## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>From the Editors</td>
<td>1</td>
</tr>
<tr>
<td>Imprint</td>
<td>2</td>
</tr>
<tr>
<td>Cover Picture Story</td>
<td></td>
</tr>
<tr>
<td><em>Ekkart Kindler:</em> An ECNO semantics for Petri nets</td>
<td>3</td>
</tr>
<tr>
<td>Events</td>
<td></td>
</tr>
<tr>
<td><em>Jan Martijn van der Werf:</em></td>
<td></td>
</tr>
<tr>
<td>Report on PETRI NETS 2012 and ACSD 2012</td>
<td>17</td>
</tr>
<tr>
<td>Announcement</td>
<td></td>
</tr>
<tr>
<td>PNSE 2013</td>
<td>21</td>
</tr>
<tr>
<td>Research Groups</td>
<td></td>
</tr>
<tr>
<td>Fernuniversität in Hagen: Group of Jörg Desel</td>
<td>23</td>
</tr>
<tr>
<td>Newcastle University: Group of Maciej Koutny</td>
<td>27</td>
</tr>
<tr>
<td>Order Form</td>
<td>31</td>
</tr>
</tbody>
</table>

**Deadline for next volume: March 30, 2013**
From the editors

In this issue of the Petri Net Newsletter we present two Petri net research groups, one national group (Jörg Desel from the Distance University in Hagen) and one international group (Maciej Koutny from Newcastle University).

In the column of past events there is a report on the co-located conferences PETRI NETS and ACSD 2012 in Hamburg by Jan Martijn van der Werf. Moreover, there is an announcement of the workshop PNSE 2013 in Milano by Daniel Moldt.

The cover picture by Ekkart Kindler shows a model of the behaviour of place/transition Petri nets in the Event Coordination Notation (ECNO).

On behalf of all editors,

Augsburg, October 2012
Robert Lorenz
Scope: The Petri Net Newsletter serves as a medium for the rapid distribution of any information about Petri Nets and related system models all over the world. Topics include:
- reviewed technical contributions including surveys and state-of-the-art-reports
- work in progress including problems and puzzles
- reports on research groups, departments, institutes, personalities, companies, projects, local activities, events
- information about new books and PhD thesis

Subscription: Members of FG 0.0.1 of the GI receive one copy of the Newsletter free of charge. On the last page of this issue, you will find an application form containing the address of the GI. Additional and former issues can also be obtained from the GI.

Contributions: Any contributions to the field are welcome. It should be clearly stated whether a contribution is submitted as a technical contribution or as work in progress. Papers submitted as technical contributions will be reviewed. They should be 6 to 15 pages long and in A4 format. Articles submitted as work in progress will not be reviewed. As contributions will be printed as submitted, make sure that no space is wasted. Contributions should be sent to:

Robert Lorenz  
Lehrprofessur für Informatik, Universität Augsburg  
Universitätsstrasse 2, 86159 Augsburg, Germany  
robert.lorenz@informatik.uni-augsburg.de

Conference Announcements and Conference Reports: Conference announcements such as Calls for Papers and Calls for Registration are no longer in the focus of the Petri Net Newsletter. On special request, they will still be published. They should be formatted according to the layout of the Newsletter and take at most 2 pages.

Deadlines: There are two issues per year. Deadlines are the end of March and the end of September.
Abstract. The Event Coordination Notation (ECNO) allows modelling the behaviour of software on top of structural software models – and to generate program code from these models fully automatically. ECNO distinguishes between the local behaviour of elements (objects) and the global behaviour, which defines the coordination of the local behaviour of the different elements. The global behaviour is defined by ECNO’s coordination diagrams, whereas the local behaviour of the different elements can, for example, be modelled by a simple form of Petri nets, ECNO nets.

The ideas of ECNO have already been presented in earlier work. In this paper, we will show that the ECNO, in turn, can be used for modelling the behaviour of Petri nets in a simple and concise way. What is more, we will show that the ECNO semantics of Place/Transition Systems can easily be extended to so-called signal-event nets.

Keywords: Model-based Software Engineering, Local and global behaviour modelling, Event coordination, Petri net semantics.

1 Introduction

The cover picture of this issue of the Petri Net Newsletter shows the semantics of Place/Transition Systems – or at least a major part of it – in the Event Coordination Notation (ECNO). In this Cover Picture Story, we will use this semantics as an example to explain and illustrate the principles, concepts, and modelling philosophy behind the ECNO. Since the ECNO has already been explained in earlier work [1–3], we do not go into its details and do not dwell on a systematic explanation of its concepts. We rather explain the ECNO concepts as far as they are needed to understand the Petri net semantics formulated in terms of the ECNO.
2 Model-based Software Engineering

Before explaining the cover picture and the ECNO semantics of Place/Transition Systems (P/T-systems) [4] and of signal-event nets (SE-nets) [5, 6], let us give a brief background on Model-based Software Engineering (MBSE) by following up a narrative that we started some years ago [7].

One of the main ideas of Model-based Software Engineering is that the complete program code or major parts of it can be generated from models, which essentially capture the software’s domain on a high level of abstraction. We had illustrated this idea by modelling the domain of Petri nets and by providing an idea of how a graphical Petri net editor could be generated fully automatically from such a domain model combined with some additional information on the graphical representation of Petri nets [7].

Figure 1 shows a UML class diagram, which captures the main concepts of P/T-systems. A Petri net consists of a number of Nodes and a number of Arcs. Each Arc has exactly one source and one target node; in turn, every node can have any number of in-going and out-going arcs. In fact, nodes can be of two kinds: Transitions or Places, and Places may contain any number of Tokens. For

P/T-systems, there would be an additional constraint to the end that an arc may connect only a place to a transition or the other way round. This could easily be formalized in the Object Constraint Language (OCL). But, we do not formalize this constraint here for two reasons: firstly, the focus of this paper is
not on modelling the structural or syntactic aspects of a domain in UML or OCL; secondly, we will reuse the same model later in Sect. 4 for signal-event nets, in which an arc may run between two transitions – as so-called signal arc.

The model from Fig. 1 defines the concepts of Petri nets, which are models themselves. Therefore, a model such as the one from Fig. 1 is often called a meta model: a model of another model or modelling notation. But, we stick with the term domain model in this paper. If we had included OCL expressions for the additional constraints, this model would exactly capture what P/T-systems are – from a syntactic point of view. In a sense, this domain model is the software engineering way of rephrasing the well-known mathematical definition of P/T-systems as a tuple $\Sigma = (S, T; F, M_0)$ – leaving out capacities and arc weights from the definition of [4].

From a Petri net point of view, being able to formulate the syntax of some version of Petri net is not too exciting. Formulating the semantics of some version of Petri net, i.e. its firing rule, is more interesting. And also from the software engineering point of view, modelling the behaviour of a system so that the program code of the system can be generated from this model fully automatically is the more challenging part. And this is what the ECNO has been made for.

3 Firing rule of P/T-systems

As mentioned earlier, the cover picture of this issue of the Petri Net Newsletter essentially formalizes the firing rule of P/T-systems as an ECNO coordination diagram. We will explain this diagram and some supplementary models in this section. In order to make the paper self-contained and to ease following the explanations, the coordination diagram is shown again in Fig. 2. In the explanations, we jump right into the middle of the ECNO diagram – explaining ECNO’s concepts and notations on the way.

3.1 Events

In order to be able to coordinate “something” among the different elements of a Petri net, the diagram of Fig. 2 defines some events, which are graphically represented by rounded boxes. The most important event for defining the behaviour of Petri nets is the event fire; but, there are two auxiliary events: add and remove, which take the role of actually removing and adding the respective tokens from the respective places when a transition fires. It is the coordination of these events and their joint execution by the different elements that eventually will make up a firing step.

3.2 Coordination and interactions

In order to explain the basic idea of the coordination diagram of Fig. 2, let us assume that a Transition should participate in an event fire – i.e. the transition should fire. Then, the Transition would also need to participate in an event add
and an event remove. This is actually not defined in the coordination diagram itself; it is defined by the local behaviour of the Transition, which will be discussed later (see Fig. 4). The coordination diagram, however, says which other elements need to participate in an event when a transition participates in an event. The Transition does not require any other participants for the fire event. For the add event, however, the coordination diagram requires that every outgoing Arc must participate in the add event too. This is indicated by the green box with label “add” inside the Transition which is connected to the association out with coordination annotation “add-\text{->ALL}”; this can be read as: if a Transition participates in an add event, all elements at the other end of the links that represent out-going Arcs need to participate in the add event too. We call such a box inside an element a \textit{coordination set}. In the example, there are two coordination sets for the Transition. The coordination set for event remove along with the coordination annotation “remove-\text{->ALL}” says that every in-going Arc must participate in a remove event, when the Transition participates in a remove event. Summing up the coordination requirements for the transition, we know that, when a transition participates in a fire event, all its in-going arcs need to participate in a remove event and all its out-going arcs need to participate in a add event. The combination of all these participating elements with their events is called an \textit{interaction}. But, we are not yet finished, since the coordination diagram also imposes some requirements on an Arc that participates in an add or in a remove event. Let us have a look at the requirements for the add event for Arcs: it requires that all\footnote{Note that for P/T-systems, we know from the structure that there is always exactly one target place; therefore, whether we require one or all does not make a difference here. We will see later that it makes a difference, once we have signal arcs, which can run between two transitions. In order to be consistent with the semantics of SE-nets, which will be discussed in Sect. 4, we use the quantifier ALL already now.} Places at the target of the Arc also participate in the add event. Likewise, it requires that all\footnote{Again, we know that it is exactly one in this case.} Places at the source of the Arc participate in the remove event, when the Arc participates in a remove event. We will see later in the local behaviour of the Place (Fig. 6), that a Place that participates in an add event will, actually, create and add a new token to itself. Note that the place takes responsibility for creating the new token; this task cannot be delegated to the token itself, because the token does not even exist yet. That is why the co-

\textbf{Fig. 2.} ECNO coordination diagram for P/T-systems
ordination diagram does not have any requirement on involved other elements in an **add** event. When a **Place** participates in a **remove** event, however, there is another requirement: one of its **Tokens** must participate in the **remove** event too, which is indicated by the coordination annotation “**remove->ONE**”. The token does not have any additional requirement in the coordination diagram, but its local behaviour when participating in a **remove** event (see Fig. 7) will remove itself from the place – and the local behaviour of the token will guarantee that it can remove itself only once (i.e. it can be used only once in its life-time).

Altogether the coordination diagram from Fig. 2 together with the local behaviour, which will be discussed in Sect. 3.3, guarantee that for a transition that participates in a **fire** event, all the in-coming arcs, and with them all the places in the preset participate in the **remove** event; and, for each place, one of its **Tokens** participates in the **remove** event; likewise, all the out-going arcs, and with them all places in the postset participate in the **add** event. This combination of elements bound to some events is called a **valid** or **enabled interaction**. Figure 3 shows one example of a Petri net, in which one valid interaction is indicated as an octagon with the involved events. The dashed lines from the events to the different elements indicate which elements are involved, and which elements participate in which event. Note that for transition **t2**, there is no valid interaction in this situation. But, there would be one other interaction, which instead of the top-left token on place **p1** would involve the other (bottom-right) token on place **p2** for firing transition **t1**, which however is not shown in Fig. 3 in order not to clutter the diagram.

![Fig. 3](image)

Fig. 3. A Petri net example with an interaction

Note that if such a valid interaction is found, it contains the necessary tokens for firing the transition. And executing all the local actions (see details below) will remove these tokens from every place in the preset and add a new token to every place in the postset of the transition. Each valid interaction corresponds to an enabled transition of the P/T-system, and executing a valid interaction corresponds to atomically firing the transition.
Note that the semantics of ECNO is non-deterministic. In a given situation, any valid interaction could be executed. The actual choice is made outside ECNO by so-called controllers, which are beyond the scope of this paper. But, it is part of ECNO’s semantics that, once an interaction is executed, the interaction is executed \textit{atomically} and \textit{in isolation} (without interference of other interactions). Together this exactly reflects the semantics of non-deterministic and atomic firing of Petri nets. In a sense, the existence of exactly two valid interactions in the situation of Fig. 3, reflects what is called the \textit{individual token interpretation} of Petri nets [8]: there is one interaction for each possible choice of tokens for firing the transition.

The ECNO semantics even covers concurrency in Petri nets. Interactions that are independent of each other (i.e. the set of elements participating in the two interactions are disjoint), can be executed concurrently.

### 3.3 Local behaviour

In the discussion above, we have mentioned already that objects, or \textit{elements} as we call them in ECNO, have a local behaviour. In any given situation, the local behaviour of an element defines in which events it currently could participate in; in some cases, it even defines that it should participate in different events at the same time – as we have seen for transitions in our example. Moreover, the local behaviour of the element defines, what actually happens when the element participates in an event if and when the interaction is executed.

The ECNO does not prescribe any specific notation for the local behaviour. The local behaviour can be defined in different ways. One way of defining the local behaviour is by a simple version of Petri nets, which we call \textit{ECNO nets}. In our example, these ECNO nets are very degenerated nets: most of the transitions have empty presets and postsets, which means that these transitions are always enabled. We discuss the ECNO nets and their meaning below.

Figure 4 shows the behaviour of a \textit{Transition}. The model of it is an ECNO net with a single transition without any places in its pre- and postset. Therefore, it is always enable. The more interesting part is the annotation (in bold-face letters), which is called an \textit{event binding}: the sequence referring to the three events fire, remove, and add – the order actually does not matter. All three events being in the same transition binding means that, if the local behaviour participates in one of these events, it also needs to participate in the other two events within the same interaction. This way, the local behaviour of the \textit{Transition} enforces that a fire event always goes together with the add and the remove event.

The local behaviour for an \textit{Arc} is shown in Fig. 5. The ECNO net consists of two always enabled transitions, which are bound to either the event add or the event remove. This, basically, says that an \textit{Arc} can always participate in an add or a remove event.

The ECNO net for the local behaviour of a \textit{Place} is shown in Fig. 6. It is similar to the local behaviour of an \textit{Arc}. The two transitions with event binding add resp. remove tell that these events are always possible. But, for the transition bound to event add, there is another annotation (in normal font), which defines
an action. This action is executed if, in some interaction, the Place participates in an add event and when the resp. interaction is executed. The action is defined by a Java code snippet that creates a new token and adds it to the list of the place’s tokens (self.getTokens()). This Java code uses the API and code that was generated by the Eclipse Modeling Framework (EMF) [9] from the model of Fig. 1. In order to be able to access the EMF generated factory for creating new tokens, the ECNO net uses two other extensions: an import statement and a declaration of an attribute (the constant for the factory, in this case). These details, however, are not relevant here. What is relevant is that this action is one of the two actions, that ultimately makes something happen, when an interaction is executed: adding a token to a place participating in an add event.

```java
import dk.dtu.imm.se.ecno.example.petrinets.PetrinetsFactory;
final PetrinetsFactory factory = PetrinetsFactory.eINSTANCE;

a = add();
self.getTokens().add(factory.createToken());

r = remove();
self.setOwner(null);
```

The other local behaviour in which something happens is the behaviour of the Token. The ECNO net for the local behaviour of the Token is shown in Fig. 7. This time, the transition of the ECNO net has a place in its preset, which has one token initially. Therefore, this transition is initially enabled; after it has fired once, however, it will not be enabled anymore. Since the transition is bound to a remove event, this models the fact that a token can initially participate in a remove event; but after a remove event has occurred, the token can never participate in a remove event again. This way, every token can be consumed only once. The action attached to this transition, is the code that actually removes the token from the place. Again, the Java code snippet defining the action makes use of the EMF generated API; setting the tokens’ owner to null, thereby removing itself (self) from its place.
3.4 Variations and details

The definition of the semantics of P/T-systems by ECNO coordination diagrams and ECNO nets has some subtle issues, which we did not explain yet. Some of these subtleties actually concern variation points of the semantics of Petri nets. In this section, we discuss some of these variation points, in order to explain some more details of the concepts behind the ECNO.

Loops First, let us have a look at P/T-systems with loops, i.e. with a pair of arcs that connect the same place and transition in opposite directions. Figure 8 shows a Petri net with a loop along with a valid interaction. Note that the two arcs to and from the transition are required to participate in the event add or the event remove, respectively – as before. The difference now is that these two arcs “end up” at the same place $p_1$, which means that the same place $p_1$ must participate in the add as well as in the remove event. The same element participating in different events within the same interaction, is actually not new. We had seen that before in the local behaviour of the Transition – enforced by a specific event binding of the ECNO net (see Fig. 4). The loop example, however, is different. Two different arcs were requiring two different events to participate for the same place. Now, the question is whether the local behaviour of a Place would be able to participate in the event add as well as in the event remove. A look at the ECNO net for the local behaviour of a Place (see Fig. 6) shows that there is no transition, that is bound to both events at the same time. This might suggest that the local behaviour of a place would not be able to participate in both events. But actually, the two transitions of the ECNO net are completely independent of each other and, therefore, they can be executed in parallel to each other. Therefore, the interaction shown in Fig. 8 is valid – executing both events add and remove together (concurrently) is possible in the local behaviour of the place. This shows that it is not fully by coincidence that we have chosen an extension of Petri nets for defining the local behaviour of elements: one reason for using them is that Petri nets naturally come with a notion of independent or
concurrent execution of transitions. This makes modelling the local behaviour much easier in many cases.

Now, what would we need to do if we would not want transitions with loops to fire – Elementary Net Systems (EN systems) [10], for example, have this characteristic. In that case, the ECNO net for the local behaviour of the place would have an initially marked place, and both transitions (the one for the add as well as the one for the remove event), would have a loop to this place\(^3\). This would imply that a place cannot add a token and remove a token within the same interaction – transitions with loops would not be able to fire.

**Arc weights (multiple arcs)** In our discussion of the domain model for P/T-systems, we suggested that our P/T-system are ordinary, which means that all arc weights are equal to 1. But, our domain model for P/T-systems from Fig. 1 allows multiple arcs between the same element in the same direction. This way, an arc with a weight greater than 1 can be represented by multiple arcs between the respective nodes.

Now, the question is what does the ECNO semantics say in this case. To this end, we have a look at another example, which is shown in Fig. 9. It is a P/T-system with two arcs running from place p1 to transition t1. The figure also shows an interaction with the bindings of the events to the different elements. If it was not for a feature of ECNO, which we did not explain yet, this would be a valid interaction. For the two arcs bound to the remove event the coordination diagram from Fig. 1 requires that the place is also bound to a remove event; in turn, the requirement for the place says that also one of its tokens must be bound to a remove event – which is true. Therefore, without additional concepts, the transition with two incoming arcs from the same place could fire – even though there is only one token on the place. In the same way, multiple out-going arcs to the same transition would produce only one token. Whether this is what we want or not is a matter of taste. For ordinary P/T-systems, where all arc-weights are supposed to be one, this semantics would probably be okay, though it would.

\(^3\) ECNO nets have P/T-systems semantics; i.e. transitions with a loop can fire, if the respective place has a token – but two transitions connected with a loop to the same place cannot fire concurrently if there is only one token.
be better style if multiple arcs between the same place and transitions would be
forbidden syntactically (e.g. by adding some OCL constraints).

For the sake of an example, let us now assume that we want to enforce that
multiple arcs between the same place and transition should consume or produce
the same amount of tokens on the respective place. How would we achieve that
in the ECNO? Actually, it is in the model already, we just ignored this feature
up to now. A closer look at Fig. 2 reveals that the labels of the event remove
in the coordination sets in the Place and in the Token are shown between two
plus signs, which is the key to dealing with multiplicities. This notation indicates
that the event remove is treated as a parallel trigger or a counting event for a
Place and for a Token. Let us explain, what that means by the help Fig. 9: both
Arcs are bound to a remove event; and according to the coordination diagram
of Fig. 2, each of these bindings requires that the source (i.e. place p1) is also
bound to the remove event. Since remove is a counting event, this means that
the Place must be bound to the remove event twice. The local behaviour of the
Place does allow the respective event to occur twice. So, this would be possible.
Now, the place is bound to remove twice. This means that the remove event
needs to be bound the same number of times to some Token. This could either
be two different tokens – if there were more than two tokens on the place – or
two bindings to the same token. Since the remove event is also a counting event
for tokens, this would mean that the same token would be bound twice to a
remove event. But, the local behaviour for tokens does not allow the transition
with the remove event to occur twice in parallel to itself (see Fig. 7); actually,
it can occur only once in its entire life-time. Therefore, making remove counting
events for Place and Token, guarantees that the multiplicity of arcs is properly
taken into account. And the interaction shown in Fig. 9 is not valid.

Therefore the semantics of P/T-systems from the coordination diagram of
Fig. 2 together with the ECNO nets from Figures 4–7 is the semantics of P/T-
systems which properly takes the multiplicity of arcs (arc weights) into account.

3.5 Discussion

Altogether, we have defined the semantics of P/T-systems by ECNO coordina-
tion diagrams and ECNO nets. The semantics of ECNO and its coordination
diagrams was discussed in other papers [1–3] – though not formalized in math-
ematics yet. The semantics of the local behaviour (ECNO nets) is basically the
semantics of Petri nets (P/T-systems to be precise). Now, the attentive reader
might be alert: we have defined the semantics of P/T-systems by something
which refers to the semantics of P/T-systems again. A clear definition of a se-
manics should not have such cyclic dependencies. But, we argue our way out of
that problem. First of all, the point of this paper is not to define the semantics
of P/T-systems – the point of this paper, is to illustrate the way of how the sem-
antic of Petri nets can be expressed using the concepts of ECNO. Secondly, as
pointed out earlier – ECNO provides different ways of defining local behaviour
(one of them would be simply programming it). And the ECNO nets that we
used for modelling the local behaviour are very basic: synchronize some events,
execute some event – both of which are always possible and can be done concurrently to themselves; and an even more basic behaviour, one event may be executed exactly once. We could have introduced a dedicated notation for that – but we deemed that using Petri nets for the audience of this paper would be more readable. For this paper, we did not deem it worth the while to define such an ad hoc notation.

4 Firing rule of signal-event nets

In this section, we will present an ECNO semantics for another variant of Petri nets, which are called signal-event nets (SE-nets) [5]. What is more, it only takes a minor twist, to capture the essential idea of signal arcs.

A signal arc is an arc running from a transition to a transition. Since our domain model from Fig. 1 already caters for that, we use it also for SE-nets. The meaning of a signal arc is the following: suppose a transition $t_1$ can fire in some SE-net, and there is a signal arc from transition $t_1$ to a transition $t_2$. Then, if transition $t_2$ can fire in that situation, transition $t_2$ must fire together with transition $t_1$. If transition $t_2$, however, is not enabled, transition $t_1$ can fire alone. Since transitions now fire together, the kind of semantics used for SE-nets is called a step-semantics, but we do not go into details here. And since there can be chains of transitions connected with signal arcs, it can happen that more than two transitions must fire together due to transitive dependencies.

Figure 10 shows the coordination diagram that captures the semantics of signal arcs. Most of it – in particular the part for Place and Token – is identical to the coordination diagram for the behaviour of P/T-systems (Fig. 2). But, there are now also some coordination annotations concerning the fire event, and the arc is used to “propagate fire events” between transitions. Let us have a closer look at the coordination annotations for the fire event starting from the Transition. The reference from the Transition to its out-going Arcs annotated by “fire->ALL” makes sure that every out-going arc participates in the fire event too (but the arc itself does not take any action itself). If an Arc is involved in a fire

![Fig. 10. ECNO coordination diagram for SE-nets](image-url)
event, the annotation “fire->ALL” at the target reference to the Transition shows that all target Transitions need to participate in the fire event too (unless this is not possible, which will be explained shortly). Note, that an arc could either be a regular Petri net arc (in that case, it would run from a transition to a place) or a signal arc. If it is a regular arc, there would actually not be a transition at the other end, i.e. the requirement that all Transitions at the other end need to participate in fire is trivially valid. Note that the type of the Transition at the other end of the target reference, implicitly reduces the set of elements to be considered – Places are not considered under the ALL-quantification in this case. In order to point out that there is an implicit restriction of possible target elements by the type at the other end, the name of the reference target is shown in parentheses. Actually, the same applied to the source and target references to the Place in the coordination diagram for P/T-systems already – at that time, however, it did not have any bearing. Now it has.

Altogether, the two coordination annotations for fire guarantee that, if a Transition participates in a fire event, all the Transitions to which there are signal arcs will also participate in an fire event. This covers the first case of the semantics of a signal arc. But what, if the transition at the end of the signal arc cannot fire. In that case, it should be ignored. This is actually represented by the coordination set fire of the Arc, which is not connected with any out-going reference – therefore, the Arc can chose not to “propagate” the fire event to the target Transition. The semantics of ECNO is that, if there is more than one coordination set for the same event, any of them could be chosen to follow up alternatively. So, the unconnected coordination for fire takes care of the second case of the semantics of signal arcs: not firing the target transition. In general, the choice between different coordination sets for the same event is non-deterministic. This would mean that the signal arc could chose to involve the target transition or not. This, however, is not exactly the semantics of SE-nets: they require that the target transition must fire, if it is enabled. This is where priorities of coordination sets come into play. The empty coordination set for event fire of Arc has priority -1, which is indicated by the number in front of the colon. All other coordination sets have default priority 0. This way, the target transition must participate in the fire event if possible, since the coordination set propagating the fire event has higher priority.

Basically, the coordination diagram from Fig. 10 captures the semantics of signal arcs. As it is formalized now, it would, however, be possible that a transition with incoming signal arcs fires without one of the source transitions firing. We could change the coordination diagram to cater for that – basically, by introducing requirements in the reverse direction. But, there is a simpler practical solution here. As explained earlier, the actual control of events and interactions for them being issued for elements lies in so-called controllers. If we attach controllers for the fire event only to those transitions that do not have in-coming signal arcs, this has exactly the effect we want in signal-event nets: a transition with in-coming signal arcs, can be executed only by being triggered from other transitions.
 Altogether, this shows that with a minor twist of the original ECNO semantics of P/T-systems, we can obtain the semantics of SE-nets. What is more, the extension seems to be a quite natural translation of the two possible cases of the informal semantics of signal arcs.

5 Conclusion

In this paper, we have shown how to formulate the semantics of different variants of Petri nets by the help of the Event Coordination Notation (ECNO). We did not dive into the details of the semantics of ECNO, but rather used the examples to illustrate some of the features of ECNO. This way, we continued the narrative on Model-based Software Engineering and an example of a simple Petri net tool [7], which covers the behaviour of the domain now: the firing rule of Petri nets in our example.

ECNO actually was not specifically made for that purpose and has many more features, which we did not need for formalizing the semantics of Petri nets, such as parameters for events, inheritance on events, and inheritance on elements and their local behaviour. The selection of the concepts and features of ECNO were driven by applications like AMFIBIA [11] and the vision, that it should be able to formulate the semantics of ECNO in its own concept – a report on the ECNO formulation of its own semantics is in preparation. Formulating the semantics of some versions of Petri nets was a finger exercise on that way, which gives some insights into the concepts of ECNO and its modelling philosophy. Moreover, this exercise shows that ECNO allows to concisely formulate the semantics of Petri nets.

ECNO Tool support ECNO is implemented as an extension of Eclipse. The current version of the implementation of ECNO (0.3.0) supports ECNO coordination diagrams on top of Ecore diagrams (a kind of class diagrams) and ECNO nets for modelling the local behaviour. From these diagrams, the ECNO Tool can generate program code. The ECNO Tool includes an ECNO execution engine and runtime environment, which is able to execute the code generated from the ECNO models. The information on how to obtain and install this ECNO Tool, as well the examples discussed in this paper can be found on the ECNO home page at

http://www2.imm.dtu.dk/~eki/projects/ECNO/

References

Report on PETRI NETS 2012 and ACSD 2012

Jan Martijn van der Werf

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{j.m.e.m.v.d.werf}@tue.nl

This year, the 33rd annual international Conference on Application and Theory of Petri Nets and Other Models of Concurrency (PETRI NETS) and the 12th International Conference on Application of Concurrency to System Design (ACSD) were colocated at main campus of the University of Hamburg, near the Dammtor. Daniel Moldt and his team were responsible for the local organization of the conference. The satellite program was organized at the Informatics Campus.

The two conferences shared five keynotes: Sir Tony Hoare (Turing Award 1980) gave a lecture about concurrency in program execution; Alain Finkel presented infinite-complete Well-Structured Transition Systems (WSTS); Bart Jacobs addressed several topics in computer security and their importance; Joost-Pieter Katoen showed how continuous-time Markov Chains provide a truly simple semantics of Generalized Stochastic Petri nets; and Jens Sparso gave an overview of his work in the field of networks-on-chip and the current direction of research into networks-on-chips for real-time multi-processor systems.

This year’s program committee of PETRI NETS was chaired by Lucia Pomello and Serge Haddad. In total, 55 papers were submitted to the conference, which included 48 full papers and 7 tool papers. The authors of the papers represented 25 different countries. Out of these submitted papers, 18 regular papers and 3 tool papers were accepted for presentation. The proceedings have appeared as volume 7347 in the LNCS series [3]. Some authors were invited to submit an extended version of their paper in a special issue of the journal Fundamenta Informaticae. During the conference dinner, Antti Valmari and Henri Jansen won the best paper award with their paper on old and new algorithms for minimal coverability sets [7].

For ACSD, Jens Brandt and Keijo Heljanko cochaired the program committee. ACSD accepted 20 papers, including 2 tool papers, out of 48 submitted papers from authors of 18 different countries. In total, there were 60 abstract submissions. The selected papers have been published by IEEE Computer Society’s Conference Publishing Services [1]. Tim Strazny and Roland Meyer received the best paper award of ACSD for their work on an algorithmic framework for coverability in well-structured systems [6].

Both conferences were officially opened on Wednesday June 27. The satellite program comprised five workshops, a Petri net course including tutorials, and the second edition of the Petri Net Model Checking Contest. The Petri net course offered a thorough introduction to Petri nets in four half-day modules on basic net classes (Jörg Desel and Jetty Kleijn), coloured Petri nets (Lars Kristensen)
and Timed and Stochastic Petri nets (Susanna Donatelli and Serge Haddad). In
total 21 participants followed the course. Two additional tutorials were given,
one on fluid and hybrid nets (Manuel Silva and Christian Mahulea), and one on
net unfoldings (Thomas Chatain, Stefan Haar, Victor Khomenko and G. Michele
Pinna), with 16 and 19 participants, respectively.

The workshops and tutorials committee, chaired by Alex Yakovlev and Wil
van der Aalst, selected five workshops to be organized in association with the
PETRI NETS and ACSD conferences: Petri Nets and Software Engineering
(PNSE 12) [2], chaired by Lawrence Cabac, Michael Duvigneau and Daniel
Moldt; Biological Processes and Petri Nets (BioPPN 2012) [4], chaired by Monika
Heiner and Ralf Hofestäedt; Petri Net-based Security (WOOPS 2012) [5], chaired
by Rafael Accorsi, Tadao Murata and Silvio Ranise; Petri Net Compositions
(CompoNet 2012) [5], chaired by Hanna Klaudel and Franck Pommereau; and
Logics, Agents and Mobility (LAM 2012) [5], organized by Berndt Müller (Far-
wer) and Michael Köhler-Bußmeier.

Authors of the best workshop papers will be invited to submit a revised version
to be considered for inclusion in a volume of Transactions on Petri Nets and
other models of Concurrency (ToPNoC), an LNCS subseries.

This year, in total 145 participants, out of 25 different countries, attended
the two conferences and the satellite program. Almost a third of the participants
came from Germany. Not only we enjoyed the conference, we also had a delight-
ful social program. During the workshops on Monday and Tuesday, the local
organization took well care of the participants. In the local organization, there
was someone specially dedicated to the constant supply of warm, freshly baked
waffles. Together with the well-prepared Italian coffee in the form of espresso
and cappuccino with Belgian chocolates, it was a real treat for the participants.
On Monday, June 25 a workshop barbecue was organized for the workshop par-
ticipants.

The conference was held at the main campus of the
university of Hamburg while the last term was not yet at
its end, which made the conference location very lively. It
sometimes provided small moving scenes. For example, in
the cantines, some of the participants were pleased as they
were asked whether they were students. But it gave the lo-
cal organizers also a hard time to schedule the conference
with the regular lectures. In the end, they managed very
well. There was only one small blot on the organization’s
escutcheon. Although a lecture room was designated to
the conference, a local professor was teaching in that same
room. The local organization creatively resolved the prob-
lem by providing a beamer and a spot under the trees in
the main hall of the campus, which gave a very cozy atmosphere between the
participants.

The welcome reception was organized on the panorama deck of the Emporio
tower, which gave beautiful sights of Hamburg, although it was a bit clouded.
Thursday and Friday, the weather was perfect to take a walk in Hamburg. The conference dinner, during which the best paper ceremony took place, was held in the restaurant Parlament-Hamburg. Even the soccer enthusiasts were reckoned with a tv screen for the semi finals of the European Football Championship during the conference dinner.

Next year, the two conferences will go on their own again. PETRI NETS 2013 will take place in Milano, Italy, June 24 - 28, 2013. The local organization is in hands of Lucia Pomello. The program committee will be cochaired by José Manuel Colom and Jörg Desel. More information, including the call for papers can be found at http://www.mc3.disco.unimib.it/petrinets2013. ACSD 2013 is organized in Barcelona, Spain a few weeks later, July 8 - 10, 2013 by Josep Carmona. Marta Pietkiewicz-Koutny and Mihai Teodor Lazarescu will cochair the program committee. More information about ACSD together with a call for papers can be found at http://acsd.lsi.upc.edu/. We cordially invite you to consider submission to these conferences.

References

PNSE’13
International Workshop on Petri Nets and Software Engineering
Milano, Italy, June 24-25, 2013

a satellite event of

Petri Nets 2013
34th INTERNATIONAL CONFERENCE ON APPLICATION AND THEORY OF PETRI NETS AND CONCURRENCY

More information: http://www.informatik.uni-hamburg.de/TGI/events/pNSE13/

Contact e-mail: pnse13_at_informatik_dot_uni-hamburg_dot_de

Important Dates

- Deadline for full papers: March 22nd, 2013
- Deadline for short papers: March 22nd, 2013
- Notification of paper acceptance: April 30th, 2013
- Deadline for posters: May 2nd, 2013
- Notification of poster acceptance: May 6th, 2013
- Deadline for final revisions: May 21st, 2013

Scope

For the successful realization of complex systems of interacting and reactive software and hardware components the use of a precise language at different stages of the development process is of crucial importance. Petri nets are becoming increasingly popular in this area, as they provide a uniform language supporting the tasks of modelling, validation, and verification. Their popularity is due to the fact that Petri nets capture fundamental aspects of causality, concurrency and choice in a natural and mathematically precise way without compromising readability.

The use of Petri Nets (P/T-nets, coloured Petri nets and extensions) in the formal process of software engineering, covering modelling, validation, and verification, will be presented as well as their application and tools supporting the disciplines mentioned above.

Topics

We welcome contributions describing original research in topics related to Petri nets in combination with software engineering, addressing open problems or presenting new ideas regarding the relation of Petri nets and software engineering. Furthermore we look for surveys addressing open problems and new applications of Petri nets. Topics of interest include but are not limited to:

- Modelling
  - representation of formal models by intuitive modelling concepts
  - guidelines for the construction of system models
  - representative examples
  - process-, service-, state-, event-, object- and agent-oriented approaches
  - adaption, integration, and enhancement of concepts from other disciplines
  - views and abstractions of systems
  - model-driven architecture
  - modelling software landscapes
  - web service-based software development

- Validation and Execution
  - prototyping
  - simulation, observation, animation
  - code generation and execution
  - testing and debugging
  - process mining
  - efficient implementation

- Verification
  - structural methods (e.g. place invariants, reduction rules)
  - results for structural subclasses of nets
  - relations between structure and behaviour
  - state space based approaches
  - efficient model checking
  - assertional and deductive methods (e.g. temporal logics)
  - process algebraic methods
  - applications of category theory and linear logic
  - general analysis for software engineering contexts

- Application of Petri nets in Software Engineering / use of Petri nets in the domains of
  - flexible manufacturing,
  - logistics,
  - telecommunication,
  - workflow management and
  - embedded systems.

- Tools in the fields mentioned above

Submissions/Proceedings

The programme committee invites submissions of full contributions (up to 20 pages) or short contributions (up to 5 pages). Ongoing work (up to 2 pages) can also be presented in a special poster session.

Please note that for full contributions up to 15 pages are recommended if the paper should be considered for the journal publication.

Papers should be submitted in electronic form (PDF) using the Springer LNCS-format (see http://www.springer.de/comp/lncs/authors.html). Submissions should include title, authors’ addresses, E-mail addresses, keywords and an abstract. For your submission in PDF format please use the online conference management system at http://www.easychair.org/conferences/?conf=pnse13

Just login or create a new account and then upload your paper. (Later you will be able to see your reviews there.)

The papers will be peer reviewed by at least three members of the PC. Accepted contributions will be included in the workshop proceedings, which will be available at the workshop and published online at CEUR-WS.org.

Some of the best papers from the workshop will be invited for publication in a volume of the journal sub line of Lecture Notes in Computer Science entitled "Transactions on Petri Nets and Other Models of Concurrency" (ToPNoC). The papers are expected to be thoroughly revised and they will go through a totally new round of reviewing as is standard practice for journal papers.

Papers from previous instances of this workshop (PNSE07, PNDS08, PNSE09, PNSE10, PNSE11 and PNSE12) made it into ToPNoC volumes in the Springer LNCS series (volumes 5100, 5460, 5800, 6550, 6900 and 7400).

Chairs

- Daniel Moldt (University of Hamburg, Germany)
- Heiko Rölke (German Institute for International Educational Research)
Overview

Lehrgebiet Softwaretechnik und Theorie der Programmierung
Prof. Dr. Jörg Desel

Universitätsstraße 1
58097 Hagen

+49 (2331) 987 2964

Since 2010 Prof. Dr. Jörg Desel holds the chair “Softwaretechnik und Theorie der Programmierung” (software engineering and programming theory) at the Fernuniversität in Hagen. He currently works with the scientific assistants Robin Bergenthum, Tall Bremer and Sven Böttcher. The former member of the group Dr. Sebastian Mauser left 2011 after he received his PhD and is now working in the private sector. Just recently Dr. Vesna Milijic finished her PhD. Furthermore, the chair employs one secretary (Alexandra Lueg), one technical employee (Otto Reidinger) and about 10 student and research assistants. In 2011 the “AWPN” (Algorithmen und Werkzeuge für Petrinetze, algorithms and tools for Petri Nets) was located in Hagen and this year the “DeLFI 2012” (Die e-Learning Fachtagung Informatik, the e-learning conference informatics) took place in Hagen.

Teaching

The Fernuniversität in Hagen is the only distance university under public law in the German Higher Education Area. Currently, there are 79679 students at the Fernuniversität in Hagen, 15% of them studying at the faculty for mathematics and computer science. Unlike a “normal” university, the students of the Fernuniversität receive their lecture notes by post and / or download them and study at home. The students are mentored at the chair “Softwaretechnik und Theorie der Programmierung” either by phone or via various internet based media. Exercise sheets are submitted by the students completely electronically over a special web platform. These exercise sheets are in many cases corrected automatically. Additionally, there are special attendance days when the students come to Hagen to meet each other, to receive further study content and to seek special preparation for the examinations.

Currently the chair “Softwaretechnik und Theorie der Programmierung” offers the courses “Introduction to imperative programming”, “Software Engineering I – Methodical development of object-oriented desktop applications” and „Management of Software Projects“. The vast number of students at the Fernuniversität is reflected by the participants of the course “Introduction to imperative programming”. This course teaches each winter semester about 1500 students the concepts of imperative programming (data types, loops, recursion, pointers, etc.), the concept of an algorithm, how to translate an algorithm into a computer program and the basics of testing. The
course “Software Engineering I” teaches the development of object oriented software. The focus of this course is the Unified Modeling Language (UML), which has established itself as a de-facto standard in software modeling, and the single work steps within the process of software development (from the requirement elicitation to the detailed software design). The course is inspired by the ideas of Model-Driven Software Development and therefore describes the development of a software system as a sequence of discrete but evolutionary model transformations. While the topic of the course “Software Engineering I” is the design and modeling of software systems, the course “Management of Software Projects” covers different techniques and methods for the planning and the successful realization of software projects. In detail, the course teaches different software development models and methods for the project planning, project realization, project controlling and cost estimation. Furthermore, the human component is considered by explaining the techniques for employee motivation and management.

In addition to these regular courses, the chair offers practical courses and seminars related to Petri Nets. For example the subject of a practical course in the winter semester 2011/2012 was the development of an editor for Place/Transition Nets. Inspired by a talk of Prof. Dr. Wolfgang Reisig on PNSE’12 Workshop in Hamburg, 12 Students will develop several plugins for a Petri Net tool for the modeling of faithful Petri Nets, which increase the expressive power of traditional Petri Net models, in the winter semester 2012/2013. Many of the students who attend seminars or practical courses on the subject of Petri Nets use their acquired knowledge to write a thesis in the research area of Petri Nets.

Research

Modeling in Software Engineering

Modeling plays a key role in the early stages of system design. The research interest of the chair concentrates on process modeling, which plays an increasingly important role in the development of process orientated individual software as well as in the use of standard software and workflow systems. A fundamental modeling research not only takes into account the expressiveness and analysis respectively the verification of models, but considers models throughout the whole development process. The latest research topics of the chair are:

- the formalization and validation of models and their formal requirements,
- the synthesis of system models from specified non-sequential runs.

The validation of models and requirements, the verification of models and also the synthesis are supported by a tool chain (VIPTool) developed at the chair. Prof. Dr. Jörg Desel is one of the executive board’s members in the expert committee “Modellierung” (modeling) of the “Gesellschaft für Informatik (GI)” (German Informatics Society) and has (co-)organized the conferences “Modellierung ‘99” and “Modellierung 2005”.

Software specification techniques for embedded systems

The topic of this research area is the synthesis of models for embedded systems from models of uncontrolled systems and the requirements by different engineering disciplines to the overall system. These models are used for the specification and partly for the automatic generation of embedded software. They also serve as references for test procedures. The focus of the research is
the development of iterative methods for the formalization and validation of the requirements and for the simulation, visualization and verification of the models.

The results up to now were used successfully in an industrial project (cooperation with the Audi AG) in the area of developing control components for vehicles. Within the framework of the DFG priority program “Integration von Techniken der Softwarespezifikation für Ingenieurwissenschaftliche Anwendungen” (integration of techniques of software specification for engineering applications) Prof. Dr. Jörg Desel led together with Prof. Dr. Hans-Michael Hanisch the project “Integration von Spezifikations- und Modellierungstechniken bei der Modellsynthese im Steuerungsentwurf (SpeciMen)” (integration of specification and modeling techniques in model synthesis of control design). Additionally Prof. Dr. Jörg Desel is steering committee member of the conference series “Application of Concurrency to System Design (ACSD)” and he was program committee chair of the conference “ACSD 2005”.

Simulation, specification and validation techniques for process specifications for information systems

Under the finished DFG funded project “Verifikation von Informationssystemen durch Auswertung halbeordneter Petrinetz-Abläufe (VIP)” (verification of information systems through evaluation of partially ordered runs of Petri Nets), headed by Prof. Dr. Jörg Desel and Prof. Dr. Andreas Oberweis, the VIPTool tool was developed, which is now being refined in Hagen.

Prof. Dr. Jörg Desel is domain expert in the executive board of the GI group “Entwicklungsmethoden für Informationssysteme und deren Anwendung” (development methods for information systems and their applications) and organized (together with Prof. Dr. Ulrich Frank) the workshop “EMISA 2005” (Enterprise Modeling and Information Systems Architecture). He was also program committee chairman of the conferences “PROMISE 2002” (Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen, process orientated methods and tool for the development of information systems) and “BPM 2004” (Business Process Management). Furthermore, he is steering committee member of the conference series “Business Process Management”.

Formal principles of the process specification and analysis, especially with Petri Nets

The formal principles include verification techniques for distributed algorithms and component-based systems as well as synthesis methods and their use in early phases of software engineering.

Prof. Dr. Jörg. Desel was the speaker of the GI group “Petrinetze und verwandte Modellierungssprachen” (Petri Nets and related modeling languages) for nine years and established the yearly conference “Algorithmen und Werkzeuge für Petri Netze” 18 years ago. In 2003, he organized the fourth international “Advanced Course on Petri Nets” in Eichstätt (2 weeks, 100 participants). He is a steering committee member of the international Petri Nets conference. Furthermore, he was program committee chair of the “Petri Nets 1998” conference and he is program committee chair of the “Petri Nets 2013” conference.

Scenarios for media supported teaching

The research projects, often carried out with psychologists, concentrate on critical examination of technical options on the one hand side and on the other hand on the psychology of learning needs.
Prof. Dr. Jörg Desel was a foundation speaker of the GI group “E-Learning” and chairman of the program committees of the GI conferences “DeLFI 2003” and “DeLFI 2012”. As a representative of computer science he belongs to the auditor group of the “bmbf” (Bundesministerium für Bildung und Forschung, ministry for education and research) for evaluation and further support of projects in this area. Furthermore, he has co-organized some workshops in the area “usage of new media in teaching”. Being a speaker of the GI department “Informatik in der Ausbildung – Didaktik der Informatik” (informatics in education – principles of teaching of informatics), he is a member of the steering committee of the GI.
Petri Nets at Newcastle University

At Newcastle University there are two Petri Nets oriented research groups, one led by Maciej Koutny at the School of Computing Science, and the other led by Alex Yakovlev at the School of Electronic and Electrical Engineering. The groups collaborate on a variety of themes, jointly organising international conferences (recently Petri Nets 2011, ACSD 2011 and CONCUR 2012) and running a research seminar (ASL).

Maciej Koutny

School of Computing Science
Claremont Tower
Newcastle University
NE1 7RU
United Kingdom
Telephone: +44 (0)191 222 7972
Web: http://www.ncl.ac.uk/computing/people/profile/maciej.koutny
E-mail: maciej.koutny@ncl.ac.uk

Other group members: Victor Khomenko, Marta Pietkiewicz-Koutny and Jason Steggles

Brief Curriculum Vitae

Maciej Koutny received his MSc (1982) and PhD (1984) in Applied Mathematics from the Warsaw University of Technology, Poland. In 1985 he joined the then Computing Laboratory of the University of Newcastle upon Tyne to work as a Research Associate. In 1986 he became a Lecturer in Computing Science, and from 1994 to 2000 he held an established Readership at Newcastle University. After that he became a Professor in the School of Computing Science at Newcastle University.

Maciej Koutny is the chair of the Steering Committee of the International Conferences on Application and Theory of Petri Nets, and a member of the IFIP Working Group 2.2 on Description of Programming Concepts. He serves as an editor of the LNCS Transactions on Petri Nets and Other Models of Concurrency (ToPNoC), and the Scientific Annals of Computer Science journal. He has been a Visiting Professor at Leiden University, Nicolaus Copernicus University, Xidian University, University of Evry, and University of Paris 12. His Programme Committee chairmanships include: ICATPN’01, ACSD’08, CHINA’08 MeCBIC’10 and CONCUR’12. He was the scientific co-director of the 5th Advanced Course on Petri Nets held in 2010.

Research

Research interests of the group centre on the theory of distributed and concurrent systems, including both theoretical aspects of their semantics and application of formal techniques to the modelling and verification of such systems; in particular, model checking based on net unfoldings. Two main application areas are asynchronous VLSI circuits and systems biology. Examples of specific research topics are listed below.
Causal process semantics of concurrent systems

Occurrence nets are representatives of single runs of a Petri net, which are suitable for describing explicit causality and concurrency. In this work we extended the causal process semantics to Petri nets with inhibitor arcs and then nets with localities. We also developed an extension of casual processes to gain better understanding of complex fault-error-failure chains.

Concurrency models: unification of Petri nets and process algebras

The standard treatment of the structure and semantics of concurrent systems provided by process algebras and Petri nets is different, making it difficult to take full advantage of their relative strengths. We achieved a unification of basic classes of Petri nets and process algebras. This has been later extended to deal with asynchronous message passing and mobility.

Framed temporal logic programming

This work is concerned with various syntactic and semantic aspects of an executable temporal logic programming language, and is based on the projection temporal logic and some of its laws are investigated. A synchronous communication mechanism for concurrent programs is provided by means of a framing technique and minimal model semantics.

Higher-order algebraic methods

This work has involved the development of techniques and tools based on higher-order algebraic methods to verify stream processing systems (e.g. systolic and dataflow devices, real-time systems, and families of computing devices).

Information flow analysis through opacity

Opacity is a technique for describing security properties in information flow. It captures properties of the local states of the secure (or high-level) part of the system, based on the observation of the local states of a low-level part of the system as well as observed actions. We first formalised opacity as a Petri net modelling technique and then extended it to general labelled transition systems. We established links between opacity and anonymity and non-inference, and also investigated ways of verifying opacity.

Model-checking using net unfoldings

The group has achieved notable advances in tool support for applicable modelling and reasoning approaches, contributing high performance model-checking tools for software and hardware verification. Techniques for checking behavioural properties of concurrent systems have been developed using Petri net unfoldings and other highly condensed representations, integer programming and SAT-based techniques. In the area of hardware verification, fast SAT-based methods have been used for detecting state encoding conflicts in specifications and for logic synthesis of asynchronous circuits, as well as visualisation of concurrent behaviours.
Petri nets and biological networks

This work has investigated an application of formal techniques to biological networks and has developed a range of Petri net models and tools which can be used to model and analyse genetic regulatory networks.

Relating communicating processes with different interfaces

It is often desirable to describe the interface of an implementation system at a different (usually more detailed) level of abstraction to the interface of the relevant specification. We developed an approach to deal with behaviour abstraction in networks of communicating processes which is, in particular, suitable for compositional verification.

Synthesis of concurrent systems

We investigated the following version of the synthesis problem: given a transition system representing the desired behaviour, derive a Petri net which generates this transition system. Recently, we provided a new solution for this problem for Petri nets with localities. This has led to a proposal of a new model of concurrent behaviour involving Petri nets (providing the structure of a system) and execution policies additionally constraining their dynamic behaviour (such as maximal concurrency).

Semantics of membrane systems

Using Petri nets with explicit localities we developed a causal process model describing the structure of the behaviour of membrane systems. Each locality identifies a distinct set of transitions which may only be executed synchronously in a locally maximal concurrent manner.

Verification and synthesis of asynchronous circuits

The verification techniques based on Petri net unfoldings and SAT (see above) have been applied to solving various problems arising when designing asynchronous circuits, such as encoding conflict detection and resolution.

Verification of mobile and dynamic systems

We developed a translation of a subset of pi-calculus to Petri nets. The resulting nets can be unfolded using the methods developed by the group, and then efficiently verified.

Selected Recent Publications

• Koutrny M, Randell B. Structured Occurrence Nets: A formalism for aiding system failure prevention and analysis techniques. Fundamenta Informaticae 97, 2009.
• Mokhov A, Khomenko V, Yakovlev A. Flat Arbiters. Fundamenta Informaticae 108, 2011.
PETRI NET NEWSLETTER

The rapid information service on Petri Nets and related system models published by the Special Interest Group on Petri Nets and related system models (FG 0.0.1) of the Gesellschaft für Informatik e.V. (GI)

Two issues per year. 66 issues have been published from 1979 to 2004.

Topics include:

- technical contributions
- state-of-the-art reports
- reports on new books
- information on software packages
- project announcements and reports
- conference announcements, programs, and reports
- abstracts of recent publications

Members of FG 0.0.1 of the GI receive the Newsletter free of charge. This membership causes no further obligations or charges. For becoming a member send this application form to

Gesellschaft für Informatik e.V.
Wissenschaftszentrum
Ahrstraße 45
D - 53175 Bonn
Germany
fax +49 228 302 167

Additional and former issues of the Newsletter can be obtained from the same address.

Application Form
Petri Net Newsletter
Application Form

I order the Petri Net Newsletter starting from January, ______ until recalled, connected with membership in the special interest group FG 0.0.1 Petri Nets and related system models of the Gesellschaft für Informatik e.V.

The annual fee is:

- EURO 10,00 for members of GI
- EURO 15,00 for non-members of GI

<table>
<thead>
<tr>
<th>title/Dr./Ms</th>
<th>last name</th>
<th>first name/initials</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>address</th>
<th>O affiliation / O private</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>phone</th>
<th>fax</th>
<th>email</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>GI membership number (members only)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>place and date</th>
<th>signature</th>
</tr>
</thead>
</table>

Form of payment:

Transfer by account:

Accounts of Gesellschaft für Informatik e.V.:

- Post office account no. 1981 39502, Postbank Köln
- Bank account no. 46 581, Sparkasse Bonn (Bank code 38050000)

If a foreign bank is charged with the transfer, the invoice amount increases by EURO 8,00 bank fee.

Foreigners can also pay by credit card (not for German members).

Diners Club, Visa Card and Master Card are accepted.

<table>
<thead>
<tr>
<th>credit card company</th>
<th>card number</th>
<th>expiration date</th>
</tr>
</thead>
</table>