A Constraint Oriented Proof Methodology based on Modal Transition Systems

Kim G. Larsen*
BRICS†
Aalborg Univ., Denmark,

Bernhard Steffen
FB Math. und Informatik,
Univ. of Passau, Germany,

Carsten Weise‡
LS für Informatik I,
Aachen Univ., Germany

Abstract

In this paper, we present a constraint-oriented state-based proof methodology for concurrent software systems which exploits compositionality and abstraction for the reduction of the verification problem under investigation. Formal basis for this methodology are Modal Transition Systems allowing loose state-based specifications, which can be refined by successively adding constraints. Key concepts of our method are *projective views*, *separation of proof obligations*, *Skolemization* and *abstraction*. The method is even applicable to real time systems.

1 Introduction

The use of formal methods and in particular formal verification of concurrent systems, interactive or fully automatic, is still limited to very specific problem classes. For state-based methods this is mainly due to the state explosion problem: the state graph of a concurrent systems grows exponentially with the number of its parallel components, leading to an unmanageable size for most practically relevant systems. Consequently, several techniques have been developed to tackle

*This author has been partially supported by the European Communities under CONCUR2, BRA 7166.

†Basic Research in Computer Science, Centre of the Danish National Research Foundation.

‡Most of the work of this author was done during a visit to Aalborg University, partially supported by BRICS

this problem. Here we focus on the four main streams and do not discuss the flood of very specific heuristics. Most elegant and ambitious are *compositional* methods (e.g. [ASW94, CLM89, GS90][1]), which due to the nature of parallel compositions are unfortunately rarely applicable. *Partial order* methods try to avoid the state explosion problem by suppressing unnecessary interleavings of actions [GW91, Val93, GP93]. But also these methods, which are extremely successful in special cases, do not work in general. In practice, *Binary Decision Diagram*-based codings of the state graph are successfully applied to an interesting class of systems, see e.g. [Br86, BCMDH90, EFT91]. These codings of the state graph do not explode directly, but they may explode during verification, and it is not yet fully clear when this happens. All these techniques can be accompanied by *abstraction*: depending on the particular property under investigation, systems may be dramatically reduced by suppressing details that are irrelevant for verification, see e.g. [CC77, CGL92, GL93]. Summarizing, all these methods cover very specific cases, and there is no hope for a uniform approach. Thus more application specific approaches are required, extending the practicality of formal methods.

In this paper, we present a constraint-oriented state-based proof methodology for concurrent software systems which exploits compositionality and abstraction for the reduction of the verification problem under investigation. Formal basis for this methodology are Modal Transition Systems (MTS) [LT88] allowing loose state-based specifications, which can be refined by successively adding constraints. In particular, this allows extremely fine-granular specifications, which are characteristic for our approach: each aspect of a system component is specified by a number of independent constraints, one for each parameter configuration. This leads to a usually infinite number of extremely simple constraints which must all be satisfied by a corresponding component implementation. Beside exploiting compositionality in the standard (vertical) fashion, this extreme component decomposition also supports a horizontally compositional approach, which does not only separate proof obligations for subcomponents or subproperties but also for the various parameter instantiations. This is the key for the success of the following three step reduction, which may reduce even a verification problem for infinite state systems to a small number of automatically verifiable problems about finite state systems:

- *Separating the Proof Obligations.* Sections 3 and 4 present a proof principle justifying the separation and specialization of the various proof obligations, which prepare the ground for the subsequent reduction steps.

- *Skolemization.* The separation of the first step leaves us with problems smaller in size but larger in number. Due to the nature of their origin, these

[1]In contrast to the first reference, the subsequent two papers address compositional reduction of systems rather than compositional verification.

problems often fall into a small number of equivalence classes requiring only one prototypical proof each.

- *Abstraction.* After the first two reduction steps there may still be problems with infinite state graphs. However, the extreme specialization of the problem supports the power of abstract interpretation, which finally may reduce all the proof obligations to finite ones.

Our proof methodology is not complete, i.e., there is neither a guarantee for the possibility of a finite state reduction nor a straightforward method for finding the right amount of separation for the success of the succeeding steps or the adequate abstraction for the final verification. Still, as should be clear even from the simple accompanying example in the paper, there is a large class of problems and systems, where the method can be applied quite straightforwardly. Typical examples are systems with limited data dependence, like the one proposed by Leslie Lamport and Manfred Broy for a recent Dagstuhl Seminar as a mean to evaluate methods [BL93]. Of course, the more complex the system structure the more involved will be the required search of appropriate granularity and abstraction.

Whereas complex data dependencies may exclude any possibility of 'horizontal' decomposition, our approach elegantly extends to real time systems, even over a dense time domain. In fact, we will show that this extension does not affect the possibility of a finite state reduction. Even better, the resulting finite state problems can be automatically verified using the EPSILON verification system. All this is illustrated using a simple example of pipelined buffers.

The next section recalls the basic theory of Modal Transition Systems (MTS), which we use for system specification. Section 3 presents our notion of projective views and discusses the first reduction step. The subsequent two sections are devoted to the second and third reduction step, while Section 6 sketches the extension of our method to real time systems over a dense time domain. Finally, Section 7 summarizes our conclusion and directions to future work.

2 Modal Transition Systems

In this section we give a brief introduction to the existing theory of modal transition systems. We assume familiarity with CCS. For more elaborate introductions and proofs we refer the reader to [LT88, HL89, Lar90].

When specifying reactive systems by traditional Process Algebras like e.g. CCS [Mil89], one defines the set of action transitions that can be performed (or observed) in a given system state. In this approach, any valid implementation *must* be able to perform the specified actions, which often constrains the set of possible implementations unnecessarily. One way of improving this situation within the framework of operational specification is to allow specifications where one can explicitly distinguish between transitions that are *admissible* (or allowed) and those that are *required*. This distinction allows a much more flexible specification and a much more generous notion of implementation, and therefore improves the practicality of the operational approach. Technically, this is made precise through the following notion of *modal transition systems*:

Definition 2.1 *A modal transition system is a structure* $\mathcal{S} = (\Sigma, A, \longrightarrow_\Box, \longrightarrow_\Diamond)$, *where* Σ *is a set of states,* A *is a set of actions and* $\longrightarrow_\Box, \longrightarrow_\Diamond \subseteq \Sigma \times A \times \Sigma$ *are transition relations, satisfying the consistency condition* $\longrightarrow_\Box \subseteq \longrightarrow_\Diamond$. □

Intuitively, the requirement $\longrightarrow_\Box \subseteq \longrightarrow_\Diamond$ expresses that anything which is required should also be allowed hence ensuring the consistency of modal specifications. When the relations $\longrightarrow_\Box$ and $\longrightarrow_\Diamond$ coincide, the above definition reduces to the traditional notion of labelled transition systems.

Syntactically, we represent modal transition systems by means of a slightly extended version of CCS. The only change in the syntax is the introduction of two prefix constructs $a_\Box.P$ and $a_\Diamond.P$ with the following semantics: $a_\Diamond.P \xrightarrow{a}_\Diamond P$, $a_\Box.P \xrightarrow{a}_\Box P$ and $a_\Box.P \xrightarrow{a}_\Diamond P$. The semantics for the other constructs follow the lines of CCS in the sense that each rule has a version for $\longrightarrow_\Box$ and $\longrightarrow_\Diamond$ respectively. We will call this version of CCS *modal* CCS.

As usual, we consider a design process as a sequence of *refinement steps* reducing the number of possible implementations. Intuitively, our notion of when a specification S refines another (weaker) specification T is based on the following simple observation. Any behavioural aspect *allowed* by a S should also be allowed by T; and dually, any behavioural aspect which is already guaranteed by the weaker specification T must also be guaranteed by S. Using the derivation relations $\longrightarrow_\Box$ and $\longrightarrow_\Diamond$ this may be formalized by the following notion of *refinement*:

Definition 2.2 *A refinement* $\mathcal{R}$ *is a binary relation on* Σ *such that whenever* $S\,\mathcal{R}\,T$ *and* $a \in A$ *then the following holds:*

1. *Whenever* $S \xrightarrow{a}_\Diamond S'$, *then* $T \xrightarrow{a}_\Diamond T'$ *for some* T' *with* $S'\,\mathcal{R}\,T'$,
2. *Whenever* $T \xrightarrow{a}_\Box T'$, *then* $S \xrightarrow{a}_\Box S'$ *for some* S' *with* $S'\,\mathcal{R}\,T'$.

S *is said to be a refinement of* T *in case* (S, T) *is contained in some refinement* $\mathcal{R}$. *We write* $S \lhd T$ *in this case.* □

Note that when we apply the above definition to traditional labelled transition systems (where $\longrightarrow = \longrightarrow_\Box = \longrightarrow_\Diamond$), we obtain the well–known notion of bisimulation [Par81, Mil89]. Using standard techniques, one straightforwardly establishes that $\lhd$ is a preorder preserved by all modal CCS operators.

$\lhd$ allows *loose* specifications. This important property, which can be best explained by looking at the 'weakest' specification $\mathcal{U}$ constantly allowing any action, but never requiring anything to happen. Operationally, $\mathcal{U}$ is completely defined by $\mathcal{U} \xrightarrow{a}_\Diamond \mathcal{U}$ for all actions a. It is easily verified that $S \lhd \mathcal{U}$ for any modal specification S.

Intuitively, S and T are *independent* if they are not contradictory, i.e. any action required by one is not constraint by the other. The following formal definition is due to the fact that for S and T to be *independent* all 'simultaneously'ṙeachable processes S' and T' must be indenpendent too:

Definition 2.3 *An independence relation $\mathcal{R}$ is a binary relation on Σ such that whenever $S \mathcal{R} T$ and $a \in A$ then the following holds:*

1. *Whenever $S \xrightarrow{a}_\Box S'$, there is a unique T' such that $T \xrightarrow{a}_\Diamond T'$ and $S' \mathcal{R} T'$,*
2. *Whenever $T \xrightarrow{a}_\Box T'$, there is a unique S' such that $S \xrightarrow{a}_\Diamond S'$ and $S' \mathcal{R} T'$,*
3. *Whenever $S \xrightarrow{a}_\Diamond S'$ and $T \xrightarrow{a}_\Diamond T'$ then $S' \mathcal{R} T'$.*

S and T are said to be independent *in case (S, T) is contained in some independence relation $\mathcal{R}$.* $\Box$

Note in particular that two specifications are independent if none of them requires any actions. Independence is important, as it allows to define conjunction on modal transition systems by:

$$\frac{S \xrightarrow{a}_\Box S' \quad T \xrightarrow{a}_\Diamond T'}{S \wedge T \xrightarrow{a}_\Box S' \wedge T'} \qquad \frac{S \xrightarrow{a}_\Diamond S' \quad T \xrightarrow{a}_\Box T'}{S \wedge T \xrightarrow{a}_\Box S' \wedge T'}$$

$$\frac{S \xrightarrow{a}_\Diamond S' \quad T \xrightarrow{a}_\Diamond T'}{S \wedge T \xrightarrow{a}_\Diamond S' \wedge T'}$$

Of course, $S \wedge T$ is always a well-defined modal specifications (i.e. any required transition is also allowed), and in fact, for independent arguments S and T it defines their *logical* conjunction:

Theorem 2.4 Let S and T be independent modal specifications. Then $S \wedge T \lhd S$ and $S \wedge T \lhd T$. Moreover, if $R \lhd S$ and $R \lhd T$ then $R \lhd S \wedge T$.

In order to compare specifications at different levels of abstraction, it is important to abstract from transitions resulting from internal communication. The way this is done for modal transition systems follows the lines of traditional labelled transition systems. That is, for a given modal transition system $\mathcal{S} = (\Sigma, A \cup \{\tau\}, \longrightarrow_\Box, \longrightarrow_\Diamond)$ we derive the modal transition system $\mathcal{S}_\varepsilon = (\Sigma, A \cup \{\varepsilon\}, \Longrightarrow_\Box$

$, \Longrightarrow_\Diamond)$, where $\overset{\varepsilon}{\Longrightarrow}_\Box$ is the reflexive and transitive closure of $\overset{\tau}{\longrightarrow}_\Box$, and where $T \overset{a}{\Longrightarrow}_\Box T', a \neq \varepsilon$, means that there exist T'', T''' such that

$$T \overset{\varepsilon}{\Longrightarrow}_\Box T'' \overset{a}{\longrightarrow}_\Box T''' \overset{\varepsilon}{\Longrightarrow}_\Box T'$$

The relation $\Longrightarrow_\Diamond$ is defined in a similar manner.

The notion of *weak refinement* can now be introduced as follows: S weakly refines T in $\mathcal{S}$, $S \trianglelefteq T$, iff there exists a refinement relation on $\mathcal{S}_\varepsilon$ containing S and T.

Weak refinement $\trianglelefteq$ essentially enjoys the same pleasant properties as $\triangleleft$: it is a preorder preserved by all modal CCS operators except $+$ [HL89]. Moreover, for ordinary labelled transition systems weak refinement reduces to the usual notion of weak bisimulation ($\approx$).

3 Projective Views

In the following, we present, motivate and clarify our proof methodology by means of a minimal example, which is just sufficient to explain the various phenomena. It should, however, be noted that the method scales up and has been successfully applied to Leslie Lamport's and Manfred Broy's Specification Problem of a recent Dagstuhl Seminar, which aimed at the evaluation of formal methods, see [BL93].

Consider the parallel system in Figure 1. Here two parameterized, disposable component media (supposed to transmit natural numbers) A and B are composed in parallel yielding a pipeline. Informally, the component A is supposed to input a natural number on port a, then output this number on port b after which it will terminate. The behaviour of B is similar. Using modal transition systems,

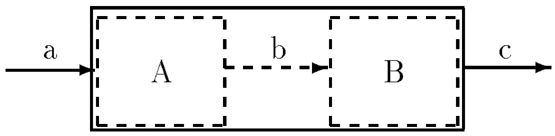


Figure 1: A Pipe Line of Two Disposable Media

the parallel system may be expressed as follows:

$$\Big(\underbrace{a_\Box x.\overline{b_\Box} x}_{A} \mid \underbrace{b_\Box x.\overline{c_\Box} x}_{B}\Big)\backslash\{b\}$$

The behaviour of A and B are given by the two infinite–width transition systems of Figure 2. However, rather than using these direct specifications of A and B we specify the two components behaviour using projective views A_n and B_n; one view for each possible natural number n. The projective view A_n specifies the

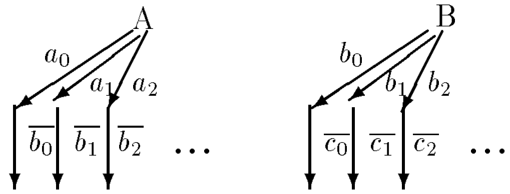
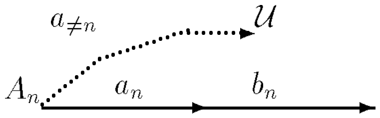


Figure 2: Behaviour of A and B.

constraints on the behaviour of the component A when focusing on transmission of the value n; this constraint can be expressed as the following modal transition system A_n (where we use solid lines for must- and dotted lines for may-transition):



Here $a_{\neq n}$ denotes all labels of the form a_m where $m \neq n$; also $\mathcal{U}$ denotes the universal modal transition system constantly allowing all actions. Note that this 'n-th view' imposes no constraint on the behaviour of A when transporting values different from n. The complete specification of the component A is the conjunction of all projective views[2] A_n. In fact it is easy to establish the following facts:

$$A \trianglelefteq \bigwedge_n A_n \quad \text{and} \quad \bigwedge_n A_n \trianglelefteq A \tag{1}$$

where A refers to the (infinite) transition system of Figure 2. Obviously, we may obtain similar projective views B_n for component B.

Let us now consider the problem of verifying that the overall system $\Big(A \,|\, B\Big)\backslash\{b\}$ is observationally equivalent to the system $C = a_\square x.\overline{c_\square} x$ (i.e. a slightly different disposable media). As A, B and C are standard transitions systems, i.e., everything allowed is also required, this problem is equivalent to showing

$$\Big(A \,|\, B\Big)\backslash\{b\} \;\trianglelefteq\; C$$

Thus (1), together with the observation that also C may be expressed as a conjunction of an infinite number of constraints C_n, leave us with the following refinement problem:

$$\Big(\bigwedge_n A_n \,|\, \bigwedge_n B_n\Big)\backslash\{b\} \;\trianglelefteq\; \bigwedge_n C_n \tag{2}$$

[2]Note that all the projective views of A are pairwise independent.

4 Sufficient Proof Condition

Due to the properties of conjunction (cf. Lemma 2.4) the proof of (2) can obviously be reduced to the verification of

$$\Big(\bigwedge_n A_n \mid \bigwedge_n B_n\Big)\backslash\{b\} \;\trianglelefteq\; C_n$$

for each natural n. Thus due to Lemma 2.4 and the fact that $\trianglelefteq$ is preserved by parallel composition and restriction, it suffices to prove

$$\forall n \in N.\ \Big(A_n \mid B_n\Big)\backslash\{b\} \;\trianglelefteq\; C_n \tag{3}$$

There is a general proof principle behind this reduction: in order to conclude:

$$\Big(\bigwedge_{i\in I_1} A_i^1 \mid \ldots \mid \bigwedge_{i\in I_k} A_i^k\Big)\backslash L \;\trianglelefteq\; \bigwedge_{j\in I} C_j$$

it suffices for each $j \in I$ to establish:

$$\Big(\bigwedge_{i\in I_{1,j}} A_i^1 \mid \ldots \mid \bigwedge_{i\in I_{k,j}} A_i^k\Big)\backslash L \;\trianglelefteq\; C_j$$

where $I_{l,j} \subseteq I_l$ for each $l = 1 \ldots k$.

Of course, in general the power of this proof principle strongly depends on a good choice of the $I_{l,j}$, which was trivial in our example.

5 Skolemization and Abstraction

So far we have reduced the overall verification problem of (2) to that of (3). At first sight this doesn't seem much of a reduction as (3) requires a refinement proof to be established for each natural number. Fortunately, these proofs are not really sensitive to the actual value of the natural number n. Letting k be an arbitrary natural number (or a Skolem constant) it suffices to prove:

$$\Big(A_k \mid B_k\Big)\backslash\{b\} \;\trianglelefteq\; C_k \tag{4}$$

in order to infer (3). Thus we are now left with the problem of establishing a *single* refinement. But still, though finite *state* the specifications A_k and B_k both have infinitely many *transitions* (as $a_{\neq k}$ is an inifinite label set). This problem can be overcome using *abstraction* (or *factorization*) with respect to an appropriate equivalence relation.

Definition 5.1 *Let $\mathcal{S} = (\Sigma, S, \longrightarrow_\Box, \longrightarrow_\Diamond)$ be a modal transition system, let $\equiv_\Sigma$ and $\equiv_S$ be equivalence relations on Σ and S, and let $\Sigma^\equiv$ and $S^\equiv$ be the sets of equivalence classes. Then* the factorization of $\mathcal{S}$ *is the modal transition system $\mathcal{S}^\equiv = (\Sigma^\equiv, S^\equiv, \longrightarrow'_\Box, \longrightarrow'_\Diamond)$, where $\longrightarrow'_\Box, \longrightarrow'_\Diamond$ are defined as follows:*

$$\frac{s \xrightarrow{a}_\Box s'}{[s]^\equiv \xrightarrow{[a]^\equiv}'_\Box [s']^\equiv} \qquad \frac{s \xrightarrow{a}_\Diamond s'}{[s]^\equiv \xrightarrow{[a]^\equiv}'_\Diamond [s']^\equiv}$$

Equivalence relations $\equiv_\Sigma$ and $\equiv_S$ are called compatible *with the modal transition system $\mathcal{S}$ iff for all $a \equiv_\Sigma b, s \equiv_S t, s' \equiv_S t'$:*

$$s \xrightarrow{a}_\Box s' \quad \textit{iff} \quad t \xrightarrow{b}_\Box t' \qquad \textit{and} \qquad s \xrightarrow{a}_\Diamond s' \quad \textit{iff} \quad t \xrightarrow{b}_\Diamond t'$$

For compatible equivalence relations, the following reduction lemma is straightforward:

Lemma 5.2 *Let $S^\equiv$ and $T^\equiv$ be processes in the factorization $\mathcal{S}^\equiv$ of $\mathcal{S}$ with respect to compatible equivalence relations $\equiv_\Sigma$ and $\equiv_S$. Then we have for arbitrary representatives S of $S^\equiv$ and T of $T^\equiv$:*

$$S^\equiv \trianglelefteq T^\equiv \quad \textit{implies} \quad S \trianglelefteq T$$

This Lemma allows us to reduce verification problems for infinite systems to problems for finite systems, as soon as an appropriate factorization can be found.

For our example, let us consider the equivalence relation $\equiv$ defined by $x_k \equiv x_k$ and $x_i \equiv x_j$ whenever $i, j \neq k$, where x ranges over $\{a, b, c\}$. Obviously, $\equiv$ is compatible with the underlying transition system. Thus the verification of (2) can further be reduced to the refinement proof between the finite $\equiv$–abstracted versions of A_k, B_k and C_k

$$\Big(A_k{}^\equiv \mid B_k{}^\equiv \Big) \backslash \{b\} \;\trianglelefteq\; C_k{}^\equiv \tag{5}$$

which can easily be done by means of the automatic verification tool TAV.

6 Specifications with Time

The above example can be extended to deal with real time. For the specification we use Wang Yi's Timed CCS (see [Yi91]) together with modal specifications. For details on these so called *Timed Modal Specifications* see [CGL93]. This method can be used with any totally ordered time domain, while in the following we will assume the positive real numbers.

The passing of time is modelled by a delay action $\varepsilon(d)$, where d is a positive real number. The intuitive meaning of such a delay is that d time units pass until the end of this action. Normal actions are enabled immediately, and can be taken at any time. As an example, the process $a_{\Box}x.\varepsilon(2).\overline{b_{\Box}}x$ can execute $a_{\Box}x$ at any time. Thereafter it must delay for at least two time units before it can engage in $b_{\Box}x$.

Further we assume *maximal progress*, i.e. a communication must be performed as soon as possible. Putting $a_{\Box}x.\varepsilon(2).\overline{b_{\Box}}x$ in parallel with $\overline{a_{\Box}}x.\varepsilon(3).b_{\Box}x$ would force the communication via channel a to take place immediately, and the communication via channel b to happen after exactly three time units.

For our specification, the macro $a[l,u]$ is convenient, where a is an action and l,u are real numbers with $l < u$. The intuition is that a process $a[l,u].P$ *may* enable a after l time units and *must* enable a after u time units. In other words, communication via a may be possible after at least l time units, and will be possible at any time after u time units. This macro is defined as $a[l,u].P = (\varepsilon(l).a_{\Diamond} + \varepsilon(u).a_{\Box}).P$.

Let d be a fixed real number. Then we specify a timed process $A(d)$, which reads port a and subsequently outputs its input onto port b within d time units, by $a_{\Box}x.\overline{b}x[0,d]$.Note that this is a timed version of process A. The same construction gives timed versions $B(d)$ and $C(d)$ of B and C.

We are now going to establish that a 'pipeline' with two components with delay d should not be slower than one component with delay $2d$, i.e.

$$\Big(A(d) \mid B(d)\Big) \setminus \{b\} \;\trianglelefteq\; C(2d)$$

The same method as in the untimed case reduces the situation to

$$\Big({A_k}^{\equiv}(d) \mid {B_k}^{\equiv}(d)\Big) \setminus \{b\} \;\trianglelefteq\; {C_k}^{\equiv}(2d)$$

for a Skolem constant k and the equivalence relation of the previous section. Now, given a specific value for d this proof can be carried out using the Epsilon tool (see [CGL93]), which treats real valued timer domains by means of the clock region automaton technique (see [AD94] for details). This technique relies on integer values for all explicit timer constants in the specification, which can be achieved by multiplication with an appropriate constant in most applications. As all timer constants are multiplied by the same constant, this does not affect the principle behaviour of the system. In our example, the obvious choice for this constant is $1/d$, leaving us with the following refinement problem

$$\Big({A_k}^{\equiv}(1) \mid {B_k}^{\equiv}(1)\Big) \setminus \{b\} \;\trianglelefteq\; {C_k}^{\equiv}(2)$$

which can be solved using Epsilon.

Note that this proof indeed covers the statement for any d. Thus even in the presence of real time, the original verification problem is reduced to a very simple, automatically solvable problem.

7 Conclusion and Future Work

We have introduced a new constraint-oriented method for the (automated) verification of concurrent systems. Key concepts of our 'divide and conquer' method are *projective views*, *separation of proof obligations*, *Skolemization* and *abstraction*, which together support a drastic reduction of the complexity of the relevant subproblems. Of course, our proof methodology does neither guarantee the possibility of a finite state reduction nor a straightforward method for finding the right amount of separation or the adequate abstraction. Still, there is a large class of problems and systems, where the method can be applied quite straightforwardly. Typical examples are systems with limited data dependence. Whereas involved data dependencies may exclude any possibility of 'horizontal' decomposition, our approach elegantly extends to real time systems, even over a dense time domain. In fact, the resulting finite state problems can be automatically verified using the EPSILON verification system. All this has been illustrated using a simple example of pipelined buffers. Our experience indicates that our method scales up to practically relevant problems.

Beside further case studies and the search for good heuristics for proof obligation separation and abstraction, we are investigating the limits of tool support during the construction of constraint based specifications and the application of the three reduction steps. Whereas support by graphical interfaces and interactive editors is obvious and partly implemented in the META-Framework, a management system for synthesis, analysis and verification currently developed at the university of Passau, the limits of consistency checking and tool supported search for adequate separation and abstraction are still an interesting open research topic.

References

[ASW94] H. Andersen, C. Stirling, G. Winskel. A Compositional Proof System for the Modal Mu-Calculus. in: Proceedings LICS, 1994.

[AD94] R. Alur, D.L. Dill. A Theory of Timed Automata. in: Theoretical Computer Science Vol. 126, No. 2, April 1994, pp. 183-236.

[BL93] M. Broy, L. Lamport. Specification Problem. Case study for the Dagstuhl Seminar 9439, 1994.

[Br86] R. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. in: IEEE Transactions on Computation, 35 (8). 1986.

[BCMDH90] J. Burch, E. Clarke, K. McMillan, D. Dill, L. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. in: Proceedings LICS'90.

[CGL93] K. Čerāns, J.C. Godesken, K.G. Larsen. Timed Modal Specification - Theory and Tools. in: C. Courcoubetis (Ed.), Proc. 5th Int. Conf. on Computer Aided Verification (CAV '93), Elounda, Greece, June/July 1993. LNCS 697, Springer Berlin 1993, pp. 253–267.

[CGL92] E. Clarke, O. Grumber, D. Long. Model Checking and Abstraction. in: Proceedings XIX POPL'92.

[CLM89] E. Clarke, D. Long, K. McMillan. Compositional Model Checking. in: Proceedings LICS'89.

[CC77] R. Bryant. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction and Approximation of Fixpoints. in: Proceedings POPL'77.

[EFT91] R. Enders, T. Filkorn, D. Taubner. Generating BDDs for Symbolic Model Checking in CCS. in: Proceedings CAV'91, LNCS 575, 1991, pp. 203–213

[GW91] P. Godefroid, P. Wolper. Using Partial Orders for the Efficient Verification of Deadlock Freedom and Safety Properties. in: Proceedings CAV'91, LNCS 575, 1991, pp. 332–342.

[GP93] P. Godefroid, D. Pirottin. Refining Dependencies Improves Partial-Order Verification Methods. in: Proceedings CAV'93, LNCS 697, 1991, pp. 438–449.

[GL93] S. Graf, C. Loiseaux. Program Verification using Compositional Abstraction. in: Proceedings FASE/TAPSOFT'93.

[GS90] S. Graf, B. Steffen. Using Interface Specifications for Compositional Minimization of Finite State Systems. in: Proceedings CAV'90.

[HL89] H. Hüttel and K. Larsen. The use of static constructs in a modal process logic. Proceedings of Logic at Botik'89. LNCS 363, 1989.

[Lar90] K.G. Larsen. Modal specifications. In: Proceedings of Workshop on Automatic Verification Methods for Finite State Systems LNCS 407, 1990.

[LT88] K. Larsen and B. Thomsen. A modal process logic. In: Proceedings LICS'88, 1988.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[Par81] D. Park. Concurrency and automata on infinite sequences. In P. Deussen (ed.), 5th GI Conference, LNCS 104, pp. 167–183, 1981.

[Val93] A. Valmari. On-The-Fly Verification with Stubborn Sets. in: C. Courcoubetis (Ed.), Proc. 5th Int. Conf. on Computer Aided Verification (CAV '93), Elounda, Greece, June/July 1993. LNCS 697, Springer Berlin 1993, pp. 397–408.

[Yi91] W. Yi. CCS + Time = an Interleaving Model for Real-Time Systems, Proc.18th Int. Coll. on Automata, Languages and Programming (ICALP), Madrid, July 1991. LNCS 510, Springer New York 1991, pp. 217-228.