Partial Behavioural Models for Requirements and Early Design

Marsha Chechik, Greg Brunet*, Dario Fischbein**, and Sebastian Uchitel***

Abstract. In this paper, we first motivate and summarize our recent work on creation, management, and specifically merging of partial behavioural models, expressed as model transition systems. We then address two issues coming out of MMOSS discussions: alphabet embedding as an alternative to common observational refinement and the general notion of refinement.

1 Introduction and Goals of the Project

Event-based models such as Labeled Transition Systems (LTSs) [9] have been shown to be successful for modeling and reasoning about the behavior of software systems at the architectural level. These behavior models provide a basis for a wide range of successful automated analysis techniques, such as model-checking, animation, and simulation.

However, the adoption of behavior modeling and analysis technology by practitioners has been slow. Partly, this is due to the complexity of building behavioral models in the first place – behavior modeling remains a difficult, labor-intensive task that requires considerable expertise. In addition, and perhaps more importantly, the benefits of the analysis appear only at the end of a costly process of constructing a comprehensive behavior model. The reason for the latter is that traditional behavior models are required to be complete descriptions of the system behavior up to some level of abstraction, i.e., the transition system is assumed to completely describe the system behavior with respect to a fixed alphabet of actions. This completeness assumption is limiting in the context of software development process best practices which include iterative development, adoption of use-case and scenario-based techniques and viewpoint- or stakeholder-based analysis; practices which require modeling and analysis in the presence of partial information about system behavior.

Our aim is to address the limitations of existing behavior modeling approaches by shifting the focus from traditional behavior models to partial behavior models – models that are capable of distinguishing known behavior (both required and proscribed) from unknown behavior that is yet to be elicited. Our

* Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S2E4, {chechik,gbrunet}@cs.toronto.edu.
** Department of Computing, Imperial College, d.fischbein@doc.ic.ac.uk
*** Department of Computing, Imperial College, Department of Computing, University of Buenos and CONICET, s.uchitel@doc.ic.ac.uk.

overall aim is to develop sound theory, techniques and tools that i) facilitate the construction of partial behavior models through model synthesis, ii) enable early feedback through partial behavior model analysis, and iii) support incremental elaboration of partial models.

The rest of this paper is organized as follows. We give an overview of our work in Section 2. Section 3 provides the formal background required to follow the discussions in the next two sections, which address two technical points that came out of the discussions during the Dagstuhl week: the need for observational refinement rather than standard refinement and alphabet embedding (Section 4), and the notion of a minimum refinement step (Section 5). Section 6 concludes the paper.

2 MTS Merge: A Foundation for Model Construction and Elaboration

In this section, we give an informal overview of our approach. Section 3 gives formal definitions for some of these concepts. Further details are available in [2, 13, 1].

Our approach adopts Modal Transition Systems (MTSs) [11] as the basis for describing partial system behavior. MTSs have been extensively studied and provide a formal underpinning for program analysis [8]. MTSs are a natural extension to Labelled Transition Systems (LTSs), which have been proven to be successful for modeling and analyzing the behavior of systems at the architecture level. Systems are modeled as a set of components or sub-systems that communicate and synchronize to provide system-level behavior. Each component is described as a transition system where labels on transitions represent an interaction of the component with the environment. In MTSs, each transition can be either 'required' or 'maybe'. The later means that it is not yet certain if the interaction modelled by the transition is required or prohibited in the final system. An MTS with no maybe-transitions is a model that is fully defined up to its alphabet, and hence corresponds to an LTS.

MTS models come equipped with a definition of refinement that captures the notion of "more defined than" or "more information than". A refinement step corresponds intuitively to removing maybe transitions or replacing them with a required one[1]. Refinement can be shown to preserve temporal properties [2, 8]; hence, by refining an MTS, we are guaranteed that all properties that were true (false) in the partial model will continue to be so in the refinement. However, by iteratively refining an MTS into an LTS, all the temporal properties that are undecided in the partial model eventually become either true or false depending on the decisions made in the refinement process. An important practical result is that the value of a modal μ-calculus property (true, false, or undecided) in an MTS can be obtained by performing two checks on LTSs derived from the MTS [2, 8]. This means that two model-checks are sufficient to decide effectively whether a property holds or does not hold in all possible refinements of an MTS.

[1] In fact, refinement is more subtle. We discuss it in Section 5.

Although MTS models and refinement are an appropriate fit for incremental construction of models of software development artifacts, we identified that such models lack a specific concept that is necessary in the context of model elaboration, namely, *model merging*. Behaviour descriptions are typically provided by different stakeholders with different viewpoints [7], describing different, yet overlapping aspects [3] of the same system. How should these partial models be put together?

Composition of behaviour models is not a new idea [12, 6]; however, its main focus has been on *parallel* composition which describes how two *different* components work together. In the context of model elaboration, what we are interested in is composing two partial descriptions of the *same* component to obtain a more elaborate version of both original partial descriptions. We call this operation a *merge*.

The core concept underlying MTS merging is that of a common observational refinement, which ensures that the required and proscribed behaviour of the models to be merged is preserved. Furthermore, merge should result in a model that is as least refined as possible, while still preserving the behaviour of the models being merged. The notion of common refinement also entails a natural definition of consistency as the existence of such common refinement.

We have studied MTS merge and produced a number of theoretical results and algorithms that support MTS construction and elaboration [2, 13]. Our results build on extensive existing research on MTSs (e.g., [8, 11, 10]). We have developed algorithms [13] for checking consistency of MTSs, merging consistent MTSs, and for approximating the merge of MTSs when the "best" merge is not unique and thus cannot be obtained algorithmically. We have characterized merge in terms of 3-valued modal μ-calculus [2] and extended these results to temporal logics that are well suited for requirements specifications, such as the linear temporal logic of fluents (FLTL) [5]. In addition, because in practice MTS merging is likely to be combined with other operations over MTSs such as parallel composition, we have studied the algebraic properties of merging [2]. Finally, we have also started to study other refinement notions over MTSs, such as branching refinement [4].

3 Background

In this section, we give formal definitions of the key concepts that our work is based on: MTSs, alphabet, refinement, etc.

Definition 1. *Let* States *be a universal set of states, Act be a universal set of observable action labels, and $Act_\tau = Act \cup \{\tau\}$. An* LTS *is a tuple $P = (S, L, \Delta, s_0)$, where $S \subseteq$* States *is a finite set of states, $L \subseteq Act_\tau$ is a set of labels, $\Delta \subseteq (S \times L \times S)$ is a transition relation, and $s_0 \in S$ is the initial state. We use $\alpha P = L \setminus \{\tau\}$ to denote the communicating alphabet (vocabulary) of P.*

Definition 2. *An* MTS *M is a structure $(S, L, \Delta^r, \Delta^p, s_0)$, where $\Delta^r \subseteq \Delta^p$, (S, L, Δ^r, s_0) is an LTS representing* required *transitions of the system and*

Fig. 1. Example MTSs $\mathcal{I}$ and $\mathcal{J}$ and their minimal common refinements $\mathcal{K}$ and $\mathcal{L}$.

(S, L, Δ^p, s_0) is an LTS representing its possible *(but not necessarily required) transitions. We use $\alpha M = L \setminus \{\tau\}$ to denote the communicating alphabet of M.*

When depicting MTSs, we label states for reference and assume that they start in state 0. Maybe transitions are denoted with a question mark following the label, and transitions on sets are short for a single transition on every element of the set. For example, the MTSs $\mathcal{I}$ with alphabet $\{c\}$ and $\mathcal{J}$ with alphabet $\{a, b, c\}$ are shown in Figure 1.

Given an MTS $M = (S, L, \Delta^r, \Delta^p, s_0)$, we say that M transitions on ℓ through a required transition to M', denoted $M \xrightarrow{\ell}_{\mathrm{r}} M'$, if $M' = (S, L, \Delta^r, \Delta^p, s_0')$ and $(s_0, \ell, s_0') \in \Delta^r$, and M transitions through a possible transition, denoted $M \xrightarrow{\ell}_{\mathrm{p}} M'$, if $(s_0, \ell, s_0') \in \Delta^p$. Similarly, for $\gamma \in \{r, p\}$, we write $M \xrightarrow{\hat{\ell}}_{\gamma} M'$ to denote that either $M \xrightarrow{\ell}_{\gamma} M'$, or $\ell = \tau$ and $P = P'$, and we use $P \xRightarrow{\ell}_{\gamma} P'$ to denote $P (\xrightarrow{\tau}_{\gamma})^* \xrightarrow{\ell}_{\gamma} (\xrightarrow{\tau}_{\mathrm{r}})^* P'$.

We capture the notion of elaboration of a partial description into a more comprehensive one using refinement and observational refinement:

Definition 3. (Strong Refinement) *Let $\wp$ be the universe of all MTSs. N is a refinement of M, written $M \preceq N$, when $\alpha M = \alpha N$ and (M, N) is contained in some* refinement relation *$R \subseteq \wp \times \wp$, for which the following holds for all $\ell \in Act_\tau$ and all $(M, N) \in R$:*

$$1.\ (M \xrightarrow{\ell}_r M') \implies (\exists N' \cdot N \xrightarrow{\ell}_r N' \wedge (M', N') \in R)$$
$$2.\ (N \xrightarrow{\ell}_p N') \implies (\exists M' \cdot M \xrightarrow{\ell}_p M' \wedge (M', N') \in R)$$

Definition 4. (Observational Refinement) *N is an observational refinement of M, written $M \preceq_o N$, if $\alpha M = \alpha N$ and (M, N) is contained in some refinement relation $R \subseteq \wp \times \wp$ for which the following holds for all $\ell \in Act_\tau$:*

$$1.\ (M \xrightarrow{\ell}_r M') \implies (\exists N' \cdot N \xRightarrow{\hat{\ell}}_r N' \wedge (M', N') \in R)$$
$$2.\ (N \xrightarrow{\ell}_p N') \implies (\exists M' \cdot M \xRightarrow{\hat{\ell}}_p M' \wedge (M', N') \in R)$$

Two models are strongly (observationally) equivalent, denoted $\equiv$ ($\equiv_o$), if they strongly (observationally) refine each other. We denote by $M@X$ the result of restricting αM to X, i.e., replacing actions in $Act \backslash X$ with τ and reducing αM to X. An *implementation* of an MTS M is an *LTS* N that refines M.

We now review the process of merge. The intuition behind this process is to find a more precise system by combining what is known from two partial descriptions of that system. This is a process aimed at finding a common observational refinement, and may require human intervention [13].

Definition 5. *An MTS P is a* common refinement (CR) *of MTSs M and N if $\alpha P \supseteq (\alpha M \cup \alpha N)$, $M \preceq_o P@\alpha M$ and $N \preceq_o P@\alpha N$.*

We denote the set of CRs of models M and N by $\mathcal{CR}(M,N)$. Two MTSs, M and N, are *consistent* iff $\mathcal{CR}(M,N) \neq \emptyset$. In [13], it is argued that the merged model should not introduce unnecessary behaviours, and is therefore based on finding a *minimal common refinement.*

Definition 6. *An MTS P is a* minimal common refinement (MCR) *of MTSs M and N if $P \in \mathcal{CR}(M,N)$, $\alpha P = \alpha M \cup \alpha N$, and there is no MTS $Q \not\equiv_o P$ such that $Q \in \mathcal{CR}(M,N)$ and $Q@\alpha P \preceq_o P$.*

For example, the models $\mathcal{I}$ and $\mathcal{J}$ in Figure 1 have two incomparable minimal common refinements, $\mathcal{K}$ with alphabet $\{c\}$ and $\mathcal{L}$, also shown in Figure 1. We denote the set of MCRs of models M and N by $\mathcal{MCR}$, so $\mathcal{MCR}(\mathcal{I},\mathcal{J}) = \{\mathcal{K},\mathcal{L}\}$.

4 Alphabet Embedding

A major goal of our work is treatment of merge for models with different vocabularies. Such treatment allows integrating descriptions that have different scopes. Varying the scope of a description to include only the relevant aspects of the problem to be described makes model construction simpler. Consequently, it is common to find models describing different viewpoints provided by different stakeholders with different alphabets.

The way we have addressed merging models with different vocabularies is to define observational refinement and algorithms that construct models that are common observational refinements. A reasonable question to ask is why not unify vocabularies first, embedding each model into the unified vocabulary, and then merge the resulting systems which have the same vocabulary? Such an embedding may allow us to use the strong refinement and conjunction operator defined by Larsen [10] for MTSs with same alphabets. We first provide an informal argument as to why we believe such an embedding is not possible, and then show that even if it were possible, it does not resolve the major issues that merging models with different alphabets has.

Recall the models $\mathcal{I}$–$\mathcal{L}$ shown in Figure 1 and described in Section 3. Starting with a model $\mathcal{I}$ with an alphabet $\{c\}$, we would like to produce an embedding, $\mathcal{I}'$, with alphabet $\{a,b,c\}$, such that the minimal common refinements of $\mathcal{I}'$ and $\mathcal{J}$ are still $\mathcal{K}$ and $\mathcal{L}$. The embedding should reflect the fact that $\mathcal{I}$ does not say anything about labels a and b, hence the embedding $\mathcal{I}'$ should not require nor proscribe the occurrence of a and b events. In other words, all states of $\mathcal{I}'$ should always have maybe transitions on a and b.

A natural choice of embedding is to add transitions on the new labels from and to any state of $\mathcal{I}$, yielding the embedding $\mathcal{I}_1'$ (see Figure 2). The problem with $\mathcal{I}_1'$ is that it is inconsistent with $\mathcal{J}$. Note that the initial state of $\mathcal{I}_1'$ has an outgoing required transition on c while the initial state of $\mathcal{J}$ does not have possible transitions on c from it (i.e., it disallows c). Because of this, it is easy to see that there can be no common refinement of $\mathcal{I}_1'$ and $\mathcal{J}$.

Thus, the embedding of $\mathcal{I}$ should result in a model which in the initial state does not have a required c transition enabled. Yet, clearly, to preserve the semantics of $\mathcal{I}$, a state in which there is an enabled required c transition should

Fig. 2. Two candidate embeddings of an MTS $\mathcal{I}$.

be reachable. Furthermore, this state should be reachable only through maybe transitions on a and b.

Consider a second candidate embedding $\mathcal{I}_2'$ (see Figure 2) of $\mathcal{I}$. This model satisfies the criteria described above. However, note that $\mathcal{J}$ is a common refinement of $\mathcal{I}_2'$ and $\mathcal{J}$. Indeed, the combination of $\mathcal{J}$ and $\mathcal{I}_2'$ has lost the requirement that c must happen if a or b occur. Furthermore, $\mathcal{J}$ is less refined than $\mathcal{K}$ and $\mathcal{L}$. In conclusion, by embedding $\mathcal{I}$ into the alphabet $\{a, b, c\}$ as described by $\mathcal{I}_2'$, we have changed the semantics of merge.

We now address the general case. We shall refer to a state which has a required c transition enabled as a c state for brevity. We have already discussed why the initial state of the embedding of $\mathcal{I}$ is not a c state. We also know that the embedding must have a reachable c state. If that state is reachable in strictly more than one a or b step, then the embedding has a state which can be reached by one a or b step which proscribes c. This would be inconsistent with $\mathcal{J}$. Suppose then that all c states are reachable in exactly one a or b step. If any of these steps are required, then the embedding will rule out either $\mathcal{K}$ or $\mathcal{L}$. If neither are required, we have $\mathcal{I}_2'$ which is not appropriate either. Hence, we can conclude that it is not possible to embed $\mathcal{I}$ into $\{a, b, c\}$ while preserving the minimal common refinements of $\mathcal{I}$ and $\mathcal{J}$.

The above argument leads to the following theorem.

Theorem 1. *There does not exist an embedding function $f : MTS \times 2^{Act} \Rightarrow MTS$ that satisfies the condition that for all MTSs A and B, $\mathcal{MCR}(A, B) = \mathcal{MCR}(f(A, \alpha A \cup \alpha B), f(B, \alpha A \cup \alpha B))$.*

5 Refinement

In this section, we discuss refinement as an operation to support elaboration. Is the refinement really just removing a maybe transition or replacing it with a required one? Can every model be reached by refinement? Is there a notion of minimal (one-step) refinement? We discuss these issues below.

5.1 Increasing the Number of States in the Model

We begin by addressing a question raised at the workshop. How can refinement capture the elaboration process that starts with an MTS that knows nothing about its alphabet to an LTS of the system to be? For instance, how can an MTS such as that in Figure 3(a) ever be refined to describe a set of traces that requires more than one MTS state? The answer is simple. The MTS in

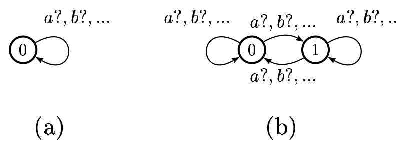

Fig. 3. Exploding states in MTS models.

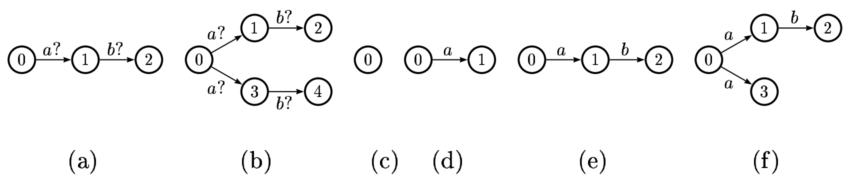

Fig. 4. (a)-(b) Two equivalent MTS models; (c)-(f) – their implementations.

Figure 3(a) is equivalent to the one in Figure 3(b) (which has two states) and similarly equivalent to an MTS with an arbitrary number of states!

States can be replicated even when the original system has no loops. Consider Figure 4(a). This model is equivalent to the one in Figure 4(b). In general, it is easy to generate equivalent MTS by simply replicating branches of the MTS and introducing non-determinism.

Figure 4(a) is an interesting model to use to address some erroneous intuitions that one might have regarding MTS refinement. Figure 4(a) seems to provide two decision points: one for a and one for b, which may lead us to think that it has only three implementations (see Figure 4(c), (d), and (e)). Yet since models in Figure 4(a) and Figure 4(b) are equivalent, it also clearly admits the implementation in Figure 4(f). Indeed, the intuition that refinement (in this example, strong refinement) is just making a maybe transition required or removing it is deceiving!

5.2 Are There Normal Forms for Refinement?

Another reasonable question to ask is whether it is possible to transform an MTS into its "normal form" such that all of the refinements of the former can be produced by a simple removal or replacement of maybe transitions with required ones of the latter.

On the first glance, it seems possible for some notions of refinement. For example, consider the model in Figure 5(a). Its maybe transition can be refined to false and to true, resulting in the models in Figure 5(b) and (c), respectively. However, the models in Figure 5(d) and (e) are also observational refinements of Figure 5(a). These additional models either sometimes provide a, or sometimes inevitably provide a. Note that all four models (Figure 5(b)-(e)) are incomparable using observational refinement of MTSs.

Now consider the model in Figure 5(f). This model is observationally equivalent to Figure 5(a) and can be refined into any of its refinements by making

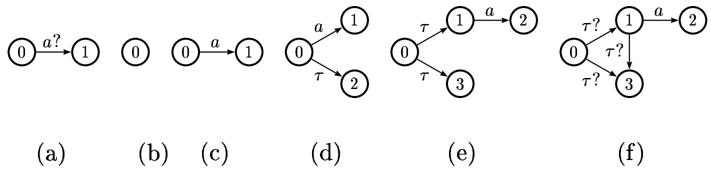

Fig. 5. (a): an MTS; (b)-(e) its refinements; (f) "normal form" of model in (a).

binary choices on τ transitions! This model can be thought of as a "normal form" of the model in Figure 5(a).

Unfortunately, normal forms do not seem to exist for MTSs that contain loops (which, of course, are present in the absolute majority of MTSs in practice). As we discussed in Section 5.1, loop unrollings can yield a system with potentially an unbounded number of states. So, a normal form for the model in Figure 3(a) would have to capture refinements that potentially have an unbounded number of states, which is clearly impossible.

5.3 Minimal Refinement Steps

The final question we consider is whether there is a notion of strict refinement steps that can be used to generate all possible refinements of an MTS.

We begin by giving a candidate definition of one-step (minimal) refinement:

Definition 7. *We say that an MTS N is a* one-step *(minimal) refinement of an MTS M, denoted $M \preceq_s N$, if $M \neq N$ and $\forall P \cdot M \preceq P \preceq N, P \equiv M \vee P \equiv N$.*

We can now reformulate the above question as the following proposition:

Proposition 1. *For all MTSs M and N: $M \preceq N \wedge M \not\equiv N \implies \exists P_1 \cdots P_n \cdot M \equiv_o P_1 \preceq_s P_2 \preceq_s \cdots \preceq_s P_n \equiv_o N$.*

We conjecture that the above proposition is false. Consider the MTS in Figure 6(a), which will be referred to as M. This model admits refinements that require a b if an a transition has been taken. The MTS in Figure 6(b) (referred to as N) is a refinement of Figure 6(a) which cannot be obtained as a sequence of the minimal refinement steps (at least as defined by Definition 7). Consider models P_i of the form shown in Figure 6(c). They are strict refinements of M, and N strictly refines them when $n > 2$. There is potentially an infinite number of such models, and therefore it is not true that $\exists P_1 \cdots P_n \cdot M \equiv_o P_1 \preceq_s P_2 \preceq_s \cdots \preceq_s P_n \equiv_o N$. Therefore, it is not possible to find a strict refinement of Figure 6(a) such that it is also a strict refinement of all MTS of the form of Figure 6(c). As a consequence, it is not possible to define a finite set of strict refinement steps that will allow us to generate each refinement of Figure 6(a). Note that this negative result does not prevent defining generator rules, as discussed above, that will produce all refinements of an MTS.

To summarize, in this section we looked at operations associated with refinement. Specifically, we (1) showed how to replicate MTS states, so that refinement

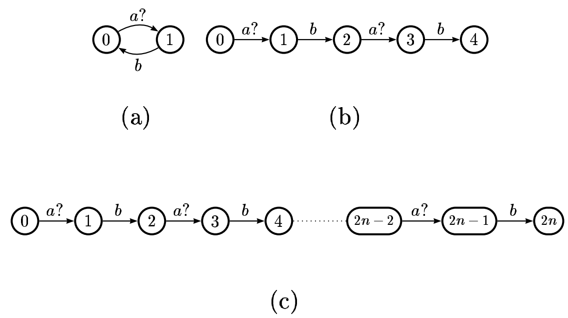

Fig. 6. (a): A model and (b)-(c): some of its refinements.

can be used to create arbitrarily large models; (2) showed that refinement is not strictly about removing maybe transitions or replacing them with required ones; (3) showed that while it is plausible to produce a set of operations on MTSs that will produce all possible refinements, they may not yield a guarantee of minimality or presence of normal forms.

6 Conclusion

In this paper, we gave a high-level overview of the process of merging partial behavioural models. We have also identified and discussed two problems that arose in this context during the MMOSS week: whether alphabet embedding provides an easy and natural way for merging MTSs with different vocabularies, and the intuition behind refinement and the "minimum refinement step".

In addition to the aspects described here, we are working on the problem of synthesizing partial behavioural models from a collection of positive and negative scenarios and temporal properties; giving user support for choosing the best common refinement out of a set of possible MCRs; helping deal with inconsistency resolution, and many others. Overall, our goal is to provide a set of techniques, tools and methodologies to support all aspects of engineering with partial behavioural models.

Acknowledgment. We thank the participants of the Dagstuhl MMOSS Seminar in August 2006 for many fruitful discussions. Many of the questions addressed here have been asked by Ed Brinkma. This research has been supported by NSERC, EPSRC PBM EP/C541133/1, SENSORIA and The Leverhulme Trust.

References

1. G. Brunet. "A Characterization of Merging Partial Behavioural Models". Master's thesis, University of Toronto, Department of Computer Science, January 2006.

2. G. Brunet, M. Chechik, and S. Uchitel. "Properties of Behavioural Model Merging". In *Proceedings of 14th International Symposium on Formal Methods (FM'06)*, volume 4085 of *LNCS*, pages 98–114. Springer, August 2006.
3. E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. "Progress on the State Explosion Problem in Model Checking". In R. Wilhelm, editor, *Informatics. 10 Years Back. 10 Years Ahead*, volume 2000 of *LNCS*, pages 176–194. Springer-Verlag, 2001.
4. D. Fischbein, S. Uchitel, and V. Braberman. "A Foundation for Behavioural Conformance in Software Product Line Architectures". In *Proceedings of ISTTA'06 Workshop on Role of Software Architecture for Testing and Analysis (ROSETEA'06)*, pages 39–48. ACM Press, 2006.
5. D. Giannakopoulou and J. Magee. "Fluent Model Checking for Event-Based Systems". In *Proceedings of the 9th joint meeting of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'03)*, pages 257–266. ACM Press, September 2003.
6. C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, New York, 1985.
7. A. Hunter and B. Nuseibeh. "Managing Inconsistent Specifications: Reasoning, Analysis and Action". *ACM Transactions on Software Engineering and Methodology*, 7(4):335–367, October 1998.
8. M. Huth, R. Jagadeesan, and D. A. Schmidt. "Modal Transition Systems: A Foundation for Three-Valued Program Analysis". In *Proceedings of 10th European Symposium on Programming (ESOP'01)*, volume 2028 of *LNCS*, pages 155–169. Springer, 2001.
9. R. Keller. "Formal Verification of Parallel Programs". *Communications of the ACM*, 19(7):371–384, 1976.
10. K.G. Larsen, B. Steffen, and C. Weise. "A Constraint Oriented Proof Methodology based on Modal Transition Systems". In *Tools and Algorithms for Construction and Analysis of Systems (TACAS'95)*, LNCS, pages 13–28. Springer, May 1995.
11. K.G. Larsen and B. Thomsen. "A Modal Process Logic". In *Proceedings of 3rd Annual Symposium on Logic in Computer Science (LICS'88)*, pages 203–210. IEEE Computer Society Press, 1988.
12. R. Milner. *Communication and Concurrency.* Prentice-Hall, New York, 1989.
13. S. Uchitel and M. Chechik. "Merging Partial Behavioural Models". In *Proceedings of 12th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 43–52, November 2004.