Modal Abstractions in μCRL*

Jaco van de Pol and Miguel Valero Espada

Centrum voor Wiskunde en Informatica,
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
{Jaco.van.de.Pol,Miguel.Valero.Espada}@cwi.nl

Abstract. We describe a framework to generate modal abstract approximations from process algebraic specifications, written in the language μCRL. We allow abstraction of state variables and action labels. Moreover, we introduce a new format for process specifications called *Modal Linear Process Equation* (MLPE). Every transition step may lead to a set of abstract tates labelled with a set of abstract actions. We use MLPEs to characterize abstract interpretations of systems and to generate *Modal Labelled Transition Systems*, in which transitions may have two modalities *may* and *must*. We prove that the abstractions are sound for the full action-based μ-calculus. Finally, we apply the result to check some safety and liveness properties for the bounded retransmission protocol.

1 Introduction

The theory of abstract interpretation [4, 13] denotes a classical framework for program analysis. It extracts program approximations by eliminating uninteresting information. Computations over concrete universes of data are performed over smaller abstract domains. The application of abstract interpretation to the verification of systems is suitable since it allows to formally transform possibly infinite instances of specifications into smaller and finite ones. By loosing some information we can compute a desirable view of the analysed system that preserves some interesting properties of the original.

The achievement of this paper is to enhance existing process algebraic verification tools (e.g. LOTOS, μCRL) with state-of-the-art abstract interpretation techniques that exist for state-based reactive systems. These techniques are based on homomorphisms [3] (easier to use) or Galois Connections [16, 5, 12, 9] (more precise abstractions). The latter are sound for safety as well as liveness properties. A three-valued logic ensures that the theory can be used for proofs and refutations of temporal properties. We transpose this to a process algebra setting, allowing abstraction of states and action labels, and treating homomorphisms and Galois Connections in a uniform way. A preliminary step was already taken in [7]; those authors show how process algebras can benefit from abstract interpretation *in principle*. To this end they work with a basic LOTOS language and

* Partially supported by PROGRESS, the embedded systems research program of the Dutch organisation for Scientific Research NWO, the Dutch Ministry of Economic Affairs and the Technology Foundation STW, grant CES.5009.

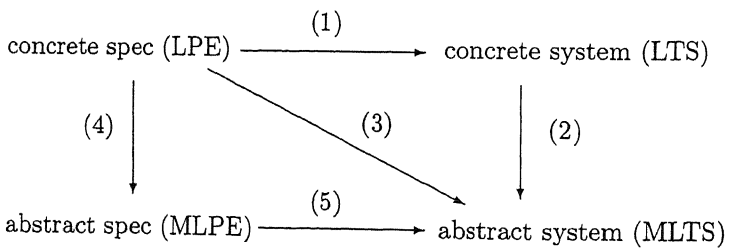
C. Rattray et al. (Eds.): AMAST 2004, LNCS 3116, pp. 409–425, 2004.

a simple temporal logic; their abstractions preserve linear-time safety properties only.

Semantically, our method is based on *Modal Labelled Transition Systems* [15]. MLTSs are *mixed* transition systems in which transitions are labelled with actions and with two modalities: *may* and *must*. They are appropriate structures to define abstraction/refinement relations between processes. *May* transitions determine the actions that possibly occur in all refinements of the system while *must* transitions denote the ones that necessarily happen. On the one hand, the *may* part corresponds to an over-approximation that preserves *safety* properties of the concrete instance and on the other hand the *must* part under-approximates the model and reflects *liveness* properties. We define approximations and prove that they are sound for all properties in the full (action-based) μ-calculus [14], including negation. We had to extend the existing theory by allowing abstraction and information ordering of action labels, which is already visible in the semantics of μ-calculus formulas. This is treated in Section 2.

This theory is applied to μCRL specifications [10], which (as in LOTOS) consist of an ADT part defining data operations, and a process specification part, specifying an event-based reactive system. Processes are defined using sequential and parallel composition, non-deterministic choice and hiding. Furthermore, atomic actions, conditions and recursion are present, and may depend on data parameters. The μCRL toolset [1, 2] transforms specifications to *linear process equations* (LPE). An LPE is a concise representation of all possible interleavings of a system in which parallel composition and hiding are eliminated. Several tools that manipulate LPEs have been developed; they do, for example, symbolic model checking, state space reduction, confluence analysis, etc... The μCRL language and tool set have been used in numerous verifications of communication and security protocols and standards, distributed algorithms and industrial embedded systems.

We implement abstract interpretation as a transformation of LPEs to MLPEs (modal LPEs). MLPEs capture the extra non-determinism arising from abstract interpretation. They allow a simple transition to lead to a *set* of states with a *set* of action labels. We show that the MLTS generated from an MLPE is a proper abstraction of the LTS generated from the original LPE. This implies soundness for μ-calculus properties. Section 4 is devoted to this part. The next figure shows the different possibilities to extract abstract approximations from a concrete specifications.



From a concrete system, encoded as an LPE, we can:

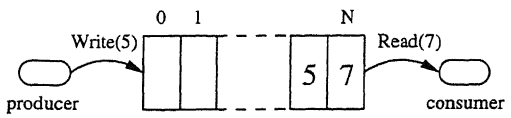
- Generate the concrete transition system (1), from which we compute the abstraction (2). This solution is not very useful for verification because the generation of the concrete transition system may be impossible due to the size of the state space.
- Generate directly the abstract *Modal*-LTS (3), by interpreting the concrete specification over the abstract domain. This solution avoids the generation of the concrete transition system.
- First, generate a symbolic abstraction of the concrete system (4), and then extract the abstract transition system (5).

Typically, standard abstract interpretation frameworks implement the second approach, however we believe that the third one is more modular. MLPEs act as intermediate representation that may be subjected to new transformations. Furthermore, MLPEs can be encoded as LPEs thus our method integrates perfectly with the existing transformation and state space generation tools of the μCRL toolset. Also, the three valued model checking problem can be rephrased as the usual model checking problem, along the lines of [9]. This enables the reuse of the model checkers in the CADP toolset [8]. The latter two transformations are not detailed in the current paper.

The main part of the theory mentioned above has been defined and proved correct in an elegant way using the computer assisted theorem prover PVS [18]. The use of the theorem prover gives extra confidence about the correctness of the theory. Furthermore, the definitions and proofs can be reused to easily extend the theory, to prove other transformations, or to apply the same techniques to another specification language. Also, this prover could be used to prove the safety conditions generated for user-specified abstractions.

To improve usability, we have predefined a few standard abstractions. Of course, the user can define specific abstractions, which in general lead to the generation of safety conditions. Finally, thanks to the uniform treatment, the tool can automatically lift a (predefined or user specified) homomorphism to a Galois Connection, thus combining ease with precision.

As running example we use a simple system in which two processes communicate by sending natural numbers through a channel described as a FIFO buffer of size N, see the figure below. The system has two sources of infinity: the size of the buffer and the value of the data, which should be abstracted if we want to apply model checking techniques to its verification. Finally, we demonstrate the method by checking some safety and liveness properties of the bounded retransmission protocol in Section 5.



2 Transition Systems

Part of the results included in this section are well known in the field of abstract interpretation. We adapt classical frameworks for generating safe abstract approximations, by doing a non-trivial extension of them in order to allow the explicit abstraction of action labels which will permit to manipulate infinitely branching systems. First, we start by defining some general concepts and then we continue by introducing the two abstraction techniques.

The semantics of a system can be defined by a *Labelled Transition System*. We assume a non-empty set S of states, together with a non-empty set of transition labels A:

Definition 1 *A transition is a triple $s \xrightarrow{a} s'$ with $a \in A$ and $s,s' \in S$. We define a* Labelled Transition System *(LTS) as tuple $(S, A, \rightarrow, s_0)$ in which S and A are defined as above and $\rightarrow$ is a possibly infinite set of transitions and s_0 in S is the initial state.*

To model abstractions we are going to use a different structure that allows to represent approximations of the concrete system in a more suitable way. As introduced before, in *Modal Labelled Transition Systems* transitions have two modalities *may* and *must* which denote the possible and necessary steps in the refinements. This concept was introduced by Larsen and Thomsen [15]. Let us see the definition:

Definition 2 *A* Modal Labelled Transition System *(MLTS) or* may/must *labelled transition system (*may/must*-LTS) is a tuple $(S, A, \rightarrow_\Diamond, \rightarrow_\Box, s_0)$ where S, A and s_0 are as in the previous definition and $\rightarrow_\Diamond, \rightarrow_\Box$ are possibly infinite sets of (may or must) transitions of the form $s \xrightarrow{a}_x s'$ with $s,s' \in S$, $a \in A$ and $x \in \{\Diamond, \Box\}$. We require that every* must*-transition is a* may*-transition ($\xrightarrow{a}_\Box \subseteq \xrightarrow{a}_\Diamond$).*

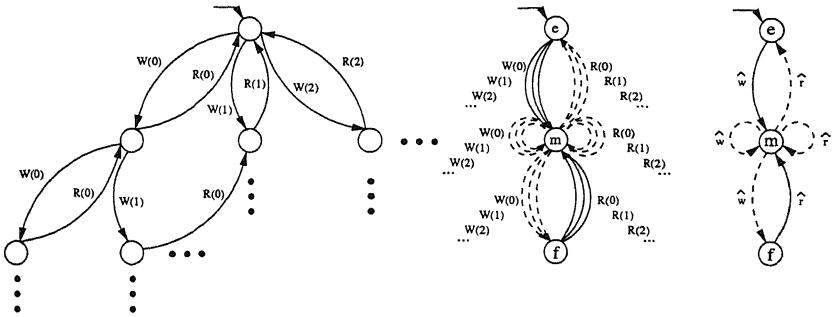
MLTSs are suitable structures for stepwise refinements and abstractions. A refinement step of a system is done by preserving or extending the existing *must*-transitions and by preserving or removing the *may*-transitions. Abstraction is done the other way around. To every LTS corresponds a trivially equivalent MLTS constructed by labelling all transitions with both modalities; we will call it the corresponding *concrete* MLTS.

Having a set of states S and a set of action labels A with their corresponding abstract sets, denoted by $\widehat{S}$ and $\widehat{A}$, we define a homomorphism H as a pair of total and surjective functions $\langle h_S, h_A \rangle$, where h_S is a mapping from states to abstract states, i.e., $h_S : S \rightarrow \widehat{S}$, and h_A maps action labels to abstract action labels, i.e., $h_A : A \rightarrow \widehat{A}$. The abstract state $\widehat{s}$ corresponds to all the states s for which $h_S(s) = \widehat{s}$, and the abstract action label $\widehat{a}$ corresponds to all the actions a for which $h_A(a) = \widehat{a}$.

Definition 3 *Given a* concrete *MLTS P $(S, A, \rightarrow_\Diamond, \rightarrow_\Box, s_0)$ and a mapping H, we say that $\widehat{P}$ defined by $(\widehat{S}, \widehat{A}, \rightarrow_\Diamond, \rightarrow_\Box, \widehat{s}_0)$ is the* minimal may/must$_H$-abstraction *(denoted by $\widehat{P} = min_H(P)$) if and only if $h_S(s_0) = \widehat{s}_0$ and the following conditions hold:*

- $\widehat{s} \xrightarrow{\widehat{a}}_{\Diamond} \widehat{r} \iff \exists\, s, r, a.\, h_S(s) = \widehat{s} \wedge h_S(r) = \widehat{r} \wedge h_A(a) = \widehat{a} \wedge s \xrightarrow{a}_{\Diamond} r$
- $\widehat{s} \xrightarrow{\widehat{a}}_{\Box} \widehat{r} \iff \forall\, s.h_S(s) = \widehat{s}.\, (\exists\, r, a.\, h_S(r) = \widehat{r} \wedge h_A(a) = \widehat{a} \wedge s \xrightarrow{a}_{\Box} r)$

This definition gives the most accurate abstraction of a concrete system by using a homomorphism, in other words the one that preserves most information of the original system. The figure below shows, on the left side, the *concrete* MLTS corresponding to the buffer model[1]. In the middle we see the minimal abstraction of the system by only abstracting states with h_S defined as follows: it maps the initial state to the abstract state e, which means *empty*, the states in which there are N entries in the buffer to f, which represents *full*, and the rest of the states to m, which means something in the *middle*; we can see that the resulting abstract system is infinitely branching, therefore in order to be able to do model checking to the system we need also to abstract the action labels. We define h_A as follows: it maps all the write actions to $\widehat{w}$ and all the read actions to $\widehat{r}$. On the right side, we see the result of applying both abstractions. In the final system, we have removed all the information about the values that are in the buffer and the transferred data, only preserving the information about whether the buffer is empty, full or none of them. This abstraction allows to have a small finite model which keeps some information about the original. The example clearly illustrates the importance of the abstraction of action labels.



Instead of using mappings between concrete and abstract domains we can define relations. The other classical approach we present is based on Galois Connections between domains, and it was introduced in the late seventies by Cousot and Cousot [4], see also [5] for a good introduction. Two functions α and γ over two partially ordered sets $(P, \subseteq)$ and $(Q, \preccurlyeq)$ such that $\alpha : P \to Q$ and $\gamma : Q \to P$ form a Galois Connection if and only if the following conditions hold:

1. α and γ are total and monotonic.
2. $\forall p : P, p \subseteq \gamma \circ \alpha(p)$.
3. $\forall q : Q, \alpha \circ \gamma(q) \preccurlyeq q$.

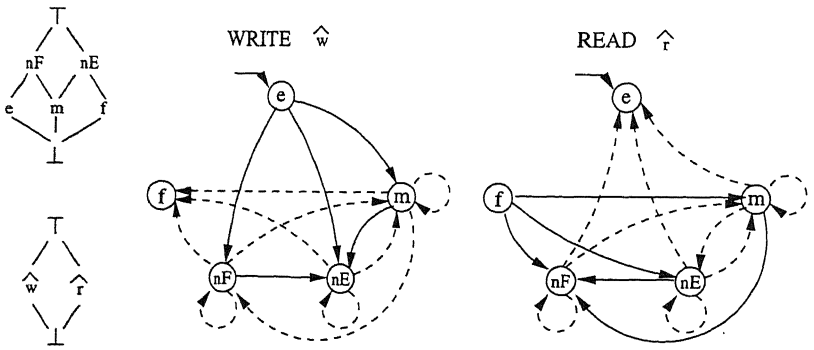
[1] For clarity we use dashed lines to represent *may* transitions and normal lines to represent *must* transitions. Note that when there is a *must* transition we do not include the corresponding *may* one.

α is the lower adjoint and γ is the upper adjoint of the Galois Connection, and they uniquely determine each other. Galois Connections enjoy suitable properties to perform abstractions. Let $\mathcal{P}(S)$ and $\mathcal{P}(A)$ be partially ordered sets ordered by the set inclusion operator and the abstract $\widehat{S}$ and $\widehat{A}$ both being posets equipped with some order $\preccurlyeq$. The order gives a relation of the precision of the information contained in the elements of the domain. We define a pair G of Galois Connections: $G = \langle(\alpha_S, \gamma_S), (\alpha_A, \gamma_A)\rangle$. α is usually called the abstraction function and γ the concretization function. As in the previous case we define the minimal abstraction, as follows:

Definition 4 *Given two systems P and $\widehat{P}$ defined as in definition 3 and a pair of Galois Connections G, $\widehat{P}$ is the* minimal may/must$_G$-abstraction *(denoted by $\widehat{P} = min_G(P)$) if and only if $s_0 \in \gamma_S(\widehat{s}_0)$ and the following conditions hold:*

- $\widehat{s} \xrightarrow{\widehat{a}}_{\Diamond} \widehat{r} \iff \exists\, s \in \gamma_S(\widehat{s}),\, r \in \gamma_S(\widehat{r}),\, a \in \gamma_A(\widehat{a}).\, s \xrightarrow{a}_{\Diamond} r$
- $\widehat{s} \xrightarrow{\widehat{a}}_{\Box} \widehat{r} \iff \forall\, s \in \gamma_S(\widehat{s}).\, (\exists\, r \in \gamma_S(\widehat{r}),\, a \in \gamma_A(\widehat{a}).\, s \xrightarrow{a}_{\Box} r)$

The following figure presents a part of the minimal abstraction of the buffer system[2]. On the left side we can see the two abstract lattices, and on the right side we see the transitions corresponding to the abstract write and read actions. The abstract lattices are: $\{\bot, e, m, f, nE, nF, \top\}$ and $\{\bot, \widehat{w}, \widehat{r}, \top\}$, the Galois Connection is trivially defined following the previous example and considering that nE represents the states in which the buffer is not empty and nF in which it is not full.



Due to the order over the abstract states and actions, the minimal system defined by the above presented definition is saturated of *may* and *must* transitions, i.e. there are transitions that do not add any extra information. We can easily see in the previous figure that the *must* part is saturated for example, the transition $e \xrightarrow{\widehat{w}}_{\Box} nE$ does not add any information because we have $e \xrightarrow{\widehat{w}}_{\Box} m$ which is more precise. We can restrict our definition by requiring that the abstract transitions

[2] For readability, we separate write transitions: $\xrightarrow{\widehat{w}}$ and read transitions $\xrightarrow{\widehat{r}}$ and we do not include transitions to and from $\top$, or labelled with it.

are performed only between the most precise descriptions of the concrete transitions, as done in [5][3]. In the previous figure, this would remove: $e \xrightarrow{\widehat{w}}_{\Diamond,\Box} nF$, $e \xrightarrow{\widehat{w}}_{\Diamond,\Box} nE$, $m \xrightarrow{\widehat{w}}_{\Diamond} nF$, $m \xrightarrow{\widehat{r}}_{\Diamond} nE$, $f \xrightarrow{\widehat{r}}_{\Diamond,\Box} nF$, $f \xrightarrow{\widehat{r}}_{\Diamond,\Box} nE$, $nF \xrightarrow{\widehat{w},\widehat{r}}_{\Diamond} nF$ and $nE \xrightarrow{\widehat{w},\widehat{r}}_{\Diamond} nE$. We call the system in which all redundant transitions have been eliminated *restricted* and it is denoted by $\widehat{P}\downarrow$.

We formalize now the approximation relation between MLTSs:

Definition 5 *Given two MLTSs* P $(S, A, \rightarrow_{\Diamond}, \rightarrow_{\Box}, s_0)$ *and* Q $(S, A, \rightarrow_{\Diamond}, \rightarrow_{\Box}, s_0)$ *built over the same sets of states* $\langle S, \preccurlyeq_S \rangle$ *and actions* $\langle A, \preccurlyeq_A \rangle$*; Q is an abstraction of P, denoted by $P \sqsubseteq_{\preccurlyeq} Q$, if the following conditions hold:*

- $\forall s, a, r, s'.\ s \xrightarrow{a}_{\Diamond} r \wedge s \preccurlyeq_S s' \implies \exists a', r'.\ s' \xrightarrow{a'}_{\Diamond} r' \wedge r \preccurlyeq_S r' \wedge a \preccurlyeq_A a'$
- $\forall s', a', r', s.\ s' \xrightarrow{a'}_{\Box} r' \wedge s \preccurlyeq_S s' \implies \exists a, r.\ s \xrightarrow{a}_{\Box} r \wedge r \preccurlyeq_S r' \wedge a \preccurlyeq_A a'$

$P \sqsubseteq_{\preccurlyeq} Q$ means that Q is more abstract than P and it preserves all the information of the *may*-transitions of P and at least all *must* transitions present in Q are reflected in P. The *may* part of Q is an over-approximation of P and the *must* part is an under-approximation. The refinement relation is the dual of the abstraction. Note that for the homomorphism approach there is no order defined between states or actions so we substitute $\preccurlyeq$ by $=$.

3 Logical Characterization

To express properties about systems we are going to adapt the highly expressive temporal logic (action-based) μ-calculus [14] which is defined by the following syntax, where a is an action in A:

$$\varphi ::= T \mid F \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid [a]\varphi \mid \langle a \rangle \varphi \mid Y \mid \mu Y.\varphi \mid \nu Y.\varphi$$

Formulas are assumed to be monotonic. Following [12], a given formula is interpreted dually over an MLTS, i.e. there will be two sets of states that satisfy it. A set of states that necessarily satisfy the formula and a set of states that possibly satisfy it. Thus, the semantics of the formulas is given by $[\![\varphi]\!] \in 2^S \times 2^S$ and the projections $[\![\varphi]\!]^{nec}$ and $[\![\varphi]\!]^{pos}$ give the first and the second component, respectively. We show below the simultaneous recursive definitions of the evaluation of a state formula. In the state formulas, the propositional context $\rho : Y \rightarrow 2^S \times 2^S$ assigns state sets to propositional variables, and the $\oslash$ operator denotes context overriding. Note that the precision order between action labels plays an important role in the definition of the semantics of the modalities.

[3] The main difference with Dams' approach is that we preserve the condition $\xrightarrow{a}_{\Box} \subseteq \xrightarrow{a}_{\Diamond}$.

$$
\begin{aligned}
[\![F]\!]_\rho &= \langle \emptyset, \emptyset \rangle \\
[\![T]\!]_\rho &= \langle S, S \rangle \\
[\![\neg\varphi]\!]_\rho &= \langle S \backslash [\![\varphi]\!]_\rho^{pos}, S \backslash [\![\varphi]\!]_\rho^{nec} \rangle \quad \text{(Note the switch of } pos \text{ and } nec\text{)} \\
[\![\varphi_1 \wedge \varphi_2]\!]_\rho &= \langle [\![\varphi_1]\!]_\rho^{nec} \cap [\![\varphi_2]\!]_\rho^{nec}, [\![\varphi_1]\!]_\rho^{pos} \cap [\![\varphi_2]\!]_\rho^{pos} \rangle \\
[\![\varphi_1 \vee \varphi_2]\!]_\rho &= \langle [\![\varphi_1]\!]_\rho^{nec} \cup [\![\varphi_2]\!]_\rho^{nec}, [\![\varphi_1]\!]_\rho^{pos} \cup [\![\varphi_2]\!]_\rho^{pos} \rangle \\
[\![[a]\varphi]\!]_\rho &= \langle \{s \mid \forall r, a'.\, a \preccurlyeq a' \wedge s \xrightarrow{a'}_{\Diamond} r \Rightarrow r \in [\![\varphi]\!]_\rho^{nec}\}, \\
&\qquad \{s \mid \forall r, a'.\, a' \preccurlyeq a \wedge s \xrightarrow{a'}_{\Box} r \Rightarrow r \in [\![\varphi]\!]_\rho^{pos}\} \rangle \\
[\![\langle a \rangle\varphi]\!]_\rho &= \langle \{s \mid \exists r, a'.\, a' \preccurlyeq a \wedge s \xrightarrow{a'}_{\Box} r \wedge r \in [\![\varphi]\!]_\rho^{nec}\}, \\
&\qquad \{s \mid \exists r, a'.\, a \preccurlyeq a' \wedge s \xrightarrow{a'}_{\Diamond} r \wedge r \in [\![\varphi]\!]_\rho^{pos}\} \rangle \\
[\![Y]\!]_\rho &= \rho(Y) \\
[\![\mu Y.\varphi]\!]_\rho &= \langle \bigcap \{S' \subseteq S \mid \Phi_\rho^{nec}(S') \subseteq S'\}, \bigcap \{S' \subseteq S \mid \Phi_\rho^{pos}(S') \subseteq S'\} \rangle \\
[\![\nu Y.\varphi]\!]_\rho &= \langle \bigcup \{S' \subseteq S \mid S' \subseteq \Phi_\rho^{nec}(S')\}, \bigcup \{S' \subseteq S \mid S' \subseteq \Phi_\rho^{pos}(S')\} \rangle \\
&\qquad \text{where } \Phi_\rho^x : 2^S \to 2^S \text{ with } x \in \{nec, pos\},\ \Phi_\rho^x(S') = [\![\varphi]\!]^x_{(\rho \oslash [S'/Y])}
\end{aligned}
$$

We say that a state s necessarily satisfies a formula φ, denoted by $s \models^{nec} \varphi$, iff $s \in [\![\varphi]\!]^{nec}$ and dually s possibly satisfies a formula φ, denoted by $s \models^{pos} \varphi$, iff $s \in [\![\varphi]\!]^{pos}$. We remark that from the semantics of the negation follows:

- s necessarily satisfies $\neg\varphi$ iff s not possibly satisfies φ.
- s possibly satisfies $\neg\varphi$ iff s not necessarily satisfies φ.

It is not difficult to see that if s necessarily satisfies a formula φ then also s possibly satisfies φ. This follows from the fact that every *must*-transition is also a *may*-transition. Without this condition we would be able to prove $s \models^{nec} \varphi$ and $s \models^{nec} \neg\varphi$ for some φ which will lead to an *inconsistent* logic. In fact, it cannot be proved for any φ $s \models^{nec} \varphi$ and also $s \models^{nec} \neg\varphi$, i.e. the necessarily interpretation is consistent and it is always possible to prove $s \models^{pos} \varphi$ or $s \models^{pos} \neg\varphi$ which means that the possibly interpretation is complete. The semantics gives a three valued logic:

- s necessarily satisfies φ.
- s possibly satisfies φ but not necessarily satisfies φ.
- s not possibly satisfies φ.

Since the abstraction of a system preserves some information of the original one, the idea is to prove properties on the abstract and then to infer the result for the original. Since action labels occur in μ-calculus formulas, formulas over A are distinct from formulas over $\widehat{A}$. Therefore, we need to define the meaning of the satisfaction relation of an abstract formula on a concrete state: $[\![\widehat{\varphi}]\!]_\xi$ where ξ is either h_A or α_A depending on whether we use a homomorphism or a Galois Connection. $[\![\widehat{\varphi}]\!]_\xi$ gives the set of concrete states that (*necessarily* or *possibly*) satisfy an abstract formula. An extract of the *necessarily* semantics is given below, note that for the rest of the cases (T, F, $\wedge$, $\vee$ and fixpoints) the definition does not change, and the *possibly* semantics are dual:

$$[\![[\widehat{a}]\widehat{\varphi}]\!]^{nec}_{\xi} = \{s \mid \forall r, a.\, \widehat{a} \preccurlyeq \xi(\{a\}) \wedge s \xrightarrow{a}_{\Diamond} r \Rightarrow r \in [\![\widehat{\varphi}]\!]^{nec}_{\xi}\}$$
$$[\![\langle\widehat{a}\rangle\widehat{\varphi}]\!]^{nec}_{\xi} = \{s \mid \exists r, a.\, \xi(\{a\}) \preccurlyeq \widehat{a} \wedge s \xrightarrow{a}_{\Box} r \wedge r \in [\![\widehat{\varphi}]\!]^{nec}_{\xi}\}$$

A concrete state s necessarily satisfies the abstract formula $\widehat{\varphi}$, denoted by $s \models^{nec}_{\xi} \widehat{\varphi}$, iff $s \in [\![\widehat{\varphi}]\!]^{nec}_{\xi}$. And dually, $s \models^{pos}_{\xi} \widehat{\varphi}$, iff $s \in [\![\widehat{\varphi}]\!]^{pos}_{\xi}$. Now we can give the property preservation result:

Theorem 6 *Let P be the MLTS $(S, A, \rightarrow_{\Diamond}, \rightarrow_{\Box}, s_0)$, X be either a homomorphism $H = \langle h_S, h_A\rangle$ between (S, A) and $(\widehat{S}, \widehat{A})$ [in which case ξ stands for h] or a Galois Connection $G = \langle(\alpha_S, \gamma_S), (\alpha_A, \gamma_A)\rangle$ between $(\mathcal{P}(S), \mathcal{P}(A))$ and $(\widehat{S}, \widehat{A})$ [in which case ξ stands for α] and let $\widehat{P}\downarrow$ (over $\widehat{S}$ and $\widehat{A}$) be the minimal (restricted) abstraction of P. And finally let $\widehat{\varphi}$ be a formula over $\widehat{A}$, then for all $p \in S$ and $\widehat{p} \in \widehat{S}$ such that $\xi(\{p\}) \preccurlyeq \widehat{p}$:*

- $\widehat{p} \models^{nec} \widehat{\varphi} \Rightarrow p \models^{nec}_{\xi} \widehat{\varphi}$
- $\widehat{p} \not\models^{pos} \widehat{\varphi} \Rightarrow p \not\models^{pos}_{\xi} \widehat{\varphi}$

The proof follows from the fact that every *may* trace of P is mimicked on $\widehat{P}$ by some related states and, on the other hand, every *must* trace of $\widehat{P}$ is present in P. We refer to the technical report [19] for the proof.

The theorem states that we can infer the satisfaction of a formula on a concrete system if it is *necessarily* satisfied on the abstract. Furthermore, if the formula is not *possibly* satisfied on the abstract it will not hold on the concrete either, so it can be refuted. For example, the two presented abstractions (by homomorphism and by Galois Connection) prove: *"It is possible to write something if the buffer is empty"* expressed as $e \models^{nec} \langle\widehat{w}\rangle T$, which means that in the concrete system s_0 either satisfies $\langle w(0)\rangle T$ or $\langle w(1)\rangle T$ or $\langle w(2)\rangle T$ or ... Furthermore, we can prove on the abstract $f \not\models^{pos} \langle\widehat{w}\rangle T$ which means that in the concrete, all the corresponding concrete states satisfy neither $\langle w(0)\rangle T$ nor $\langle w(1)\rangle T$ nor $\langle w(2)\rangle T$ nor ...

In general, abstractions produced by using Galois Connections preserve more information than the ones generated by homomorphisms, however the state space reduction is stronger in the latter case. For example, the abstraction with the Galois Connection can prove *absence of deadlock* since there is a *must* transition from every state that *may* be reached, however the abstraction done by the homomorphism is not able to prove the property for there is no *must* transition from the state *middle* so we cannot infer the existence of some transition from the related states in the concrete system.

Abstract approximations also preserve properties, this idea is stated in the following lemma:

Lemma 7 *Given two MLTSs P and Q, over the same sets of states and labels S and A, with $P \sqsubseteq_{\preccurlyeq} Q$ then for all p and q in S such that $p \preccurlyeq q$ and for all formula φ then:*

- $q \models^{nec}_{Q} \varphi \Rightarrow p \models^{nec}_{P} \varphi$
- $q \not\models^{pos}_{Q} \varphi \Rightarrow p \not\models^{pos}_{P} \varphi$

This result is useful because, by performing symbolic abstractions (see next section) we generate approximations of the minimal abstraction, so the lemma states that we can still infer the satisfaction/refutation of the properties from the approximation to the original.

4 Process Abstraction

μCRL is a combination of process algebra and abstract data types. Data is represented by an *algebraic specification* $\Omega = (\Sigma, E)$, in which Σ denotes a many-sorted *signature* (S, F), where S is the set of sorts and F the set of functions, and E a set of *Σ-equations*, see [10]. All specifications must include the boolean data type with the constants true and false (`T` and `F`). From process algebra μCRL inherits the following operators: `p.q` (perform p and then perform q); `p + q` (perform arbitrarily either p or q); $\sum_{d:D}$ `p(d)` (perform $p(d)$ with an arbitrarily chosen d of sort D); `p` $\lhd$ `b` $\rhd$ `q` (if b is true, perform p, otherwise perform q); `p || q` (run processes p and q in parallel). δ stands for deadlock. Atomic actions may have data parameters. Every μCRL system can be transformed to a special format, called *Linear Process Equation* or *Operator*. An LPE (see definition below) is a single μCRL process which represents the complete system and from which communication and parallel composition operators have been eliminated.

$$X(d:D) = \sum_{i \in I} \sum_{e_i : E_i} a_i(f_i[d, e_i]).X(g_i[d, e_i]) \lhd c_i[d, e_i] \rhd \delta \tag{1}$$

In the definition, d denotes a vector of parameters d of type D that represents the state of the system at every moment. We use the keyword *init* to declare the initial vector of values of d. The process is composed by a finite number I of summands, every summand i has a list of local variables e_i, of possibly infinite domains, and it is of the following form: a condition $c_i[d, e_i]$, if the evaluation of the condition is true the process executes the action a_i with the parameter $f_i[d, e_i]$ and will move to a new state $g_i[d, e_i]$, which is a vector of terms of type D. $f_i[d, e_i], g_i[d, e_i]$ and $c_i[d, e_i]$ are terms built recursively over variables $x \in [d, e_i]$, applications of function over terms $t = f(t')$ and vectors of terms. To every LPE specification corresponds a labelled transition system. The semantics of the system described by an LPE is given by the following rules:

- $s_0 = \text{init}_{lpe}$
- $s \xrightarrow{a} s'$ iff there exist $i \in I$ and $e : E_i$ such that $c_i[s, e] = \mathtt{T}$, $a_i(f_i[s, e]) = a$ and $g_i[s, e] = s'$

Terms are interpreted over the universe of values $\mathcal{D}$. The following μCRL specification corresponds to the LPE of the example, in which the buffer is modeled by a list:

$$X(l : List) = \sum_{d:D} W(d).X(cons(d,l)) \lhd length(l) < N \rhd \delta + R(head(l)).X(tail(l)) \lhd 0 < length(l) \rhd \delta$$

4.1 Abstract Interpretation of μCRL: Data Part

In order to abstract μCRL specifications we may define the relations between concrete and abstract values by means of a mapping $H : \mathcal{D} \to \widehat{\mathcal{D}}$ or a Galois Connection $(\alpha : \mathcal{P}(\mathcal{D}) \to \widehat{\mathcal{D}}, \gamma : \widehat{\mathcal{D}} \to \mathcal{P}(\mathcal{D}))$.

To abstract data terms, first a mapping $\widehat{\ }$ on (lists of) data sorts must be provided. $\widehat{D}$ represents the sort to which D is abstracted. We require that $\widehat{Bool} = Bool$. Next, for each function symbol f, its abstract counterpart $\widehat{f}$ must be provided. In particular, if $f : D \to E$ then $\widehat{f} : \mathcal{P}(\widehat{D}) \to \mathcal{P}(\widehat{E})$. These ingredients allow to extend $\widehat{\ }$ to data terms, by replacing all symbols by their abstract counterpart. In particular, a data term $t : D$ with variables $x : E$ is abstracted to a data term $\widehat{t} : \mathcal{P}(\widehat{D})$ with variables $\widehat{x} : \mathcal{P}(\widehat{E})$.

We now explain why we lifted all symbols to sets of abstract sorts. For example, in our buffer specification we may have a function S which computes the successor of a natural number. The abstract version of S may be defined as follows:

- For the homomorphism case: $\widehat{S}(empty) = middle$, $\widehat{S}(middle) = middle$ or $\widehat{S}(middle) = full$. It will be undefined for *full*.
- For the Galois Connection approach: $\widehat{S}(empty) = middle$, $\widehat{S}(middle) = nonEmpty$, $\widehat{S}(nonFull) = nonEmpty$, $\widehat{S}(nonEmpty) = nonEmpty$, $\widehat{S}(full) = \top$, $\widehat{S}(\bot) = \bot$ and $\widehat{S}(\top) = \top$.

Abstract interpretation of functions may add non-determinism to the system, for example $\widehat{S}(middle)$ in the homomorphism case may return different values (*middle* and *full*). Furthermore, not all the sorts of a specification need to be abstracted, for example, a predicate $\widehat{empty?}$ applied to *empty* will return true, however, applied to *nonFull* it can be either true or false. To deal with these two considerations, we have lifted abstract functions to return sets of values. So for the homomorphism case, $\widehat{S}(\{middle\}) = \{middle, full\}$. For the Galois case, $\widehat{S}(\{empty\}) = \{nonEmpty\}$ and $\widehat{empty?}(nonFull) = \{T, F\}$.

Not all possible abstract interpretations are correct; in order to generate *safe* abstractions the data terms involved in the specification and their abstract versions have to satisfy a formal requirement, usually called *safety condition*. The condition for the homomorphism function and Galois connections are expressed as follows (note that in the second case t is applied pointwisely to sets).

- Homomorphism: for all d in D. $H(t[d]) \in \widehat{t}[\{H(d)\}]$
- Galois connection: for all $\widehat{d}$ in $\widehat{D}$. $\widehat{t}[\widehat{d}] \succcurlyeq \alpha(t[\gamma(\widehat{d})])$

4.2 Abstract Interpretation of μCRL: Process Part

We will now present a new format, the *Modal Linear Process Equation* (MLPE), and the abstraction mapping from LPEs to MLPEs. An MLPE has the following form:

$$X(d:\mathcal{P}(D)) = \sum_{i\in I}\sum_{e_i:E_i} a_i(F_i[d,e_i]).X(G_i[d,e_i]) \lhd C_i[d,e_i] \rhd \delta \qquad (2)$$

The definition is similar to the one of *Linear Process Equation*, the difference is that the state is represented by a list of powersets of abstract values and for every i: C_i returns a non-empty set of booleans, G_i a non-empty set of states and F_i also a non-empty set of action parameters. Actions are parameterized with sets of values, as well. From an MLPE we can generate a *Modal Labelled Transition System* following these semantic rules:

- $S_0 = \text{init}_{mlpe}$
- $S \xrightarrow{A}_{\Box} S'$ iff there exist $i \in I$ and $e \in E_i$ such that $\mathrm{F} \notin C_i[S,e]$, $A = a(F_i[S,e])$ and $S' = G_i[S,e]$
- $S \xrightarrow{A}_{\Diamond} S'$ iff there exist $i \in I$ and $e \in E_i$ such that $\mathrm{T} \in C_i[S,e]$, $A = a(F_i[S,e])$ and $S' = G_i[S,e]$

To compute an abstract interpretation of a linear process, we define the operator "$\bar{}$": $LPE \rightarrow MLPE$ that pushes the abstraction through the process operators till the data part:

$p = X(t)$ then $\bar{p} = X(\widehat{t})$ where X is a process name	$p = p_0.p_1$ then $\bar{p} = \bar{p_0}.\bar{p_1}$
$p = a(t)$ then $\bar{p} = \bar{a}(\widehat{t})$ where a is an action label	$p = \delta$ then $\bar{p} = \delta$
$p = p_0 + ... + p_n$ then $\bar{p} = \bar{p_0} + ... + \bar{p_n}$	$p = \sum_{e:E} p$ then $\bar{p} = \sum_{\widehat{e}:\widehat{E}} \bar{p}$
$p = p_l \lhd t_c \rhd p_r$ then $\bar{p} = \bar{p_l} \lhd \widehat{t_c} \rhd \bar{p_r}$	

MLPEs allow to capture in a uniform way both approaches: Galois Connection and Homomorphism as well as the combination of both consisting in the lifting of a mapping to a Galois Connection. The lifting is done by considering the abstract domain as a lattice over the powerset of the abstract values ordered by subset inclusion and $\alpha(X)$ will be defined as $\{H(x) \mid x \in X\}$ and $\gamma(\widehat{X})$ as $\{x \mid H(x) \in \widehat{X}\}$. In the example, the abstract values $nonEmpty$ and $nonFull$ will be captured by $\{middle, full\}$ and $\{empty, middle\}$ respectively. The successor of $nonFull$ will be the union of the successor of $empty$ and $middle$: $\{middle, full\}$. In this example, the lifting of the homomorphism saves the extra effort of defining abstract functions. In case we use a plain homomorphism (without lifting it to a Galois Connection), we restrict the semantics of the MLPE by letting S_0, S, A and S' be only singleton sets. The MLPE below models the buffer example:

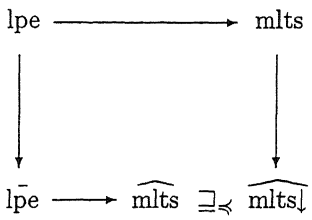
$$X(\widehat{l}:\mathcal{P}(\widehat{List})) = \sum_{\widehat{d}:\widehat{D}} \widehat{w}(\widehat{d}).X(\widehat{cons}(\widehat{d},\widehat{l})) \lhd \widehat{length}(\widehat{l}) < \widehat{N} \rhd \delta + \widehat{r}(\widehat{head}(\widehat{l})).X(\widehat{tail}(\widehat{l})) \lhd \widehat{0} < \widehat{length}(\widehat{l}) \rhd \delta$$

Just considering that all functions are pointwisely extended in order to deal with sets of values, the above MLPE can be used equally for any kind of relation between the data domains: homomorphisms, arbitrary Galois Connections and lifted homomorphisms. The following theorem asserts that the abstract interpretation of a linear process produces an abstract approximation of the (restricted) minimal abstraction of the original.

Theorem 8 *Let* lpe *be Linear Process Equation (as defined in (1)),* mlpe *a Modal Linear Process Equation (as defined in (2)) and* X *an abstraction relation between their data domains (where* X *is either a homomorphism or a Galois Connection). Then if:*

1. mlpe *is the abstract interpretation of* lpe (mlpe = $\bar{\text{lpe}}$),
2. mlts *is the* concrete *MLTS generated from* lpe
3. $\widehat{\text{mlts}\downarrow}$ *is the (restricted) minimal w.r.t X of* mlts
4. *All pairs* (f, F), (g, G) *and* (c, C) *satisfy the safety condition.*

- *Then, the MLTS* ($\widehat{\text{mlts}}$) *generated from* mlpe *is an abstraction of* $\widehat{\text{mlts}\downarrow}$, *i.e,* ($\widehat{\text{mlts}\downarrow} \sqsubseteq_{\preccurlyeq} \widehat{\text{mlts}}$)



The proof is done by checking that every *may* transition generated by the abstract *Modal Linear Process Equation* has at least one more precise counterpart in the (restricted) minimal abstraction of the concrete system (and the other way around for the *must* transitions). By Theorem 6 and Lemma 7, we can prove (refute) properties for *lpe* by considering *mlpe* directly.

5 The Bounded Retransmission Protocol

The BRP is a simplified variant of a Philips' telecommunication protocol that allows to transfer large files across a lossy channel. Files are divided in packets and are transmitted by a sender through the channel. The receiver acknowledges every delivered data packet. Both data and confirmation messages, may be lost. The sender will attempt to retransmit each packet a limited number of times.

The protocol has a number of parameters, such as the length of the lists, the maximum number of retransmissions and the contents of the data, that cause the state space of the system to be infinite and limit the application of automatic verification techniques such as model checking. Following the ideas presented along the paper, we abstract the specification by eliminating some uninteresting information. We base our solution on the μCRL model presented in

the paper [11], in which Groote and van de Pol proved using algebraic methods that the model is branching bisimilar to the desired external behavior also specified in μCRL. This proof requires a strong and creative human interaction in order to be accomplished. However by computing an abstraction of the original specification we can automatically model check some properties.

The system is composed by a sender that gets a file which consists of a list of elements. It delivers the file frame by frame through a channel. The receiver sends an acknowledgment for each frame, when it receives a packet it delivers it to the external receiver client attaching a positive indication I_{fst}, I_{inc} or I_{ok}. The sender, after each frame, waits for the acknowledgments, if the confirmation message does not arrive, it retransmits the packet. If the transmission was successful, i.e. all the acknowledgments have arrived, then the sender informs the sending client with a positive indication. When the maximum number of retransmissions is exceeded, the transmission is canceled and I_{nok} is sent to the exterior by both participants. If the confirmation of the last frame is lost the sender cannot know whether the receiver has received the complete list, therefore it sends *"I don't know"* to the sending client, I_{dk}.

We are interested in proving that the external indications delivered by the sender and the receiver are "consistent". For that purpose, we chose an abstraction that abstracts away the data stored in the file and maps the list to three critical values: *abs_empty, abs_one, abs_more.* The first one denotes the empty list, *abs_one* when it has only one element, and *abs_more* when it has more than one. The maximum number of retransmissions is removed (abstracted to a single abstract value *abs_N*) which makes the sender to non-deterministically choose between resending a lost packet or giving up the transmission.

The next table presents the abstract specification of the data and the abstraction mappings. The concrete specification of the list and the sort Nat are standard. The function *indl* indicates by a bit whether a list is at its end (it is empty or has only one element) or not.

sort	abs_D, abs_Nat, abs_List		$tail(abs\_more) =$
func	$abs\_D :\rightarrow abs\_D$		$\quad \{abs\_more, abs\_one\}$
	$abs\_N :\rightarrow abs\_Nat$		$last(abs\_empty) = \{\mathsf{t}\}$
	$abs\_empty :\rightarrow abs\_List$		$last(abs\_one) = \{\mathsf{t}\}$
	$abs\_one :\rightarrow abs\_List$		$last(abs\_more) = \{\mathsf{f}\}$
	$abs\_more :\rightarrow abs\_List$		$indl(abs\_empty) = \{\mathsf{e}_1\}$
	$succ : abs\_Nat \rightarrow \mathcal{P}(abs\_Nat)$		$indl(abs\_one) = \{\mathsf{e}_1\}$
	$lt : abs\_Nat \times abs\_Nat \rightarrow \mathcal{P}(Bool)$		$indl(abs\_more) = \{\mathsf{e}_0\}$
	$head : abs\_List \rightarrow \mathcal{P}(abs\_D)$	**func**	$H : List \rightarrow abs\_List$
	$tail : abs\_List \rightarrow \mathcal{P}(abs\_List)$		$H : D \rightarrow abs\_D$
	$last : abs\_List \rightarrow \mathcal{P}(Bool)$		$H : Nat \rightarrow abs\_Nat$
	$indl : abs\_List \rightarrow \mathcal{P}(Bit)$	**var**	$l : List$
var	$\widehat{l} :\rightarrow abs\_List$		$d, d' : D$
	$\widehat{m}, \widehat{n} : abs\_Nat$		$m : Nat$
rew	$succ(\widehat{m}) = \{abs\_N\}$	**rew**	$H(empty) = abs\_empty$
	$lt(\widehat{m}, \widehat{n}) = \{\mathsf{t}, \mathsf{f}\}$		$H(add(d, empty)) = abs\_one$
	$head(\widehat{l}) = \{abs\_D\}$		$H(add(d, add(d', l))) = abs\_more$
	$tail(abs\_empty) = \{abs\_empty\}$		$H(d) = abs\_D$
	$tail(abs\_one) = \{abs\_empty\}$		$H(m) = abs\_N$

This abstraction allows to reason about the execution of the final part of the protocol without knowing the exact content of data files or the number of retrials. For example the following *safety* property: *"after a positive notification by the receiver, the sender cannot send a negative one"* is *necessarily* satisfied by the abstract system. We express the property in the Regular Alternation-free μ-calculus, which is a fragment of the modal μ-calculus extended with regular expressions, as follows[4]:

$$[T*.\text{'R}_{may}(.*,\{I_{ok}\})'.(\neg\,\text{'S}.*\,')*.\text{'S}_{may}(\{I_{nok}\})']\,F$$

The following *liveness* property expresses that: *"After a negative notification by the receiver, there exists a path which leads to a negative or* don't know *notification by the sender".*

$$[T*.\text{'R}_{may}(.*,\{I_{nok}\})']\,\langle'.*_{must}'*.(\text{'S}_{must}(\{I_{nok}\})'\vee\text{'S}_{must}(\{I_{dk}\})')\rangle\,T$$

The next property is stronger than the previous, instead of only requesting that there exists a path it states that the expected sender notification is inevitably achieved:

$$[T*.\text{'R}_{may}(.*,\{I_{nok}\})']\;\mu X(\langle'.*_{must}'\rangle\,T\wedge[\neg(\,\text{'S}_{must}(\{I_{nok}\})'\vee\text{'S}_{must}(\{I_{dk}\})')]\,X)$$

These three properties are *necessarily* satisfied in the abstract system, therefore we can infer its satisfaction in the original one. However, the following property which states that *"after a positive notification by the receiver there exists a path which leads to a positive or* don't know *notification by the sender"* is not satisfied in the abstract system. The reason is that we have abstracted away the maximum number of retransmissions, therefore if all the acknowledgments are lost the sender can retransmit the frames forever:

$$[T*.\text{'R}_{may}(.*,\{I_{ok}\})']\,\langle'.*_{must}'*.(\text{'S}_{must}(\{I_{ok}\})'\vee\text{'S}_{must}(\{I_{dk}\})')\rangle\,T$$

Other papers have verified some properties of the protocol using abstract interpretation, we refer among others to [17, ?]. The approach of Manna et al. is based on automatic predicate abstractions and is limited to the proof of invariants. However Dams and Gerth propose a number of creative abstractions in order to prove the satisfaction of some safety properties about the sequentiality on the delivering of the frames.

6 Conclusion

In order to apply the abstract interpretation framework for reactive systems [5, 12] to μCRL processes, we extended it with the explicit abstraction of action labels. This required non-trivial changes in most definitions. A μCRL specification in LPE-form can be abstracted to a modal LPE; the state space of this MLPE corresponds to a reduced MLTS, approximating the original LTS. This

[4] We have used the CADP model checker to prove the properties we describe, therefore we present the formulas with the CADP syntax. The reader not familiar with the logic can safely skip them.

approximation can be used to verify and refute safety as well as liveness formulas for the original system.

The resulting approach incorporates the homomorphism approach (which is easier to understand and use) and the Galois Connection approach (which preserves more properties, especially liveness properties). A user-defined homomorphism can also be lifted to a Galois Connection automatically, combining ease with precision.

We already have a working prototype implementation, which will be described in a separate paper. It is based on a projection of MLPEs to LPEs, in order to reuse existing state space generation tools. We will apply the techniques from [12] in order to translate the three-valued model checking problem to regular model checking, in order to also reuse a model checker for the modal μ-calculus, e.g. CADP [8]. Another interesting question, optimality of abstractions, can in principle be addressed along the lines of [5].

Our theory allows the collection of standard data abstractions with accompanying safety criteria proofs. Such abstraction libraries can be freely used in various protocols, without additional proof obligations. This will enhance automatic verification of protocols in specialized domains.

Model checking becomes more and more crucial for the correctness of software, but in practice additional techniques, such as abstraction, are needed. This may affect the correctness and modularity of the resulting verification methodology and tools. We support modularity by implementing abstraction as an LPE to LPE transformation, which can be composed freely by other existing transformations [1]. We feel that it is important to provide a rigorous basis for verification technology, so we have checked the main part of the results in this paper in the theorem prover PVS.

References

1. S.C.C. Blom, W. Fokkink, J.F. Groote, I.A. van Langevelde, B. Lisser, and J.C. van de Pol. μCRL: A toolset for analysing algebraic specifications. In *CAV*, LNCS 2102, pages 250–254, 2001.
2. S.C.C. Blom, J.F. Groote, I.A. van Langevelde, B. Lisser, and J.C. van de Pol. New developments around the μCRL tool set. In *ENTCS 80*, 2003.
3. E. M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. In *TOPLAS*, ACM 16, pages 1512–1542, 1994.
4. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction of approximation of fixed points. In *POPL*, ACM, pages 238–252, 1977.
5. D. Dams. Abstract Interpretation and Partition Refinement for Model Checking. PhD thesis, Eindhoven University of Technology, 1996.
6. D. Dams and R. Gerth. The bounded retransmission protocol revisited. In *ENTCS 9*, 2000.
7. A. Fantechi, S. Gnesi, and D. Latella. Towards automatic temporal logic verification of value passing process algebra using abstract interpretation. In *CONCUR*, LNCS 1119, pages 562–578, 1996.

8. J.-C. Fernandez, H. Garavel, A. Kerbrat, L. Mounier, R. Mateescu, and M. Sighireanu. CADP – a protocol validation and verification toolbox. In *CAV*, LNCS 1102, pages 437–440, 1996.
9. P. Godefroid, M. Huth, and R. Jagadeesan. Abstraction-based model checking using modal transition systems. In *CONCUR*, LNCS 2154, pages 426–440, 2001.
10. J. F. Groote and A. Ponse. The syntax and semantics of μCRL. In *ACP*, Workshops in Computing Series, pages 26–62, 1995.
11. J. F. Groote and J.C. van de Pol. A bounded retransmission protocol for large data packets. In *AMAST*, LNCS 1101, pages 536–550, 1996.
12. M. Huth, R. Jagadeesan, and D. Schmidt. Modal transition systems: a foundation for three-valued program analysis. In *ESOP*, LNCS 2028, pages 155–169, 2001.
13. N. D. Jones and F. Nielson. Abstract interpretation: A semantics-based tool for program analysis. In *Handbook of Logic in Computer Science*, pages 527–636. 1995.
14. D. Kozen. Results on the propositional μ-calculus. In *ICALP*, LNCS 140, pages 348–359, 1982.
15. K. G. Larsen and B. Thomsen. A modal process logic. In *LICS*, pages 203–210. IEEE, 1988.
16. C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, and S. Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design 6*, pages 11–44, 1995.
17. Z. Manna, M. Colon, B. Finkbeiner, H. Sipma, and T. E. Uribe. Abstraction and modular verification of infinite-state reactive systems. In *RTSE*, LNCS 1526, pages 273–292, 1997.
18. S. Owre, S. Rajan, J.M. Rushb, N. Shankar, and M.K. Srivas. PVS: Combining specification, proof checking, and model checking. In *CAV*, LNCS 1102, pages 411–414, 1996.
19. J.C. van de Pol and M. Valero Espada. Modal abstraction in μCRL. Technical Report SEN-R0401, CWI, 2004.