mapping $ins : IIS \to IS$ such that $\forall t \in IIS$: if there is an isomorphic subrule embedding $t \xrightarrow{\sim} u$ then $ins(t) = u$.*

Definition 12 (Covering Construction). *Let IS be an interaction scheme and G an $ASSIG$-algebra. A partial covering $COV = (IIS, MA)$ consists of an instance interaction scheme IIS of IS and a set MA of matches from all rules of all subrule embeddings in IIS to G such that they commute with the subrule embeddings, i.e. for any two subrule embeddings $t_1 : \hat{s} \to \hat{p}_1$ and $t_2 : \hat{s} \to \hat{p}_2$ in IIS there are two matches $m_s : L_s \to G$ and $m_p : L_p \to G$ in MA with $m_p \circ e = m_s$. Let $t_1 : \hat{s} \to \hat{p}_1$ and $t_2 : \hat{s} \to \hat{p}_2$ be any two subrule embeddings in IIS and $m_{p_1} : L_{p_1} \to G$ and $m_{p_2} : L_{p_2} \to G$ corresponding matches in MA.*

1. *COV is called* local*, if IIS is local, and if $\hat{p}_1$ is isomorphic to $\hat{p}_2$, then m_{p_1} has to be non-isomorphic to m_{p_2}.*
2. *COV is called* fully synchronized*, if there are two subrule embeddings $u_1 : \hat{s}' \to \hat{p}_1$ and $u_2 : \hat{s}' \to \hat{p}_2$ such that $m_{p_1}(L_{p_1}) \cap m_{p_2}(L_{p_2}) = m_{s'}(L_{s'})$.*

Since category **ASSIG-Alg** has initial objects being empty graphs attributed over $T_{DSIG}(X)$, and pushouts, it is finitely cocomplete [17], i.e. has all finite colimits. This is the basis to build the amalgamated rule of any partial covering which glues all parallel rules according to their subrule embeddings. Applying the amalgamated rule afterwards according to Def. 5 completes a parallel graph transformation step.

Definition 13 (Amalgamated Rule and Transformation). *Let G be a graph and $COV = (IIS, MA)$ be a covering construction with $IIS = \bigcup_{n \in N} (t_n : \hat{s}_n \to \hat{p}_n)$ being an instance interaction scheme with $\hat{s}_n = ((L_{s_n} \overset{l_{s_n}}{\leftarrow} I_{s_n} \overset{r_{s_n}}{\to} R_{s_n}), C_{s_n}, Y_{s_n})$ and $\hat{p}_n = ((L_n \overset{l_n}{\leftarrow} I_n \overset{r_n}{\to} R_n), C_n, Y_n)$ and $MA = \cup_{n \in N} m_n : L_n \to G$. The* amalgamated rule *$\hat{p}_{COV} = ((L \overset{l}{\leftarrow} I \overset{r}{\to} R), C, Y)$ is constructed by the following steps:*

1. *Let L be the colimit object of $\bigcup_{n\in N} V_L(t_n) : V_L(s_n) \to V_L(p_n)$ with $a_n : V_L(p_n) \to L$.*
2. *Let I be the colimit object of $\bigcup_{n\in N} V_I(t_n) : V_I(s_n) \to V_I(p_n)$ with $b_n : V_I(p_n) \to I$.*
3. *Let R be the colimit object of $\bigcup_{n\in N} V_R(t_n) : V_R(s_n) \to V_R(p_n)$ with $c_n : V_R(p_n) \to R$.*
4. *Morphisms l and r are uniquely determined by the universal property of colimit (I, b_n) such that $a_n \circ l_n = l \circ b_n$ and $c_n \circ r_n = r \circ b_n$.*
5. *$C = \bigcup_{n\in N} C_{s_n} \cup \bigcup_{n\in N} C_n$.*
6. *$Y = \bigcup_{n\in N} Y_{s_n} \cup \bigcup_{n\in N} Y_n$.*

Match $m_{COV} : L \to G$ is uniquely determined by the universal property of colimit (L, a_n), i.e. $m \circ a_n = m_n$. An amalgamated graph transformation *is a direct transformation $G \overset{\hat{p}_{COV}, m_{COV}}{\Longrightarrow} H$ applying amalgamated rule $\hat{p}$ at match m.*

*A parallel attributed graph transformation system $PAGTS = (S, IScheme)$ based on (**ASSIG-Alg**, M) consists of an $ASSIG$-algebra S, called start graph and a set $IScheme$ of interaction schemes.*

Parallel transformation sequences and the language of a parallel attributed graph transformation system are defined analogously to Def. 5.

4 Translating AHL Nets to Sequential Graph Transformation Systems

The translation of AHL nets to attributed graph transformation systems generalizes that of P/T nets into graph transformation systems as proposed in the literature [3, 15] and reviews in a slightly modified form the concepts and results in [1]. An initially marked AHL net N together with its behavior is translated to an attributed graph transformation system $AGT = (G, P)$ with start graph G being the translation of the AHL net N with initial marking to an attributed graph typed over the type graph for AHL nets ASSIG_{AHL} (Def. 4), and the set of rules P being behavior rules $\hat{p_t}$, one for each transition $t \in T$ where L and R contain the transition's pre- and postdomain, and the rule application condition corresponds to the firing condition of t.

Definition 14 (Translation of a marked AHL net to an Attributed Graph). *Given an AHL net $N = (SPEC, P, T, pre, post, cond, A)$ with marking $m \in (A \times P)^{\oplus}$. The translation Tr of (N, m) is given by the function $Tr : (AHLnet, (A \times P)^{\oplus}) \to \mathsf{ASSIG}_{AHL}\text{-}\mathsf{Alg}$ from the set of pairs of AHL nets plus markings to the set of algebras wrt. the attributed graph structure signature ASSIG_{AHL} (Def. 4) with*

$$\begin{aligned} Tr(N,m) = G = (&G_{Place}, G_{Trans}, G_{Token}, G_{EdgeTk}, G_{ArcPT}, G_{ArcTP},\\ &op_{sPT}, op_{tPT}, op_{sTP}, op_{tTP}, op_{sTk}, op_{tTk},\\ &attr_{tv}, attr_{iPT}, attr_{iTP}, attr_{cond}), \qquad \textit{where} \end{aligned}$$

$G_{DSIG} = T_{OP}(X) \uplus A$ (disjoint union of the term algebra with variables over $ASSIG_{AHL}$ and A),
$G_{Place} = P$ (the place nodes), $G_{Trans} = T$ (the transition nodes),
$G_{Token} = \{tk | tk = (a, p, i) \in \widetilde{m}\}$. The multiset $m \in (A \times P)^{\oplus}$ is given by the set $\widetilde{m} = \{(a, p, i) \in A \times P \times \mathbb{N}\, | 0 < i \leq m(a, p)\}$, where multiple occurrences of the same element in m are numbered by i in $\widetilde{m}$,
$G_{EdgeTk} = \{e_{tk} | tk \in G_{Token}\}$,
$G_{ArcPT} = \{arcPT | arcPT = (term, p, i) \in PreSet\}$,
$G_{ArcTP} = \{arcTP | arcTP = (term, p, i) \in PostSet\}$, where the multisets of terms in arc inscriptions are given by the sets $PreSet = \cup_{t \in T} PreSet_t$ and $PostSet = \cup_{t \in T} PostSet_t$ where $PreSet_t = \{(term, p, i) | pre(t)(term, p) \geq i > 0\}$ corresponds to $pre(t)$ and, analogously, $PostSet_t$ to $post(t)$.

$op_{sPT} : G_{ArcPT} \to G_{Place}$ with $op_{sPT}(term, p, i) = p\ \forall (term, p, i) \in G_{ArcPT}$,
$op_{tPT} : G_{ArcPT} \to G_{Trans}$ with $op_{tPT}(term, p, i) = t$, if $(term, p, i) \in PreSet_t$, $\forall (term, p, i) \in G_{ArcPT}$,
$op_{sTP} : G_{ArcTP} \to G_{Place}, op_{tTP} : G_{ArcTP} \to G_{Trans}$: analogously,
$op_{sTk} : G_{EdgeTk} \to G_{Token}$ with $op_{sTk}(e_{(a,p,i)}) = (a, p, i)\ \forall e_{(a,p,i)} \in G_{EdgeTk}$,
$op_{tTk} : G_{EdgeTk} \to G_{Place}$ with $op_{tTk}(e_{(a,p,i)}) = p\ \forall e_{(a,p,i)} \in G_{EdgeTk}$,

$attr_{tv} : G_{Token} \to \mathbb{N}$ with $attr_{tv}((a, p, i)) = a\ \forall (a, p, i) \in G_{TV}$,
$attr_{iPT} : G_{ArcPT} \to T_{OP}(X)$ with $attr_{iPT}((term, p, i)) = term\ \forall (term, p, i) \in G_{ArcPT}$, $attr_{iTP} : G_{ArcTP} \to T_{OP}(X)$: analogously,
$attr_{cond} : G_{Trans} \to \mathcal{P}_{fin}(EQNS(X))$ with $attr_{cond}(t) = cond(t)\ \forall t \in G_{Trans}$

Example 2 (AHL net Dining Philosophers *translated to an attributed graph).* Fig. 3 shows the attributed graph resulting from the translation of the initially marked AHL net presented in Fig. 1 (b). We visualize Place nodes as ellipses, Transition nodes as rectangles, and Token nodes as coloured circles containing the token value attributes. Token nodes are connected to their places by EdgeTk arcs. ArcPT and ArcTP symbols are drawn as edges which are attributed by the arc inscription terms. In this example we have no firing conditions.

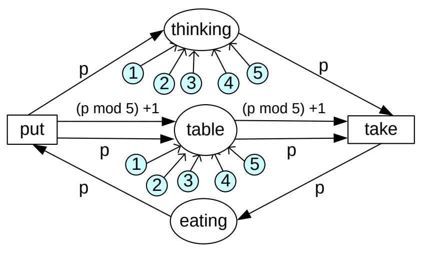

Fig. 3. Translation of AHL net *Dining Philosophers* with initial marking

In addition to the statical structure of the AHL net and the net marking, we now define the translation of a net's firing behavior into a set of graph rules P_{Tr}, the so-called behavior rules. Each behavior rule encorporates the firing behavior of one transition: the left-hand side contains its predomain, the right-hand side its postdomain. The firing condition $cond(t)$ is translated to the attribute condition of the behavior rule for transition t. The firing rules P_{Tr} together with the translated initially marked AHL net $Tr(N,m)$ form an attributed graph transformation system, the translation of the AHL net N including its behavior.

Definition 15 (Translation of AHL net firing behavior to graph rules). *Let $N = (SPEC, P, T, pre, post, cond)$ be an AHL net. We translate the firing behavior of N to a set of behavior rules $P_{Tr} = \{p_t = (L_t \overset{l_t}{\leftarrow} I_t \overset{r_t}{\rightarrow} R_t) | t \in T\}$ where for each transition $t \in T$ the rule components L_t, I_t and R_t are attributed graphs over ASSIG_{AHL} (Def. 4), defined as follows:*
The interface I_t contains only nodes of sort Place *(the environment of transition t) and no operations. All sorts and operations in L_t and R_t are empty, except* Place, Token, EdgeTk *and the adjacent operations:*

- $L_{Place} = I_{Place} = R_{Place} = \{p | p \in pre(t) \cup post(t)\}$
- $L_{Token}[R_{Token}] = \{tk | tk = (term, p, i) \in PreSet_t[PostSet_t]\}$
- $L_{EdgeTk} = \{e_{tk} | tk \in L_{Token}\}$,

- $op^L_{sTK} : L_{EdgeTk} \to L_{Token}$ *with* $op^L_{sTk}(e_{(term,p,i)}) = (term, p, i)$,
- $op^L_{tTK} : L_{EdgeTk} \to L_{Place}$ *with* $op^L_{tTk}(e_{(term,p,i)}) = p$,
- $attr^L_{tv} : L_{Token} \to T_{OP}(X)$ *with* $attr_{tv}((term, p, i)) = term$
  (analogously for $R_{EdgeTk}, op^R_{sTK}, op^R_{tTK}$ and $attr^R_{tv}$)

The rule morphisms $L_t \overset{l_t}{\leftarrow} I_t$ and $I_t \overset{r_t}{\rightarrow} R_t$ are given by $(p_{Place}, p_{Trans}, p_{Token}, p_{ArcPT}, p_{ArcTP}, p_{EdgeTk}) = (id_{Place}, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$. Let $C = cond(t)$ be a set of attribute conditions over $T_{OP}(X)$ as defined in Def. 6. Then, $\hat{p_t} = (p_t, C, X)$ is the conditional rule corresponding to the firing behavior of transition t.

Remarks. Both L_t and R_t contain only the places of the transition's environment and tokens connected to these places, where the tokens are attributed by terms of $T_{OP}(X)$. The difference between L_t and R_t is that L_t corresponds to $pre(t)$ whereas R_t corresponds to $post(t)$. The Token symbols are not in the interface I_t as the rule models the deletion of tokens from the predomain (L_t) and the addition of tokens to the postdomain (R_t).

Combining the translations of a marked AHL net and of its firing behavior, we obtain a complete translation of a marked AHL net including its behavior to an attributed graph transformation system:

Definition 16 (Translation of a marked AHL net and its Firing Behavior to an Attributed Graph Transformation System). *Let N be an AHL net and m its initial marking. Then the translation $Tr^{AGT}(N,m) : (AHLnet, M^{\oplus}) \to \mathsf{AGT}$ from the set of pairs of AHL nets plus markings to the set* AGT *of attributed graph transformation systems over graph structure signature ASSIG_{AHL} (Def. 4) is defined by $Tr^{AGT}(N,m) = (S_{Tr}, P_{Tr})$ where start graph $S_{Tr} = Tr(N,m)$ is the translated AHL net marked by m according to Def. 14, and the set of conditional behavior rules $P_{Tr} = \{(\hat{p_t}, C, X) | t \in T\}$ is the translation of the firing behavior of all transitions $t \in T$ as defined in Def. 15.*

Example 3 (Attributed graph transformation system for the Dining Philosophers*).* Let N be our AHL net as shown in Fig.1 (b), and $S_{Tr} = Tr(N, m)$ be its translation to an attributed graph as shown in Fig. 3. Then, the behavior transformation system for our AHL net is given by $Tr^{AGT}(N) = (S_{Tr}, P_{Tr})$ with P_{Tr} being the set of two behavior rules constructed according to Def. 15. These behavior rules are shown in Fig. 4. Note that place nodes are preserved by the rule mapping (equal numbers for an object in L and R means that this object is contained in the interface I), and token nodes are deleted (predomain tokens) or generated (postdomain tokens).

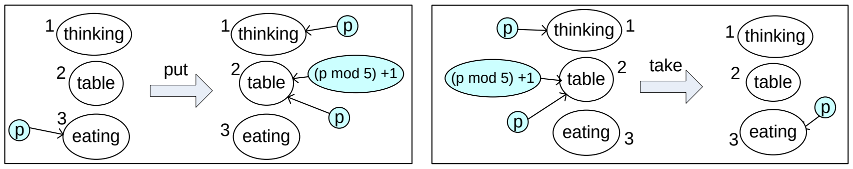

Fig. 4. Translated firing behavior of the AHL net *Dining Philosophers*

In $Tr^{AGT}(N)$ the model behavior is simulated by applying the behavior rules from P_{Tr} to the start graph S_{Tr} and to the sequentially derived graphs which correspond to different markings of N.

Proposition 2 (Semantical Compatibility of AHL net N and its translation to an Attributed Graph Transformation System). *The semantics of an AHL net N with initial marking m_{init} and the semantics of the translation $Tr^{AGT}(N, m_{init})$ are compatible, denoted by $Sem_{AHL}(N, m_{init}) \cong Sem_{AGT}(Tr^{AGT}(N, m_{init}))$, where the semantics of an AHL net is given by a set of firing sequences (firing steps), and the semantics of an attributed graph transformation system by a set of transformation sequences.*

For the proof of Proposition 2 we need to translate an attributed graph $Tr(N, m)$, which is the translation of the marked AHL net (N, m), back to a marked AHL net N_{back}. In Lemma 1 we show that the marking of N_{back} is the same as the marking of the original net N.

Definition 17 (Backward Translation of the Attributed Graph $Tr(N, m)$ to a Marked AHL Net). *Given an AHL net $N = (P, T, pre, post, SPEC, cond, A)$ with marking $m \in (A \times P)^{\oplus}$ and its translation, the attributed graph $Tr(N, m)$, constructed as defined in Def. 14. Then the backward translation Back of $Tr(N, m)$ is the marked AHL net N_{back} with:*

$P_{back} = G_{Place}$, $T_{back} = G_{Trans}$,
$\forall t \in T : pre_{back}(t) = \sum_{j=1}^{|G_{ArcPT}|}(term_i, p_i)$ *with* $(term_j, p_j, i) \in G_{ArcPT}$,
$\forall t \in T : post_{back}(t) = \sum_{j=1}^{|G_{ArcPT}|}(term_i, p_i)$ *with* $(term_j, p_j, i) \in G_{ArcTP}$,

$\forall t \in T : cond_{back}(t) = attr_{cond}(t)$,
$m_{back} = \sum_{tk \in G_{Token}} (attr_{tv}(op_{sTk}(e_{tk}), op_{tTk}(e_{tk})))$

Lemma 1 (Compatibility of Markings of N and $Back(Tr(N,m))$).
Let $m \in (A \times P)^{\oplus}$ be a marking of an AHL net N, $G = Tr(N,m)$ be the translation of N with marking m according to Def. 14, and $Back(G)$ the backward translation as defined in Def. 17. Then, the marking of $Back(G)$ is the same as the marking of N, i.e. $m_{back} = m$.

Proof of Lemma 1:

$$\begin{aligned} m_{back} &= \textstyle\sum_{e_{tk} \in G_{EdgeTk}} (attr_{tv}(op_{sTk}(e_{tk})), op_{tTk}(e_{tk})) \\ &= \textstyle\sum_{e_{(a,p,i)} \in G_{EdgeTk}} (a,p) = \sum_{(a,p,i) \in G_{Token}} (a,p) = m \end{aligned}$$

$\triangle$

Proof of Proposition 2

We show that

1. For each firing step $m[t, asg\rangle m'$ and for $G = Tr(N,m)$ there is a transformation step $d : G \xRightarrow{p_t} H$ where p_t is the behavior rule corresponding to transition t, such that the marking m' is the same as the marking of the backward translated AHL net $Back(H)$.
2. Each firing sequence $\sigma \in Sem_{AHL}(N, m_{init})$ corresponds to a transformation sequence $\sigma' \in Sem_{AGT}(Tr^{AGT}(N, m_{init}))$. This means, for all firing sequences $\sigma_i = (m_i[t_i, asg_i\rangle m'_i) \wedge 1 \leq i \leq n$ such that $m'_{i-1} = m_i$, we have $Tr(N, m'_{i-1}) = Tr(N, m_i)$ in the corresponding transformation sequence $Tr(N, m_i) \xRightarrow{r_{t_i}} Tr(N, m'_i)$, for $1 \leq i \leq n$.

ad (1)
Let $m[t, asg\rangle m'$ be a firing step, $G = Tr(N,m)$ the translation of N, m to an attributed graph and $p_t : L_t \to R_t$ the behavior rule for transition t as defined in Def. 15.

We define the match $m_{asg} : L_t \to G$ as follows: m_{asg} is the identity on the sorts $G_{Place}, G_{Transition}, G_{ArcTP}$ and G_{ArcPT}. For tokens in G_{Token} we define $m_{asg}((term, p, i)) = (\overline{asg}_A(term), p, i)$ and for token edges in G_{EdgeTk} we define $m_{asg}(e_{(term,p,i)}) = e_{(\overline{asg}_A(term),p,i)}$. $m_{asg_A} : T_{OP}(X) \to T_{OP}(X) \cup A$ maps each term in $T_{OP}(X)$ to its extended assignment in A: $m_{asg_A}(term) = \overline{asg}_A(term)$.
We show that the match m_{asg} satisfies the graph morphism properties for the token edges $e_{(term,p,i)} \in G_{EdgeTk}$:
$m_{asg}(op_{sTk}(e_{(term,p,i)})) = m_{asg}((term, p, i)) = (\overline{asg}_A(term), p, i) =$
$op_{sTk}(e_{(\overline{asg}_A(term),p,i)}) = op_{sTk}(m_{asg}(e_{(term,p,i)}))$.
Analogously, $m_{asg}(op_{tTk}(e_{(term,p,i)})) = op_{tTk}(m_{asg}(e_{(term,p,i)}))$.

The transformation step $d : G \xRightarrow{p_t} H$ is constructed as follows: Using rule p_t and match m_{asg}, we obtain for $G = Tr(N,m)$ a transformation step as double pushout of $p_t : L_t \xleftarrow{l_t} I_t \xrightarrow{r_t} R_t$ and m_{asg} in **ASSIG-Alg** due to Def. 5. The gluing object I_t just contains all places of L_t as these are the only nodes preserved by the rule. As the m_{asg} is injective, the pushout object H is constructed by $H \tilde{=} G - m_{asg}(L_t) + m_{asg}(l_t(I_t)) + m^*_{asg}(R_t) - m^*_{asg}(r_t(I_t))$. As $D \tilde{=} G - m_{asg}(L_t) + m_{asg}(l_t(I_t))$ and $g : D \to G$, $h : D \to H$ are inclusions, we have $H \tilde{=} G - m_{asg}(L_t) + m^*_{asg}(R)$.

Places are preserved by the rule, i.e. if $p \in G$ then $p \in H$. The transition and its adjacent arcs of type $ArcPT$ and $ArcTP$ are deleted and generated again. This means, if $t \in G$ then $t \in H$ and if $e_{ArcTP}, e_{ArcPT} \in G$ then $e_{ArcTP}, e_{ArcPT} \in H$. Tokens $tk \in G_{Token}$ which are in the match of m_{asg} are deleted. All other tokens are preserved, i.e. they are in D. Tokens in R_{Token} are generated, i.e. if $tk \in R_{Token}$ then $m^*_{asg}(tk) \in H_{Token}$. For token values of $tk \in m^*_{asg}(R_{Token})$ we have $attr^H_{tv}(tk) = m^*_{asg}(attr^R_{tv}(tk))$ and for the token edges $e_{tk} \in m^*_{asg}(R_{EdgeTk})$ we have $attr^H_{tv}(op^H_{sTk}(e_{tk})) = m^*_{asg}(attr^R_{tv}(op_{sTk}(e_{tk})))$.

We show now that $H = Tr(N, m')$. We know that $H = Tr(N)$ for all sorts except Token, EdgeTk and the adjacent operations, because rule p_t preserves all places and deletes and generates the transition and arcs of type $ArcPT$ and $ArcTP$. Concerning the tokens in H, we have to show that m_{back}, the marking of $Back(H)$, equals m', the resulting marking after the firing step.

$$
\begin{aligned}
m_{back} = & \sum_{e_{tk} \in H_{EdgeTk}} (attr^H_{tv}(op^H_{sTk}(e_{tk})), op^H_{tTk}(e_{tk})) \\
& \text{(due to the definition of } m_{back} \text{ in Def. 17)} \\
= & \sum_{e_{tk} \in G_{EdgeTk}} (attr^G_{tv}(op^G_{sTk}(e_{tk})), op^G_{tTk}(e_{tk})) \\
& \quad - \sum_{e_{tk} \in m_{asg}(L)} (attr^L_{tv}(op^L_{sTk}(e_{tk})), op^L_{tTk}(e_{tk})) \\
& \quad + \sum_{e_{tk} \in m^*_{asg}(R)} (attr^R_{tv}(op^R_{sTk}(e_{tk})), op^R_{tTk}(e_{tk})) \\
& \text{(due to definition of } H \text{ as pushout)} \\
= m - & \sum_{e_{(\overline{asg}_A(term),p,i)} \in m_{asg}(L)} (\overline{asg}_A(term, p)) \\
& + \sum_{e_{(\overline{asg}_A(term),p,i)} \in m^*_{asg}(R)} (\overline{asg}_A(term, p)) \\
& \text{(due to def. of } m \text{ as marking of } Back(G) \text{, and of } attr_{tv}, op_{sTk} \text{ and } op_{tTk}) \\
= m - & \sum_{(term,p,i) \in PreSet_t} (\overline{asg}_A(term), p) \\
& + \sum_{(term,p,i) \in PostSet_t} (\overline{asg}_A(term), p) \\
& \text{(due to def. of } L_{EdgeTk}, R_{EdgeTk} \text{ in Def. 15)}
\end{aligned}
$$

$$= m - \sum_{j=1}^{n} (\overline{asg}_A(term_i), p_i) \text{with } (term_j, p_j, i) \in PreSet_t, n = |PreSet_t|$$

$$+ \sum_{j=1}^{n} (\overline{asg}_A(term_i), p_i) \text{with } (term_j, p_j, i) \in PostSet_t, n = |PostSet_t|$$

$$= m \ominus pre_A(t, asg_A) \oplus post_A(t, asg_A)$$

(due to def. of $pre_A, post_A$ in Def. 2)

$$= m'$$

(due to definition of m' in Def. 2)

△

ad (2)
We prove the correspondence of firing sequences and transformation sequences by structural induction over the length of the sequences.

Induction Anchor
For the length of 0, there is no firing sequence. We only have to show that for a marking m of N and the translation $G = Tr(N, m)$ the marking of the backward translation $B = Back(Tr(N, m))$ equals m. We have shown this in Prop. 1.

For the length of 1, we have one firing step $m[t, asg\rangle m'$ and we have to show that for $G = Tr(N, m)$ there is a transformation step $d : G \overset{p_t}{\Longrightarrow} H$ where p_t is the behavior rule corresponding to transition t, s.t. the marking m' corresponds to the marking of the backward translated AHL net $Back(H)$. We have proven this in part 1 (ad (1)) of this proof.

Induction Step
We have to show that if there is a correspondence of firing sequences and transformation sequences of length n, then there is also a correspondence of sequences of length $n+1$. We know that the n'th firing step from the firing sequence of length n, $m_n[t_n, asg_n\rangle m'_n$ corresponds to a transformation $Tr(N, m_n) \overset{r_{t_n}}{\Longrightarrow} Tr(N, m'_n)$. For the firing step $m_{n+1}[t_{n+1}, asg_{n+1}\rangle m_{n+1}$ with $m'_n = m_{n+1}$ the corresponding transformation step is defined by $Tr(N, m_{n+1}) \overset{r_{t_{n+1}}}{\Longrightarrow} Tr(N, m'_{n+1})$.

We have to show that the marking of $Back(Tr(N, m_{n+1}))$ equals the marking of $Back(Tr(N, m_{n'}))$. By the definition of backward translation we know that the marking of $Back(Tr(N, m_{n+1}))$ equals m_{n+1} and the marking of $Back(Tr(N, m'_n))$ equals m'_n. As we know that $m'_n = m_{n+1}$, the transformation sequence is now of length $n+1$ and because the marking of $Back(Tr(N, m'_{n+1}))$ equals m'_{n+1}, it has the desired property.

△

Up to now we discussed an approach of simulating AHL net behavior by graph transformation which is based on compiling the behavior of AHL nets into graph rules. The disadvantage of this approach is that for Petri nets in contrast to other visual languages there is no general behavior transformation system which can be applied to all language elements, but that for each different model (i.e. for each AHL net), the compilation or translation to its corresponding graph transformation system has to be performed according to Def. 16.

In order to have a more general approach for modeling Petri net behavior by graph transformation, we propose to use parallel graph transformation and thus avoid the model-specific translation of transitions to behavior rules.

5 AHL Net Simulation by Parallel Graph Transformation

In this section, we define AHL net behavior by parallel graph transformation (interpreter approach) and compare this approach to the compiler approach presented in Section 4.

For the construction of the covering construction for behavior rules we need a graph to define the set of matches MA from all subrules and rules in the interaction scheme (see Def. 12). This graph needs to supply all the information we need for the behavior rule construction. It contains the predomains of all transitions in form of virtual tokens, i.e. tokens being the terms in $PreSet$ corresponding to the ArcPT inscriptions, and the information about the postdomains in form of ArcTP inscriptions. As we use only "virtual" tokens, we call this graph V *virtually marked AHL net graph.* The amalgamation construction over V then yields amalgamated rules containing the transitions and the adjacent arcs. Thus we apply a restriction functor after the amalgamation and show that the result is equivalent to the sequential behavior rules. Note that so far we do not consider firing conditions in the amalgamation, i.e. the correspondence result (Prop. 4) holds only for AHL nets without firing conditions like the Dining Philosophers.

Definition 18 (Virtually marked AHL net graph). *Let N be an AHL net, and $Tr(N,m)$ the corresponding attributed graph (acc. to Def. 14). The virtually marked AHL net graph V corresponds to $Tr(N,m)$, but is marked by terms $(term, p, i) \in PreSet$ (which virtually enables all transitions):*

$V = Tr(N,m)$ *for all sorts except* Token, EdgeTk *and the adjacent arc operations:*

$V_{Token} = \{tk | tk = (term, p, i) \in PreSet\}$, $V_{EdgeTk} = \{e_{tk} | tk \in V_{Token}\}$, *and the operations op_{sTK}, op_{tTK}, and $attr_{tv}$ are defined as the corresponding operations for the behavior rule sides in Def. 15.*

Example 4 (Virtually marked AHL net graph for the Dining Philosophers*).* The bottom graph in Fig. 6 shows the virtually marked AHL net graph V_{DIPHI} for our sample AHL net modeling the Dining Philosophers. Note that the marking of the virtually marked AHL net graph denotes the union of predomains of all transitions and has nothing to do with a specific marking as e.g. shown in Fig. 3.

Next, we define an interaction scheme for AHL nets according to Def. 10.

Definition 19 (Interaction Scheme for AHL Nets).
The interaction scheme IS_{AHL} consists of two subrules glueTrans *and* gluePlace*,*

two extending rules get *and* put*, and four subrule embeddings* t_1 : glueTrans $\rightarrow$ get, t_2 : glueTrans $\rightarrow$ put, t_3 : gluePlace $\rightarrow$ get *and* t_4 : gluePlace $\rightarrow$ put.

Fig. 5 shows the interaction scheme IS_{AHL}*, i.e. the definitions of the subrules, the extending rules, and the four embeddings. For each rule, the algebra is the term algebra* $T_{OP}(Y)$ *where* Y *is the set of variables depicted at graph objects in Fig. 5. The interaction scheme* IS_{AHL} *is local, as e.g. subrule* glueTrans *is source of embeddings to both extending rules* get *and* put.

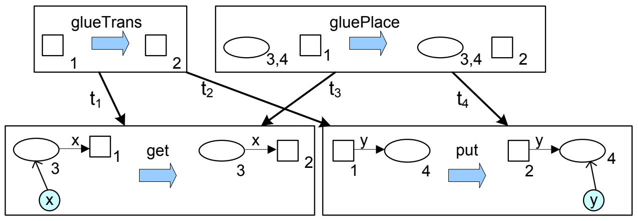

Fig. 5. Interaction Scheme for AHL nets

Example 5 (Partial Covering for the AHL net Dining Philosophers*).*
Given interaction scheme IS_{AHL} as defined in Def. 19. An instance interaction scheme IIS_{take} is shown in the upper part of Fig. 6. (Note that the detailed presentation on the left does not include all gluePlace copies.) Then, $COV_{take} = (IIS_{take}, MA_{take})$ is a partial covering with MA_{take} being a set of matches from IS_{AHL} into V_{DIPHI} as shown at the bottom of Fig. 6. The matches in MA_{take} are indicated in Fig.6 by a fat arc inscribed by MA_{take} and given precisely by node numbers. All matches from all extending rules of all subrule embeddings in IIS_{take} commute with the matches of the subrules.

COV_{take} is local as IIS_{take} is local (the subrule *glueTrans* is embedded in all extending rules) and because the matches from the left-hand sides of all three extending rule instances of get into $G_{AHL_{DIPHI}}$ are non-isomorphic. COV_{take} is additionally fully synchronized, because for each pair of extending rules we find a subrule s.t. the matches of their left-hand sides into $G_{AHL_{DIPHI}}$ overlap only in the match of this common subrule.

Note that, if a graph G and an interaction scheme are given and the covering is characterized (as e.g. for AHL nets the covering must be local, and fully synchronized), then the set of all partial coverings, i.e. the instance interaction schemes and the set of matches MA from all rules and subrules from the instance interaction scheme into G can be computed automatically.

For the covering construction for the AHL net *Dining Philosophers* this means that we can find two basic partial coverings – one for transition take in V_{DIPHI} (as shown in Fig. 6), and the other one for transition put. In the second case, a different instance interaction scheme is computed with three instances of rule put and one instance of rule get. From one instance of the subrule gluePlace there are embeddings into two of the put instances, and from one instance of the subrule glueTrans there are embeddings into all get and put instances.

A desired property of our AHL net covering construction is that it can be computed deterministically in the sense that the rules resulting from the amalgamation are unique. This property will be shown in Proposition 3.

Example 6 (Amalgamated Rule for the AHL net Dining Philosophers*).* Let $COV_{take} = (IIS_{take}, MA_{take})$ be the partial covering construction as defined in Def. 5. The LHS (RHS) of the amalgamated rule p_{take} for this partial covering is constructed according to Def. 13 by gluing the instances of the LHS (RHS) of get and put along the objects of the LHS (RHS) of their common subrules.

In the center of Fig. 6, the construction of the amalgamated rule $p_{amalg_{take}}$ from COV_{take} is shown. The embeddings of rules and subrules into the amalgamated rule are indicated by dashed arrows and given precisely by numbers.

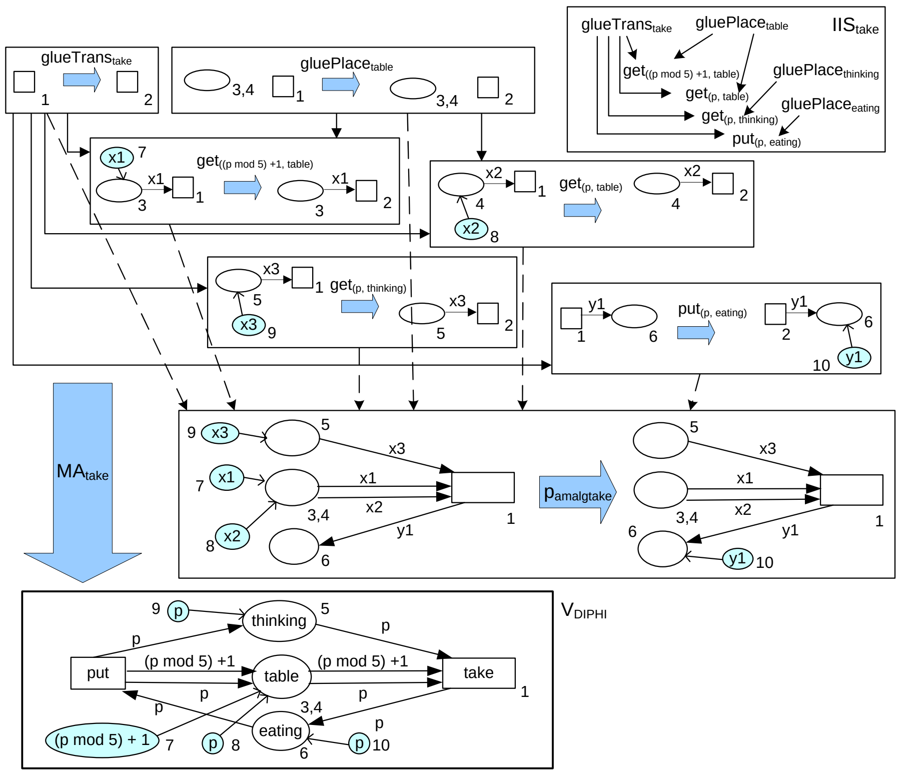

Fig. 6. Covering Construction COV_{take} and Amalgamated Rule $p_{amalg_{take}}$

The result of the amalgamation, $p_{amalg_{take}}$, is a rule corresponding to the behavior rule for transition take with two slight differences. The variables $x_1, .., x_3$

and y_1 used in the amalgamated rule have to be replaced by the right terms from $T_{OP}(X)$, and the transition and arcs must not appear in the behavior rule. The rewriting step for the variables is given by the matches in MA_{take}, where x_1 is matched to (p mod 5) + 1, and x_2, x_3 and y_1 are matched to p. The transition and arcs disappear by applying a functor restricting an ASSIG algebra such that the sorts $Trans, ArcPT$ and $ArcTP$ and the adjacent arc operations are empty.

The general construction of a partial covering for a transition $t \in T$ is the basis for the correspondence proof in Proposition 4.

Construction 1 (Partial Coverings for Amalgamated Rules modeling the Firing Behavior of AHL Net Transitions).
Let V be the virtually marked AHL net graph for net N defined in Def. 18. Let $COV_t = (IIS_t, MA_t)$ be the partial covering for a transition $t \in V_{Trans}$ with IIS_t being an instance interaction scheme of IS_{AHL} as defined in Def. 19 and MA_t the set of matches from IIS_t into V. IIS_t and MA_t are defined as follows:

- *Extending rule instances:* For each edge $arcPT \in V_{ArcPT}$ there is one instance get_{arcPT} of the extending rule get. For each edge $arcTP \in V_{ArcTP}$ there is one instance put_{arcTP} of the extending rule put.
- *Subrule instances:* There is one instance of subrule glueTrans for transition t which is embedded into all get and put instances as defined in Def. 19. For each place $p \in N_{Env_t}$ there is one gluePlace instance, called $\mathsf{gluePlace}_p$, which is embedded into all those extending rule instances get_{arcPT} with $op_{sPT}(arcPT) = p$ similar as in Def. 19. Analogously, $\mathsf{gluePlace}_p$ is embedded into all those extending rule instances put_{arcTP} with $op_{tTP}(arcTP) = p$.
- *Matches in MA_t:* The transitions of all rules and subrules in IIS_t are mapped to $t \in V_{Trans}$. The place nodes from get instances are mapped to place nodes in $pre(t)$ such that the arc inscription and the token value are mapped to the same term and the mappings overlap only in the matches of their subrules in IIS_t. Place nodes from put instances are mapped to place nodes in $post(t)$ such that the mappings overlap only in the matches of their subrules.

Proposition 3 (Existence and Uniqueness of Partial Covering COV_t). *Let V be the virtually marked AHL net graph for net N defined in Def. 18. For each transition $t \in V_{Trans}$ a local, fully synchronized partial covering $COV_t = (IIS_t, MA_t)$ constructed as in Construction 1, exists and is unique.*

Proof

We show that

1. there is at least one partial covering COV_t which is local and fully synchronized (due to the instance of glueTrans in IIS_t).
2. COV_t is unique by assuming that there are two different partial coverings $COV1_t$ and $COV2_t$ and by showing that they are equal.

ad (1)
Taking an arbitrary transition $t \in G_{Transition}$ we show that there exists exactly one partial covering $COV_t = (IIS_t, MA_t)$.

There is at least one partial covering. According to Def. 1 there is one instance of subproduction glueTrans in IIS_t. Since its left and right-hand sides contain only a transition node each, there exists a match of glueTrans to G. For all extending production instances in IIS_t we know that their matches overlap in the match of glueTrans. Thus, a partial covering COV_t exists (independent of the number of extending productions).

COV_t is local, since glueTrans is a subproduction of all instances of get and put. Moreover, each two instances of get match to two different $arcPT_1, arcPT_2 \in G_{ArcPT}$ with $op_{tPT}(arcPT_1) = op_{tPT}(arcPT_2) = t$ and each two instances of put match to two different $arcTP_1, arcTP_2 \in G_{ArcTP}$ with $op_{sTP}(arcTP_1) = op_{sTP}(arcTP_2) = t$.

COV_t is fully synchronized, since for each two instances i_1, i_2 of get or put one of the following cases holds:

- Both instance matches overlap only in transition node t. Then, their matches overlap only in the match of glueTrans.
- Both instance matches overlap also in their place node p. Then, there is an instance of gluePlace such that its match is the intersection of the matches of i_1 and i_2.

ad (2)
The partial covering is unique. Assume we have two different partial coverings $COV1_t$ and $COV2_t$. Since both are local and fully synchronized, they have to differ in the set of instances of get, put, or gluePlace. Due to Def. 1 in IIS_t, we have one instance get_{arcPT} of get for all $arcPT \in G_{ArcPT}$ with $op_{tPT}(arcPT) = t$, one instance of put_{arcTP} of put for all $arcTP \in G_{ArcTP}$ with $op_{sTP}(arcTP) = t$, and one instance of $\mathsf{gluePlace}_p$ of gluePlace for all $p \in G_P$. There are no conflicts between any two instances of get and put, since we have shown in the first part that COV_t is fully synchronized, independent of the number of instances of get and put. In the case that both instance matches overlap in more than transition t, this is only true, if there is an instance of gluePlace such that its match is the intersection of both instance matches. This holds, since there is an instance $\mathsf{gluePlace}_p$ for all $p \in N_{Env_t}$.

△

On the basis of the unique construction of the amalgamated rule $p_{amalg} : L_{amalg_t} \to R_{amalg_t}$ using the virtually marked AHL net V as host graph (step (1) in Fig. 7), we get the match $m_{cov} : L_{amalg_t} \to V$ by gluing the matches in MA_t along the matches of the subrules (step (2) in Fig. 7). Then we apply the amalgamated rule p_{amalg} at match m_{cov} to V (step (3) in Fig. 7). The resulting span $V \leftarrow V_I \to V'$ can be interpreted as rule again. This rule still contains all AHL net places, arcs and the transitions due to V being constructed once for the complete AHL net N. So we now restrict $V \leftarrow V_I \to V'$ to the elements of the environment of transition t. This transformation step is depicted as step (4) in Fig. 7. The result is the span $V|_{codom(m_{cov})} \leftarrow V_I|_{codom(i)} \to V'|_{codom(m^*_{cov})}$ which looks similar to our sequential behavior rule p_t with the difference that it still contains the transition and the adjacent arcs. Thus, in a last step (step (5) in Fig. 7) we apply a functor which forgets the transition and its adjacent arcs.

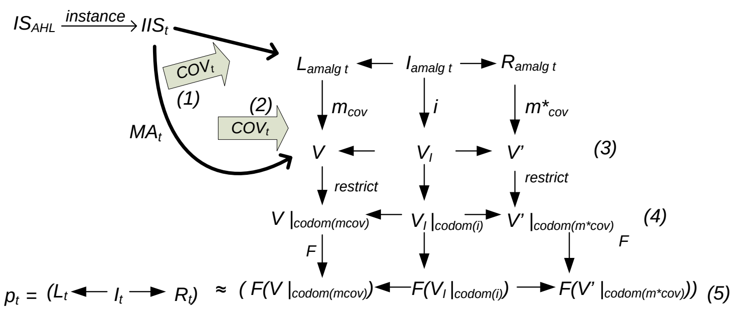

Fig. 7. Correspondence of Amalgamated Rules and Behavior Rules

Proposition 4 now formally states that the rule resulting from this functor application is isomorphic to the sequential behavior rule p_t as defined in Def. 15.

Proposition 4 (Correspondence of Amalgamated Rules and Behavior Rules for AHL Net Simulation).
Let N be an AHL net and $m \in M^{\oplus}$ its initial marking. Let $GTS_N = Tr^{AGT}(N, m) = (S_{Tr}, P_{Tr})$ be the translation of the AHL net marked by m according to Def. 16, with the set of behavior rules $P_{Tr} = \{\hat{p}_t : L_t \to R_t | t \in T\}$. Let V be the virtually marked AHL net graph for N acc. to Def. 18.

Then for each transition $t \in T$ the following holds: Given $COV_t = (IIS_t, MA_t)$, the partial covering for transition t constructed as in Constr. 1, and $p_{amalg_t} : L_{amalg_t} \to R_{amalg_t}$, the amalgamated rule for COV_t. Let m_{cov} be the match from L_{amalg_t} to V, with m_{cov} being the gluing of MA_t, and let $V \overset{p_{amalg_t}, m_{cov}}{\longrightarrow} V'$ be the transformation step. Performing an epi-mono-factorization of the corresponding rule embedding (m_{cov}, i, m^*_{cov}) leads to a new rule $p_{codom} = (codom(m_{cov}) \leftarrow codom(i) \to codom(m^*_{cov}))$. Let F be a functor that forgets transition and arcs, i.e. the sorts $Trans$, $ArcPT$, $ArcTP$ and all adjacent operations are empty. Then, $F(p_{codom}) \tilde{=} p_t$.

Proof

We construct p_{amalg_t} and $m_{cov} : L_{amalg_t} \to V$ and show that

1. the transformation step $V \overset{p_{amalg_t}, m_{cov}}{\longrightarrow} V'$ restricted to the codomain of rule embedding (m_{cov}, i, m^*_{cov}) corresponds to p_t except that it still contains the transition and adjacent arcs.
2. $F((p_{codom})$ is isomorphic to p_t.

ad (1)
Given partial covering COV_t for a transition $t \in T$, the amalgamated rule $p_{am\hat{a}lg_t} = ((p_{amalg_t} : L_{amalg_t} \overset{l_{amalg_t}}{\leftarrow} I_{amalg_t} \overset{r_{amalg_t}}{\to} R_{amalg_t}), \emptyset, Y)$ has to be constructed first. Due to the fact that colimit constructions are unique up to isomorphisms, the construction of p_{amalg_t} is also unique to isomorphism.

First, we have to show that L_{amalg_t} is isomorphic to $Tr(N_{Env_t})$ for all sorts except Token, EdgeTk and the adjacent operations as well as operations arc_{iPT} and arc_{iTP}. We have one transition node, since COV_t is local, i.e. subproduction $\mathsf{glueTrans}$ is embedded into all extending production instances of get and put. Thus, in the colimit construction all transition nodes are glued to one. For all places p in $PreSet(t)$, we know that there is at least one token required from p. Thus, there is at least one instance of get with a place node mapped to p. If there are more such instances of get, they are glued by an instance of $\mathsf{gluePlace}$, since for each place p in N_{Env_t} there is a production $\mathsf{gluePlace}$. Similarly for all places p in $PostSet(t)$, at least one token is put to p after firing t. Thus, there is at least one instance of put with a place node mapped to p. If there are more such instances of put, they are again glued by an instance of $\mathsf{gluePlace}$. All place nodes glued occur only once in L_{amalg_t}. Edges between place and transition nodes as well as tokens and their place-assigning edges in L_{amalg_t} are not glued, i.e. they occur in L_{amalg_t} as often as instances of productions get and put are in COV_t. Compare Def. 1 for the number of instances. Since there is an instance of get in COV_t for each $(term, p, i) \in PreSet_t$ and an instance of put in COV_t for each $(term, p, i) \in PostSet_t$, we get for sorts Token, EdgeTk and the adjacent operations as well as operations arc_{iPT} and arc_{iTP}:

- $L_{amalg_{t_{Token}}} = \{tk | tk = (term, p, i) \in PreSet_t\}$
- $L_{amalg_{t_{EdgeTk}}} = \{e_{tk} | tk \in L_{amalg_{t_{Token}}}\}$,

- $op_{sTK} : L_{amalg_{t_{EdgeTk}}} \to L_{amalg_{t_{Token}}}$ with $op_{sTk}(e_{(term,p,i)}) = (term, p, i)$,
- $op_{tTK} : L_{amalg_{t_{EdgeTk}}} \to L_{amalg_{t_{Place}}}$ with $op_{tTk}(e_{(term,p,i)}) = p$,

- $attr_{tv} : L_{amalg_{t_{Token}}} \to \mathbb{N}$ with $attr_{tv}((term, p, i)) = x_{(term,i)}$
- $attr_{iPT} : L_{amalg_{t_{ArcPT}}} \to \mathbb{N}$ with $attr_{iPT}((term, p, i)) = x_{(term,i)}$
- $attr_{iTP} : L_{amalg_{t_{ArcTP}}} \to \mathbb{N}$ with $attr_{iTP}((term, p, i)) = y_{(term,i)}$

I_{amalg_t} just contains all places of L_{amalg_t}. These are the only nodes preserved by the productions in IIS_t.

Since there is an instance of put in COV_t for each $(term, p, i) \in PostSet_t$, R_{amalg_t} is constructed similarly to L_{amalg_t} except for sort Token and operation $\mathsf{attr}_{\mathsf{tv}}$.

- $R_{amalg_{t_{Token}}} = \{tk | tk = (term, p, i) \in PostSet_t\}$
- $attr_{tv} : R_{amalg_{t_{Token}}} \to \mathbb{N}$ with $attr_{tv}((term, p, i)) = y_{(term,i)}$

l_{amalg_t} and r_{amalg_t} are the obvious embeddings. They result from gluing the embeddings of interfaces in productions of IIS_t.

Variable set Y to sort Nat consists of variables $\{x_i\} \cup \{y_i\}$ for $i, j \in T_{OP}(X) \times \mathbb{N}$.

The match $m_{cov} : L_{amalg_t} \to V$ is given by identical mappings on the places, transitions and arcs. For the tokens, we have $m_{cov_{DSIG}} : Y \to T_{OP}(X)$ assigning each variable in Y a term in $T_{Nat}(X)$ with $m_{Nat}(y_{(term,i)}) = term$ if $(term, p, i) \in PreSet_t$. Otherwise, $y_{(term,i)}$ is mapped to some variable in X not used in V. The codomain of m_{cov} is the subgraph of V containing the places, transitions and arcs of N_{Env_t} and all tokens from V connected to place nodes from N_{Env_t}.

In Def. 15, we defined $L_t = Tr(N_{Env_t})$ for all sorts except Token, EdgeTk and the adjacent operations. Thus, $codom(m_{cov}$ and L_t represent the same net structure (except for arc inscriptions). Moreover, there is a token in $codom(m_{cov_{Token}}$ for each $tk = (term, p, i) \in PreSet_t$, since for each $(term, p, i) \in PreSet_t$ there is an arc $arcPT \in V_{ArcPT}$, and thus one instance get_{arcPT}, due to Def. 1.

Applying p_{amalg_t} at match m_{cov} to $codom(m_{cov})$, we must show that the result graph is isomorphic to $codom(m^*_{cov})$. As stated above, L_{amalg_t} and R_{amalg_t} differ only in their token sets and adjacent edges and attributes. The same is true for $codom(m_{cov})$ and $codom(m^*_{cov})$ because here the application of p_{amalg_t} is reflected. The only difference are the token attributes which are terms in $T_{OP}(X)$ here. It is obvious that $m_{cov_{DSIG}} = m^*_{cov_{DSIG}}$.

The only difference between $codom(m_{cov})$ and L_t [$codom(m^*_{cov})$ and R_t] is that L_t [R_t] contains no arc inscriptions.

ad (2)

Applying the restriction functor F to the rule $p_{codom} : codom(m_{cov}) \rightarrow codom(m^*_{cov})$, we have to show that $F(p_{codom})$ is isomorphic to p_t. As the arc inscription operations are empty for $F(codom(m_{cov})$, the restricted rule $F(p_{codom})$ equals p_{codom} without arc inscriptions, and hence is isomorphic to p_t.

$\triangle$

6 Conclusion

In this paper we have shown how to define the behavior of AHL nets by parallel, typed and attributed graph transformation systems. This yields the advantage of an interpreter approach for simulating AHL nets. Using parallel graph transformation, possibly infinite rule sets can be described by a finite set of rules (rule schemes) modeling the elementary actions like the firing of a transition in a Petri net. The description of a rule scheme and hence of an infinite rule set is given in a purely categorical way. For AHL nets we defined an interaction scheme and constructed partial coverings. We proved the semantical compatibility between the resulting amalgamated productions and the behavior rules from the sequential graph transformation systems. The categorical definition of sequential AHL net behavior using parallel graph transformation can be extended to define *parallel* firing behavior of AHL nets. Here, the interaction scheme IS_{AHL} needs to be extended by an empty subrule to allow the construction of amalgamated rules containing more than one transition. The instance interaction schemes are still fully synchronized, but do not have to be local anymore. The amalgamation for parallel firing then yields behavior rules for combinations of different transitions and model their parallel firing.

Some restrictions had to be made when defining the behavior of AHL nets by parallel graph transformation. As we provide a general rule scheme (which is not specific to a certain net), we decided to use a fixed data signature, ASSIG_{AHL} for all AHL nets, to make use of the attribute evaluation in attributed graph transformation. Thus we use directly ASSIG_{AHL} terms and avoid to define higher-order functions operating on terms. In our running example all tokens are attributed by natural numbers. Extending the interaction scheme by variants of rules get and put allowing tokens with further kinds of attributes would increase the flexibility of the AHL net simulator. Another restriction concerns the firing conditions for transitions. In this paper, the construction of amalgamated rules and the correspondence result (Prop. 4) are defined for AHL nets without firing conditions, only. As firing conditions are translated to rule conditions in sequential graph transformation systems, this should be reflected also in the amalgamated rule construction, an extension which is planned as future work.

Tool support for AHL net simulation has been realized using GenGED [2], a tool for generating visual modeling environments. In GenGED, an alphabet editor supports the definition of the language vocabulary (alphabet) as graph structure signature and the layout of alphabet symbols by graphical constraints. A visual grammar editor allows to define different kinds of grammars based on the alphabet, e.g. for syntax-directed editing, parsing and/or simulation. Alphabet and grammars configure a specific VL environment, including an editor for the specified language (e.g. an AHL net editor). The behavior rules are used for simulation, where the underlying graph transformations are performed by Agg [21]. Moreover, they are the basis for the definition of an application-specific *animation view* [12]: the original alphabet for AHL nets is merged with a so-called *view alphabet* containing e.g. graphical symbols and layout definitions for the Dining-Philosophers example, i.e. icons for philosophers, a table and chopsticks. In addition, a so-called *view transformation grammar* is used to extend the AHL net and its behavior rules by corresponding view-specific icons. Applying the view transformation grammar to the AHL net in Fig. 1 (b) yields the animation view shown in Fig. 1 (a), and the application to the behavior rules yields the corresponding *animation rules*. These rules can be enhanced by specific *animation operations* defining e.g. the smooth movement of the philosophers' chopsticks.

In GenGED, the generated environment supports simulation/animation by applying the corresponding simulation/animation rules. Animation scenarios can be exported to the SVG format [22] and viewed by an external SVG viewer which shows continuous state changes according to the defined animation operations. Due to the generic and modular definition of syntax, behavior and animation for behavior models, the GenGED approach reduces considerably the amount of work to realize a domain-specific animation of a system's behavior. Yet, it would be even more desirable to have an interconnection between GenGED and other tools supporting the definition of visual models, e.g. the world of Petri net or UML tools. The motivations for such a tool interconnection are obvious: Petri net tools which are focused on formal analysis could profit from the animation view support offered by GenGED, whereas GenGED might export a Petri net to a Petri net tool for formal analysis. In the DFG researcher group "Petri Net Technology" [10], guided by Ehrig, Reisig and Weber, the Petri net tool infrastructure Petri Net Kernel (PNK) has been developed. As a first step towards tool interchange, an XML-based file exchange between GenGED and the PNK has been realized for place/transition nets.

Last but not least, work is in progress to implement parallel graph transformation in Agg. This extension can serve in future to simulate behavior models such as AHL nets using the interpreter approach as described in this paper.

References

1. R. Bardohl, C. Ermel, and J. Padberg. Formal Relationship between Petri Nets and Graph Grammars as Basis for Animation Views in GenGED. In *Proc. IDPT 2002: Sixth World Conference on Integrated Design and Process Technology*. Society for Design and Process Science (SDPS), 2002.
2. R. Bardohl, C. Ermel, and I. Weinhold. GenGED - A Visual Definition Tool for Visual Modeling Environments. In J. Pfaltz and M. Nagl, editors, *Proc. Application of Graph Transformations with Industrial Relevance (AGTIVE'03)*, Charlottesville/Virgina, USA, September 2003.
3. A. Corradini and U. Montanari. Specification of Concurrent Systems: From Petri Nets to Graph Grammars. In G. Hommel, editor, *Proc. Workshop on Quality of Communication-Based Systems, Berlin, Germany*. Kluwer Academic Publishers, 1995.
4. J. de Lara, C. Ermel, G. Taentzer, and K. Ehrig. Parallel Graph Transformation for Model Simulation applied to Timed Transition Petri Nets. In *Proc. Graph Transformation and Visual Modelling Techniques (GTVMT) 2004*, 2004.
5. P. Degano and U. Montanari. A model of distributed systems based on graph rewriting. *Journal of the ACM*, 34(2):411–449, 1987.
6. H. Ehrig and H.-J. Kreowski. Parallel graph grammars. In A. Lindenmayer and G. Rozenberg, editors, *Automata, Languages, Development*, pages 425–447. Amsterdam: North Holland, 1976.
7. H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer Verlag, Berlin, 1985.
8. H. Ehrig, B. Mahr, F. Cornelius, M. Grosse-Rhode, and P. Zeitz. *Mathematisch Strukturelle Grundlagen der Informatik*. Springer Verlag, 1998.
9. H. Ehrig, U. Prange, and G. Taentzer. Fundamental theory for typed attributed graph transformation. In F. Parisi-Presicce, P. Bottoni, and G. Engels, editors, *Proc. 2nd Int. Conference on Graph Transformation (ICGT'04), Rome, Italy*, pages 161–177. LNCS 3256, Springer, October 2004.
10. H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber, editors. *Advances in Petri Nets: Petri Net Technology for Communication Based Systems*. LNCS 2472. Springer, 2003.
11. H. Ehrig and G. Taentzer. From parallel graph grammars to parallel high- level replacement systems. In *Lindenmayer Systems*, pages 283–303. Springer, 1992.
12. 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*, 5, 2004.
13. C. Ermel, G. Taentzer, and R. Bardohl. Simulating Algebraic High-Level Nets by Parallel Attributed Graph Transformation. In H.-J. Kreowski, U. Montanari, F. Orejas, G. Rozenberg, and G. Taentzer, editors, *Formal Methods in Software and Systems Modeling: Essays Dedicated to Hartmut Ehrig on the Occasion of His 60th Birthday*, volume 3393 of *LNCS*. Springer, 2005.

14. U. Hummert. *Algebraische High-Level Netze*. PhD thesis, Technische Universität Berlin, 1989.
15. H.-J. Kreowski. A Comparison between Petri Nets and Graph Grammars. In *5th International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 1–19. LNCS 100, Springer, 1981.
16. J. Lilius. *On the Structure of High-Level Nets*. PhD thesis, Helsinki University of Technology, 1995. Digital Systems Laoratory, Research Report 33.
17. S. MacLane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer, New York, 1971.
18. J. Padberg, H. Ehrig, and L. Ribeiro. Algebraic high-level net transformation systems. *Mathematical Structures in Computer Science*, 5:217–256, 1995.
19. W. Reisig. *Petri Nets*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer Verlag, 1985.
20. G. Taentzer. *Parallel and Distributed Graph Transformation: Formal Description and Application to Communication-Based Systems*. PhD thesis, TU Berlin, 1996. Shaker Verlag.
21. G. Taentzer. AGG: A Graph Transformation Environment for System Modeling and Validation. In T. Margaria, editor, *Proc. Tool Exihibition at 'Formal Methods 2003'*, Pisa, Italy, September 2003.
22. WWW Consortium (W3C). *Scalable Vector Graphics (SVG) 1.1 Specification.*, 2003.