Oclets – scenario-based modeling with Petri nets

Dirk Fahland*

Humboldt-Universität zu Berlin, Institut für Informatik,
Unter den Linden 6, 10099 Berlin, Germany,
fahland@informatik.hu-berlin.de

Abstract. Scenario-based specifications are used for modeling highly-complex, distributed systems in terms of partial runs (*scenarios*) the system shall have. But it is difficult to derive an implementing, operational model from a given set of scenarios, especially if concepts like *anti-scenarios* which must not occur are used. In this paper, we present a novel model for scenario-based specifications with Petri nets including anti-scenarios; we provide an operational semantics for our model.

1 Operational semantics for scenario-based specifications

The paradigm of scenarios is widely accepted in protocol specifications using *message-sequence charts* (MSCs); *behavior* of highly-complex distributed systems is decomposed into reasonably sized artifacts called scenarios. Some classes of MSC specifications can be transformed into Petri nets [7], but usually an implementation has to be checked against an MSC specification. *Life-sequence charts* (LSCs) [5] extend the MSC paradigm by adding behavioral preconditions, anti-scenarios, and annotations to scenarios and single actions for enforcing their occurrence in the system. LSCs have a trace-based semantics (a set of charts accepts or rejects an execution trace) but, to our knowledge, there exists no complete operational semantics for the entire LSC language. Like for MSCs, subclasses of LSCs can be transformed into automata [4].

In this paper, we present an extension of Petri nets that with the key concepts of LSCs. Our model has operational semantics: For every set of scenarios, we can compute the branching process that implements the specification, extending the formal approach of [1]. Due to the very nature of Petri nets, we also introduce the notion of a local resource to LSC-style scenario-based specifications. Compared to other approaches for scenario-based specifications with Petri nets [6], we contribute the *anti-scenario* which explicitly forbids certain behavior in the system. In [3], we explained how our approach can be used for modeling adaptive processes in disaster management.

We will first sketch the key concepts of our approach in Sect. 2. We then explain our ideas related to a formal semantics for our model in Sect. 3 which we close with an outlook on future work. We assume the reader to be familiar with Petri nets and their branching time semantics in terms of branching processes; Esparza et al give a good introduction to these concepts in [2].

* The author's work is funded by the DFG-Graduiertenkolleg 1324 "METRIK".

2 Oclets - scenario-based specifications with Petri nets

A *scenario* specifies a possible course of (future) actions and the therein involved resources in the context of a larger system. Whether a scenario suits a given situation can be subject to further conditions. In our case, we conceive and formalize a scenario as a partial, partially ordered run (a labeled causal net) with a *behavioral* precondition. We define a *system model* as a set of scenarios describing sequentially connected, concurrent, mutually exclusive, and overlapping behavior. The *system behavior* shall be computed by composing its scenarios.

We formalize scenarios in our Petri net class of *oclets*. Let $Names = Actions \uplus Resources$ denote a set of labels.

Definition 1 (Oclet). *An* oclet $o = \langle P, T, F, \ell, pre, type \rangle$ *is a labeled, safe, elementary causal net* $\langle P, T, F, \ell \rangle$ *that labels places with resources and transitions with actions;* o *has a non-empty,* precondition $pre \subseteq (P \cup T)$*, that is causally closed (*$\forall x \in pre :: {}^\bullet x \subseteq pre$*), and a type* $\in \{normal, anti\}$*. The set* $(P \cup T) \setminus pre$ *is the* contribution *of* o*.*

A normal oclet describes a partial run that may occur in the system. An anti-oclet describes a partial run that may not be completed in the system; therefore an anti-oclet contributes exactly one place or transition (that must not occur). Figure 1 shows some (technical) example oclets. The system $\{o_1, \ldots, o_5\}$ shall yield the behavior that is formalized in the occurrence net β_5. The behavior of a set of oclets is constructed by repeatedly composing the oclets with a labeled occurrence net. An 'initial' occurrence net β_0 represents the initial state; composing β_i with an oclet o yields an occurrence net β_{i+1}.

Roughly, a normal oclet o is *composed* with a labeled occurrence net β, $\beta \oplus o$ by building the union of the nets, and merging two transitions (places) if they are labeled equally and have equally labeled predecessors. This is only allowed if o's precondition is found in β; all nodes of o's precondition are merged with nodes of β. To *compose* an anti-oclet o with β, $\beta \ominus o$, first compose o like for

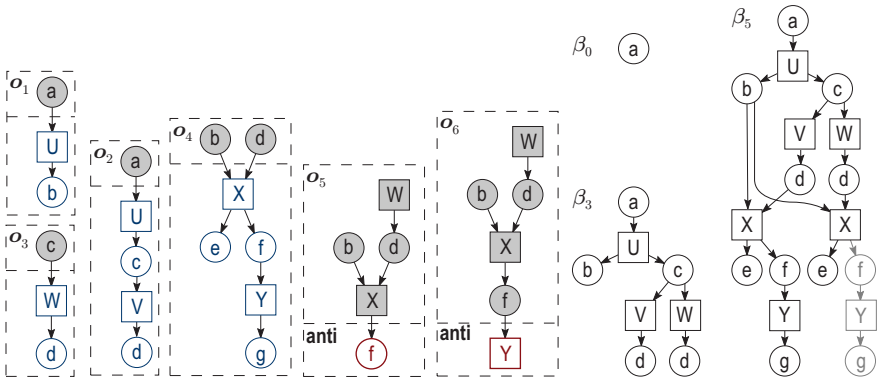

Fig. 1. Some example oclets $o_1, \ldots, o_6$ and three labeled occurrence nets $\beta_0, \beta_3, \beta_5$.

normal oclets, then remove the contribution of o and all successor nodes. Anti-oclets have priority: a node that was removed by an anti-oclet o^- is not added again by some other oclet o^+ as it is immediately removed by o^- again.

Consider the example of Fig. 1 with the initial occurrence net β_0 being a single place labeled a. Composing β_0 with oclet o_1, yields $\beta_1 := \beta_0 \oplus o_1$ which is isomorphic to o_1. $\beta_2 := \beta_1 \oplus o_2$ adds the post-place c to transition U and transition V with post-place d. In $\beta_3 := \beta_2 \oplus o_3$, transition W is added in conflict to V; see Fig. 1.

To compute $\beta_4 := \beta_3 \oplus o_4$, o_4 has to be added twice because there are two (conflicting) places d. Composing β_4 with anti-oclet o_5 removes f and successors Y and g from the branch that depends on W; the resulting occurrence net $\beta_5 := \beta_4 \ominus o_5$ is depicted in Fig. 1. Alternatively, composing with anti-oclet o_6 removes Y and successor g, but leaves f. The runs of β_5 are the runs of $\{o_1, \ldots, o_5\}$, the runs of β_6 are the runs of $\{o_1, \ldots, o_4, o_6\}$.

This informally sketched approach for scenario-based system specifications succeeds only if we can prove its formal consistency and show that branching processes (or rather a certain kind of labeled occurrence nets) are closed under our composition operations $\oplus$ and $\ominus$.

3 Formalizing oclets with canonically named nodes

Our oclet composition requires to ask frequently which nodes of an oclet o and an occurrence net β describe identical actions or resources, and, hence, must be merged. Formalizing this identity, and operations on labeled nets becomes tedious because two isomorphic nets may have disjoint, or overlapping sets of nodes. Identity can only be defined by relating labels of nodes to labels of neighboring nodes; this leads to graph isomorphism problems. Esparza and Heljanko use a formalization called *canonically named nodes* for formalizing branching processes of (safe) Petri nets [1]. In this section, we briefly sketch their key ideas and explain how we extend canonically named nodes for our model.

Canonically named nodes determine their identity by their labels and their predecessor: two nodes are identical if and only if they have identical labels and identical predecessors. The following formalization captures this *canonical identity*: The set $\mathcal{C}$ of *canonically named nodes* ($\mathcal{C}$-nodes) is defined inductively as the least set that contains $\langle a, \emptyset \rangle$ for every $a \in \mathit{Names}$ and if $x_1, \ldots, x_n \in \mathcal{C}$ and $a \in \mathit{Names}$ then $\langle a, \{x_1, \ldots, x_n\} \rangle \in \mathcal{C}$.

$\mathcal{C}$-nodes can be used as the base set of transitions and places of labeled Petri nets. A node $\langle \mathit{act}, X \rangle \in \mathcal{C}, \mathit{act} \in \mathit{Actions}$ is a *$\mathcal{C}$-transition* with label act, a node $\langle \mathit{res}, X \rangle \in \mathcal{C}, \mathit{res} \in \mathit{Resources}$ is a *$\mathcal{C}$-place* with label res. We use $\mathcal{C}$-nodes to formalize a specific class of labeled Petri nets.

Definition 2 ($\mathcal{C}$-net). *A labeled Petri net $N^{\mathcal{C}} = \langle P, T, F, \ell \rangle$ is a* $\mathcal{C}$-net *iff $P \subseteq \mathcal{C}$ are $\mathcal{C}$-places, $T \subseteq \mathcal{C}$ are $\mathcal{C}$-transitions, and for each $x := \langle a, X \rangle \in P \cup T$ holds:*

1. *if x is a $\mathcal{C}$-place, then X is a set of $\mathcal{C}$-transitions,*
2. *if x is a $\mathcal{C}$-transition, then X is a set of $\mathcal{C}$-places,*

3. $X \subseteq P \cup T$ *is the* preset *of* x*:* $y \in X$ *iff* $(y, x) \in F$*, and*
4. a *is the label of* x*:* $\ell(\langle a, X\rangle) = a$*.*

In a $\mathcal{C}$-net exist no two distinct, equally labeled nodes $\langle a, X\rangle, \langle a, Y\rangle$ with the same preset $X = Y$, this establishes the canonical identity of $\mathcal{C}$-nodes which we described above. This is a trivial mathematical consequence, but it has an interesting interpretation in branching processes: any two different actions (transitions) or resources (places) either have a different name, or a different causale. Esparza and Heljanko have shown that for this reason, $\mathcal{C}$-net structures are a good candidate to formalize branching processes (BP) of (safe) net systems [1], where the nodes of a net are labels to the nodes of the branching process.

Our oclet approach has a similar aim: construct branching-time artifacts that describe the behavior of a system. The difference is that we do not construct our artifacts from a net structure, but from oclets. Our construction does not only extend a branching process by adding a single transition (and its post-places) whenever the transition is enabled as in classical branching processes. The precondition of an oclet can be arbitrarily complex, and added nodes may have to be merged with the net. This means our formalization has to consider the causal structure of a labeled occurrence net and of an oclet *together*. To this end, we extend the $\mathcal{C}$-node approach of [1] as follows.

Operations on $\mathcal{C}$-nets and sets of $\mathcal{C}$-nodes The structure of a $\mathcal{C}$-net $N^{\mathcal{C}} = \langle P, T, F, \ell\rangle$ is completely encoded in its nodes, the information in its arcs F is redundant. Thus, the nodes $X_N^{\mathcal{C}} =_{\text{df}} P \cup T$ of a $\mathcal{C}$-net $N^{\mathcal{C}}$ are sufficient to reconstruct F and, hence, $N^{\mathcal{C}}$. Because any two isomorphic $\mathcal{C}$-nets are identical, each (normal) Petri net N has a unique, isomorphic $\mathcal{C}$-net $N^{\mathcal{C}}$ which is completely encoded in $X_N^{\mathcal{C}}$.

This greatly simplifies our composition operation: the union of two sets of $\mathcal{C}$-nodes 'merges' canonically identical nodes by their identity. If we consider the sets $X_{o_1}^{\mathcal{C}}$ and $X_{o_2}^{\mathcal{C}}$ of $\mathcal{C}$-nodes of o_1 and o_2 in Fig. 1, the composition $\beta_2^{\mathcal{C}} := (\beta_0^{\mathcal{C}} \oplus o_1^{\mathcal{C}}) \oplus o_2^{\mathcal{C}}$ can be rephrased as the union $X_{\beta_2}^{\mathcal{C}} = X_{\beta_0}^{\mathcal{C}} \cup X_{o_1}^{\mathcal{C}} \cup X_{o_2}^{\mathcal{C}}$. For instance, $\langle \mathsf{a}, \emptyset\rangle$ and $\langle \mathsf{U}, \{\langle \mathsf{a}, \emptyset\rangle\}\rangle$ occur both in $o_1^{\mathcal{C}}$ and $o_2^{\mathcal{C}}$. But this approach does not work for o_3; $o_3^{\mathcal{C}}$ contains $\langle \mathsf{c}, \emptyset\rangle$, while $\beta_2^{\mathcal{C}}$ contains $\langle \mathsf{c}, \{\langle \mathsf{U}, \{\langle \mathsf{a}, \emptyset\rangle\}\rangle\}\rangle$.

Our proposed solution is to introduce variables into nodes with empty pre-set, e.g. $\langle \mathsf{c}, v\rangle$ such that the minimal nodes of an oclet which constitute the begin of a scenario can be assigned to other 'compatible' nodes 'further down' the occurrence net during the composition.

Let *Var* denote an (infinite) set of variables. The set $\mathcal{A}$ of *canonically named abstract nodes* ($\mathcal{A}$-nodes) differs to $\mathcal{C}$ in its induction base: For every $a \in$ *Names* and every $v \in$ *Var*, $\langle a, v\rangle$ is an $\mathcal{A}$-node, and if $x_1, \ldots, x_n \in \mathcal{A}$ and $a \in$ *Names* then $\langle a, \{x_1, \ldots, x_n\}\rangle \in \mathcal{A}$. Correspondingly, the class of $\mathcal{A}$-nets can be defined; the variable takes the role of the empty pre-set, that is, a node $\langle a, v\rangle$ of an $\mathcal{A}$-net $N^{\mathcal{A}}$ has no predecessor in $N^{\mathcal{A}}$. Wlog. for all $\langle a_1, v_1\rangle, \langle a_2, v_2\rangle \in X_N^{\mathcal{A}}$ holds that $v_1 = v_2$ implies $a_1 = a_2$.

With this convention in mind, we transfer the pre-set notation ${}^\bullet(.)$ from $\mathcal{C}$-nodes (or Petri nets) to $\mathcal{A}$-nodes; we set ${}^\bullet\langle a, v\rangle =_{\text{df}} v$. This canonically lifts

all other notions like causal relation $\leq$, conflict $\sharp$, and concurrency $||$ from Petri nets and $\mathcal{C}$-nodes to $\mathcal{A}$-nodes. As a consequence, any two distinct nodes $\langle a_1, v_1\rangle, \langle a_2, v_2\rangle, v_1, v_2 \in \mathit{Var}$ are concurrent.

We introduce variables as place-holders for the pre-set of a $\mathcal{C}$-node. Thus an *assignment* α maps each variable v to a (possibly empty) set $\alpha(v)$ of $\mathcal{C}$-nodes, $\alpha : \mathit{Var} \to 2^{\mathcal{C}}$. If $x^{\mathcal{A}} \in \mathcal{A}$, then $x^{\mathcal{A}}[\alpha]$ denotes the $\mathcal{C}$-node that is obtained from $x^{\mathcal{A}}$ by simultaneously replacing every occurrence of each variable $v \in \mathit{Var}$ with $\alpha(v)$. This notion canonically lifts to sets $X^{\mathcal{A}} \subseteq \mathcal{A}$.

There is an important technical detail, that we have to consider: Let N be a safe, causal, labeled, elementary Petri net and let $X_N^{\mathcal{A}}$ be the corresponding set of $\mathcal{A}$-nodes of N. An assignment α is *feasible* on $X_N^{\mathcal{A}}$ iff for any two distinct minimal nodes $\langle a_1, v_1\rangle, \langle a_2, v_2\rangle \in X_N^{\mathcal{A}}$ holds: $\langle a_1, \alpha(v_1)\rangle \,||\, \langle a_2, \alpha(v_2)\rangle$. A feasible assignment guarantees that two concurrent nodes (like b and d of o_4 in Fig. 1) remain concurrent under the assignment.

We may now formalize our oclet composition operations.

Definition 3 (Enabling assignment). *Let $\beta^{\mathcal{C}}$ be a labeled $\mathcal{C}$-occurrence net, let $o^{\mathcal{A}}$ be an $\mathcal{A}$-oclet with precondition $\mathit{pre}^{\mathcal{A}}$. An α is* enabling for $o^{\mathcal{A}}$ in $\beta^{\mathcal{C}}$ *iff α is feasible and $\mathit{pre}^{\mathcal{A}}[\alpha] \subseteq X_\beta^{\mathcal{C}}$. Let $\mathit{enabled}(o^{\mathcal{A}}, \beta^{\mathcal{C}})$ denote the set of all enabling assignments for $o^{\mathcal{A}}$ in $\beta^{\mathcal{C}}$.*

Wlog. the set $\mathit{enabled}(o^{\mathcal{A}}, \beta^{\mathcal{C}})$ contains no two assignments α that differ only on variables which do not occur in $o^{\mathcal{A}}$.

As an example consider o_1 and β_0 of Fig. 1: $\mathit{pre}_{o_1}^{\mathcal{A}} = \{\langle \mathsf{a}, v_1\rangle\}$, $X_{\beta_0}^{\mathcal{C}} = \{\langle \mathsf{a}, \emptyset\rangle\}$. The assignment that maps v_1 to $\emptyset$ is enabling for o_1 in β_0. Oclet o_4 has two qualitatively different enabling assignments in β_3.

A further notion which we need for the composition is the *causal past* $\lfloor x \rfloor$ of a $\mathcal{C}$-node x with $\lfloor x \rfloor =_{\mathrm{df}} \{y \in \mathcal{C} \mid y \leq x\}$; this notion also lifts to sets of $\mathcal{C}$-nodes.

Definition 4 (Oclet composition). *Let $\beta^{\mathcal{C}}$ be a labeled $\mathcal{C}$-occurrence net. Let $o^{\mathcal{A}}$ be an $\mathcal{A}$-oclet with $\mathit{enabled}(o^{\mathcal{A}}, \beta^{\mathcal{C}}) = \{\alpha_1, \ldots, \alpha_k\}$.*

If $o^{\mathcal{A}}$ is a normal oclet, then the composition of $\beta^{\mathcal{C}}$ with $o^{\mathcal{A}}$ *yields the $\mathcal{C}$-net $\beta_2^{\mathcal{C}} =_{\mathrm{df}} \beta^{\mathcal{C}} \oplus o^{\mathcal{A}}$ with $X_{\beta_2}^{\mathcal{C}} =_{\mathrm{df}} X_\beta^{\mathcal{C}} \cup (X_o^{\mathcal{A}}[\alpha_1] \cup \ldots \cup X_o^{\mathcal{A}}[\alpha_k])$.*

If $o^{\mathcal{A}}$ is an anti-oclet with contribution $\{y_o\} = X_o^{\mathcal{A}} \setminus \mathit{pre}_o^{\mathcal{A}}$, then the composition of $\beta^{\mathcal{C}}$ with $o^{\mathcal{A}}$ *yields the $\mathcal{C}$-net $\beta_2^{\mathcal{C}} =_{\mathrm{df}} \beta^{\mathcal{C}} \ominus o^{\mathcal{A}}$ with $X_{\beta_2}^{\mathcal{C}} =_{\mathrm{df}} \{x \in X_\beta^{\mathcal{C}} \mid \lfloor x \rfloor \cap (y_o[\alpha_1] \cup \ldots \cup y_o[\alpha_k]) = \emptyset\}$.*

Consider $o_1^{\mathcal{A}}$ and $\beta_0^{\mathcal{C}}$ of our example; $X_{o_1}^{\mathcal{A}} = \{p_1^{\mathcal{A}}, t_1^{\mathcal{A}}, p_2^{\mathcal{A}}\}$ with $p_1^{\mathcal{A}} = \langle \mathsf{a}, v_1\rangle$, $t_1^{\mathcal{A}} = \langle \mathsf{U}, \{p_1^{\mathcal{A}}\}\rangle$, $p_2^{\mathcal{A}} = \langle \mathsf{b}, \{t_1^{\mathcal{A}}\}\rangle$. The enabling assignment $\{\alpha\} = \mathit{enabled}(o_1^{\mathcal{A}}, \beta_0^{\mathcal{C}})$ yields $X_{o_1}^{\mathcal{A}}[\alpha] = \{p_1^{\mathcal{C}}, t_1^{\mathcal{C}}, p_2^{\mathcal{C}}\}$ with $p_1^{\mathcal{C}} = \langle \mathsf{a}, \emptyset\rangle$ etc. Thus the composition $\beta_0^{\mathcal{C}} \oplus o_1^{\mathcal{A}}$ yields exactly $X_{o_1}^{\mathcal{A}}[\alpha]$, merging the two places labeled a.

The composition with an anti-oclet is formally more involved, but straight forward: All nodes of $\beta_5^{\mathcal{C}}$ in Fig. 1 including the greyly shaded ones constitute $\beta_4^{\mathcal{C}}$ where $o_5^{\mathcal{A}}$ has one enabling assignment $\{\alpha\} = \mathit{enabled}(o_5^{\mathcal{A}}, \beta_4^{\mathcal{C}})$ mapping the variables of $\langle \mathsf{b}, v_1\rangle$ and $\langle \mathsf{W}, v_2\rangle$ of $o_5^{\mathcal{A}}$ to $\{\langle \mathsf{U}, \{p_{\mathsf{a}}^{\mathcal{C}}\}\rangle\}$ and $\{\langle \mathsf{c}, \{t_{\mathsf{U}}^{\mathcal{C}}\}\rangle\}$ of $\beta_4^{\mathcal{C}}$, respectively. The contributed node of $o_5^{\mathcal{A}}$ is $y_{o_5} = \langle \mathsf{f}, \{t_{\mathsf{X}}^{\mathcal{A}}\}\rangle$; α maps y_{o_5} to the

right-most node $\langle \mathsf{f}, \{t^{\mathcal{C}}_{\mathsf{X},2}\}\rangle = y_{o_5}[\alpha]$ of $\beta^{\mathcal{C}}_4$. All nodes of $\beta^{\mathcal{C}}_4$ which have this node in their causal past are to be removed, i.e. $\langle \mathsf{f}, \{t^{\mathcal{C}}_{\mathsf{X},2}\}\rangle$ itself and all nodes reachable from it via the flow-relation. This results in $\beta^{\mathcal{C}}_5$.

With this formalization one can show that labeled $\mathcal{C}$-occurrence nets are closed under composition with $\oplus$ and $\ominus$. From the set theoretic definitions of $\ominus$ follows that $(\beta \ominus o_1) \ominus o_2 = (\beta \ominus o_2) \ominus o_1$ for any $\mathcal{C}$-occurrence net β and any two $\mathcal{A}$-anti-oclets o_1 and o_2. $(\beta \oplus o_1) \oplus o_2 = (\beta \oplus o_2) \oplus o_1$ for normal oclets o_1, o_2 holds only if o_2 does not introduce new enabling assignments for o_1. The behavior of a set of oclets is defined as its *$\mathcal{C}$-unfolding*:

Definition 5 ($\mathcal{C}$-unfolding). *Let O be a set of oclets with $\{o_1, \ldots, o_k\}$ and $\{o_{k+1}, \ldots, o_l\}$ being the normal oclets and the anti-oclets of O, respectively. Let β_0 be a $\mathcal{C}$-occurrence net. The fixed point of the sequence $\langle \beta_0, \beta_1, \beta_2, \ldots\rangle$ with $\beta_{i+1} =_{\mathrm{df}} (\beta_i \oplus o_1 \oplus \ldots \oplus o_k) \ominus o_{k+1} \ominus \ldots \ominus o_l$ is the* $\mathcal{C}$-unfolding *of O.*

Summary and Future Work. Definition 5 concludes the presentation of our basic model for scenario-based specifications with Petri nets. The presented expressive means allow specifying complex behavior in terms of partial runs which may or must not occur.

The basic model has already been implemented in the *Graphical Runtime EnvironmenT for Adaptive systems (GRETA)*. Next, we will introduce further LSC features like *hot* and *cold* annotations for specifying which actions must occur and which states are legal final states. Further, we plan to introduce the notion of an *interface* to specify system composition, and system interaction. Finally, the question which Petri net has the same behavior as a given set of oclets shall be addressed.

References

1. J. Esparza and K. Heljanko. *Unfoldings - A Partial-Order Approach to Model Checking.* Springer-Verlag, 2008.
2. J. Esparza, S. Römer, and W. Vogler. An Improvement of McMillan's Unfolding Algorithm. In *TACAS 1996*, volume 1055 of *LNCS*, pages 87–106. Springer-Verlag, 1996.
3. D. Fahland and H. Woith. Towards Process Models for Disaster Response. In *Proceedings of PM4HDPS 2008, co-located with BPM'08*, Milan, Italy, September 2008. Accepted.
4. D. Harel and H. Kugler. Synthesizing State-Based Object Systems from LSC Specifications. In *CIAA 2000*, volume 2088 of *LNCS*, pages 1–33. Springer-Verlag, 2001.
5. D. Harel and R. Marelly. *Come, Let's Play: Scenario-Based Programming Using LSC's and the Play-Engine.* Springer-Verlag New York, Inc., 2003.
6. R. Lorenz, R. Bergenthum, J. Desel, and S. Mauser. Synthesis of Petri Nets from Finite Partial Languages. In *ACSD 2007*, pages 157–166. IEEE Computer Society, 2007.
7. M. Mukund, K.N. Kumar, and P.S. Thiagarajan. Netcharts: Bridging the gap between HMSCs and executable specifications. In *CONCUR 2003*, volume 2761 of *LNCS*, pages 296–310. Springer-Verlag, 2003.