Modelling, Specifying, and Verifying Message Passing Systems

Benedikt Bollig and Martin Leucker
Lehrstuhl für Informatik II
RWTH Aachen, Germany
{bollig, leucker}@informatik.rwth-aachen.de

Abstract

We present a model for **Message Passing Systems** *unifying concepts of message sequence charts (MSCs) and Lamport diagrams. Message passing systems may be defined—similarly to MSCs—without having a concrete communication medium in mind. Our main contribution is that we equip such systems with a tool set of specification and verification procedures. We provide a global linear time temporal logic which may be employed for specifying message passing systems. In an independent step, a communication channel may be specified. Given both specifications, we construct a Büchi automaton accepting those linearisations of MSCs which satisfy the given formula and correspond to a fixed but arbitrary channel.*

1. Introduction

Components of distributed systems usually communicate with each other via message passing. A sender process sends a message over a channel of which the receiver process takes it out. A prominent formalism to model this kind of systems are *message sequence charts* (MSCs) [7]. They are standardised, can be denoted textually as well as graphically, and are often employed in industry. Furthermore, they are quite similar to UML's sequence charts [3].

An MSC defines a set of processes and a set of communication actions between these processes. In a visual presentation of an MSC, processes are drawn as vertical lines. A labelled arrow from one line to a second line corresponds to the communication event of sending the label value from the first process to the second. Since the vertical lines are interpreted as time lines, there is the general rule that arrows must not go "upwards" since this would describe a situation that a message is received before it was sent. Figure 1 (a) gives an example of an MSC.

When one considers the exact behaviour of an MSC, i.e. the sequences of actions which may be observed when the system is executed, one distinguishes between the so-called *visual order semantics* and the *causal order semantics*. The visual order assumes that the events are ordered as shown in the MSC. For example, process Q in Figure 1 (a) reads an a before it can read a b. Within causal order based semantics, read events on the same process line are not ordered unless they are "causality dependent". For example, read event b may occur before reading a. Anyway, our approach is based on the visual order semantics, but (with some modifications) it also works wrt. the causal order semantics.

There has been a lot of effort for analysing, specifying, verifying, and synthesising message passing systems [14, 15, 1, 12, 2, 11, 6]. However, most of the work was carried out for a fixed communication medium. But channels can be designed in a FIFO manner, they can behave as a *stack* or as a *multiset*, and may have a fixed *capacity* or not. They can be *reliable* or *lossy*. From the practical point of view, all types of channels are useful. Thus, it does not seem to be appropriate to provide dedicated analysation methods for every type of channel but rather to provide procedures where the channel is just a parameter.

Meenakshi and Ramanujam [11] provide specification and verification methods for a variant of MSCs and a whole class of channels. They consider channels which are either *bounded* or *implicit*. Our work can be understood as a generalisation of their approach by allowing arbitrary finite deterministic channels. A minor difference is that we provide a *global logic* while they employ a *local* one.

In this paper, we first give a definition of MSCs which is a slightly modified version of the standard definition. It captures some ideas of Lamport diagrams [8, 9] viz, for example, send events may lack a corresponding receive event (natural for lossy channels). We introduce a (global) linear time temporal logic LTML which is interpreted over MSCs and may be employed for specifying sets of MSCs. Lineartime Temporal Logic over sequences has gained a lot of attraction for the specification of concurrent systems [10], and we believe that LTML provides a similar simple method for specifying MSCs.

A channel may be understood as a process that determines which send events and receive events are associated.

0-7695-1107-4/01 $10.00 © 2001 IEEE

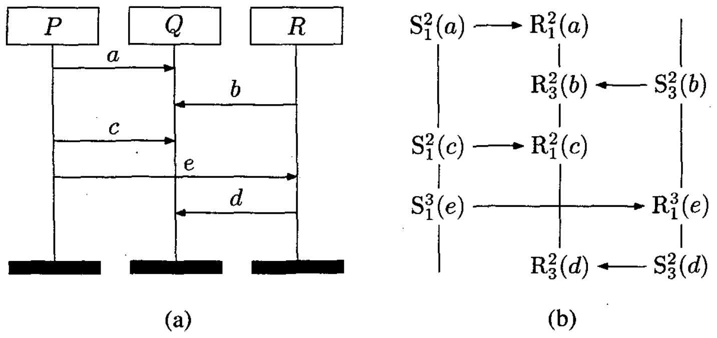

Figure 1. An MSC and its formalisation

We show that a channel may be described as a Mealy automaton, a finite automaton with input and *output*. We give characterising examples that this is an adequate approach for specifying channels.

Now, given a specification of an MSC and a channel, we provide a procedure for constructing an automaton which accepts all linearisations of MSCs that meet the specification and correspond to the channel. This automaton may be checked by standard techniques for emptiness to decide whether the given specification is satisfiable under the specified type of channel. Now, one may change the specification of the channel, take, for example, one with a smaller capacity, and redo the satisfiability test. Another application would be *model checking specifications*. Given a set of MSCs, the question is whether these MSCs meet their specification (under the given specification of the channel). This can easily be done by checking emptiness of the intersection of two languages (cf. Section 5).

The logic we present for aforesaid purposes has the property not to distinguish models from one and the same equivalence class (i.e. which are linearisations of one and the same partial order) such that one has to check a specified property for just one member of each equivalence class. Thus, it is possible to employ so-called partial order reduction methods [13, 18].

Acknowledgement: We would like to thank the reviewers for helpful comments.

2. Message Passing Systems

This section deals with general properties of what we understand by a message passing system. Such a system can be represented by several local components modelled as a family of automata and synchronising their actions by means of a channel. A channel is likewise an automaton but enriched by an output function. This output function unambiguously relates send to receive events and makes sure that executions of the system can finally be considered as MSCs.

2.1. Message Sequence Charts

Let $\mathcal{P}_N := \{1, \ldots, N\}$ be a non-empty set of *processes* and Λ a *message alphabet*. Let further $\Sigma_\mathcal{S} := \{\mathrm{S}_p^q(\lambda) | p, q \in \mathcal{P}_N,\ p \neq q,\ \lambda \in \Lambda\}$ and $\Sigma_\mathcal{R} := \{\mathrm{R}_p^q(\lambda) \mid p, q \in \mathcal{P}_N,\ p \neq q,\ \lambda \in \Lambda\}$ denote the sets of *send* and *receive actions*, respectively, and $\Sigma := \Sigma_\mathcal{S} \cup \Sigma_\mathcal{R}$ the set of *actions*. An action $\mathrm{S}_p^q(\lambda)$ stands for sending a message λ from process p to process q, an action $\mathrm{R}_p^q(\lambda)$ for the corresponding receive action, which is executed by process q.

Definition 1 *A* message sequence chart *(MSC) wrt. $\mathcal{P}_N$ and Λ is a tuple $M = (\{E_p\}_{p\in\mathcal{P}_N}, \{\preceq_p\}_{p\in\mathcal{P}_N}, f, L)$ such that*

- *$\{E_p\}_{p\in\mathcal{P}_N}$ is a family of pairwise disjoint countable nonempty sets of so-called* events *(let $E := \bigcup_{p\in\mathcal{P}_N} E_p$),*
- *for each $p \in \mathcal{P}_N$, $\preceq_p \subseteq E_p \times E_p$ is a well-founded total order (sometimes considered as a relation on $E \times E$),*
- *there is a partition of E into* send events *(S),* receive events *(R), and events without communication partner (G) such that*
- *f is a bijective mapping $S \to R$ satisfying the following (where we let $P : E \cup \Sigma \to \mathcal{P}_N$ yield the process an event or an action belongs to, i.e., $P(e) = p$ iff $e \in E_p$, $P(\mathrm{S}_p^q(\lambda)) = p$, and $P(\mathrm{R}_p^q(\lambda)) = q$):*
  - *There is no sequence $e_1, \ldots, e_n$ of events such that $f(e_n) = e_1$ and for all $i \in \{1, \ldots, n-1\}$, either $f(e_i) = e_{i+1}$ or $e_i \preceq_{P(e_i)} e_{i+1}$.*
  - *$L : E \to \Sigma$ such that $e \in S$ implies both $L(e) = \mathrm{S}_{P(e)}^{P(f(e))}(\lambda)$ and $L(f(e)) = \mathrm{R}_{P(e)}^{P(f(e))}(\lambda)$ for any $\lambda \in \Lambda$, and $e \in G$ implies $P(L(e)) = P(e)$.*

The mapping f associates each send event with a receive event, whereas L provides information about the messages being interchanged by such communicating events.

For example, Figure 1 (b) presents a formal definition of the MSC depicted in Figure 1 (a).

From now on, all premises and definitions are made wrt. a fixed set $\mathcal{P}_N$ of processes and a fixed message alphabet Λ. Σ will denote the corresponding set of actions.

The *visual order* of an MSC M is denoted by $\preceq \subseteq E \times E$ and is defined to be the reflexive and transitive closure of $\bigcup_{p \in \mathcal{P}_N} \preceq_p \cup \{(e, f(e)) \mid e \in S\}$. Obviously, $\preceq$ is a partial order.

A partial execution (configuration) of an MSC can be described by a downwards closed subset of events, the events occurred so far. So let M be an MSC as above. A *configuration* of M is a finite subset E' of E satisfying $E' = \{e \mid \exists e' \in E' : e \preceq e'\}$. Let $Conf(M)$ denote the set of configurations of M. The execution of an MSC may be described by a transition relation over its configurations. $\longrightarrow_M \subseteq Conf(M) \times \Sigma \times Conf(M)$ is defined according to $c \xrightarrow{\sigma}_M c'$ iff $\exists e \in c' - c : L(e) = \sigma$ *and* $c' = c \cup \{e\}$.

In order to relate MSCs to the rich theory of automata over words, the concept of linearisations of an MSC is essential. As usual, for an MSC M with labelling function L, set of events E, and corresponding partial order $\preceq$, let $Lin(M) := \{L(u) \mid u \text{ is a linearisation of } (E, \preceq)\}$ denote the set of *linearisations* of M where L is used as expected. Furthermore, for a set $\mathcal{M}$ of MSCs, we canonically define $Lin(\mathcal{M}) := \bigcup\{Lin(M) \mid M \in \mathcal{M}\}$.

2.2. Channels

A natural approach is to model channels as state machines in which every state represents (an equivalence class of) the channel's configuration. The key idea for signalising matching send and receive events is to employ, besides the transition function, a further output function.

Definition 2 *A* channel C *(wrt. the set of processes and the message alphabet), is a (kind of) Mealy automaton* $(S, \delta, \delta_{\mathbb{N}}, s_{in}, AC)$ *where* S *is its set of* states *containing a state* fail, $\delta : S \times \Sigma \to S$ *its* transition function *with* $\delta(\text{fail}, \sigma) = \text{fail}$ *for all* $\sigma \in \Sigma$, $\delta_{\mathbb{N}} : S \times \Sigma \to \mathbb{N}$ *its* output function, $s_{in} \in S$ *the* initial state, *and finally* $AC = (AC_\rho, AC_{\rho_{\mathbb{N}}})$, $AC_\rho \subseteq S^\infty$ *and* $AC_{\rho_{\mathbb{N}}} \subseteq \mathbb{N}^\infty$, *an* acceptance condition. *Furthermore, whenever* $\delta_{\mathbb{N}}(s_1, \Gamma^{q_1}_{p_1}(\lambda_1)) = \delta_{\mathbb{N}}(s_2, \Omega^{q_2}_{p_2}(\lambda_2))$ *then* $p_1 = p_2$, $q_1 = q_2$, *and* $\lambda_1 = \lambda_2$.

Let $\overline{\delta} : S \times \Sigma^* \to S$ and $\overline{\delta}_{\mathbb{N}} : S \times \Sigma^* \to \mathbb{N}$ extend the above functions in the canonical way, respectively. In particular, $\overline{\delta}_{\mathbb{N}}(s, v\sigma) = \delta_{\mathbb{N}}(\overline{\delta}(s, v), \sigma)$.

The *state run* $\rho(w)$ of C on $w = \sigma_1\sigma_2 \ldots \in \Sigma^\infty$ is $s_{in}\overline{\delta}(s_{in}, \sigma_1)\overline{\delta}(s_{in}, \sigma_1\sigma_2) \ldots \in S^\infty$. The respective *output run* $\rho_{\mathbb{N}}(w)$ is the sequence $\overline{\delta}_{\mathbb{N}}(s_{in}, \sigma_1)\overline{\delta}_{\mathbb{N}}(s_{in}, \sigma_1\sigma_2) \ldots \in \mathbb{N}^\infty$. We say $\rho(w)$ is *accepting* iff both in $\rho(w)$ fail does not occur and $\rho(w) \in AC_\rho$. $\rho_{\mathbb{N}}(w)$ is *accepting* as soon as it is contained in $AC_{\rho_{\mathbb{N}}}$. $\mathcal{L}(C) := \{w \in \Sigma^\infty \mid \rho(w)$ and $\rho_{\mathbb{N}}(w)$ are accepting$\}$ associates with C a word language. Finally, we may call C *finite* iff S is finite.

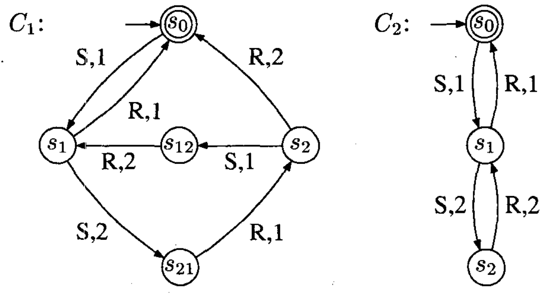

Figure 2. A FIFO channel and a stack

Example 1 Have a look at the finite FIFO channel C_1 given by the left transition graph in Figure 2 and the acceptance condition $AC = (S^\infty, \{x \in \mathbb{N}^\infty \mid$ each number occurs either even or infinitely often$\})$. For the sake of clarity, we omit exactly those transitions leading to fail. Furthermore, let S stand for $\mathrm{S}^2_1(a)$ and R for $\mathrm{R}^2_1(a)$. Hence, our channel only permits process 1 to send and process 2 to receive a message. In order to depict the acceptance condition, we suggest that every message deposited with the channel is taken out sometime. The channel on the right hand side of Figure 2 works in a stack-based manner for the same setup.

To be able to handle channels algorithmically, it is useful to restrict our considerations to channels with finitely many states and a decidable acceptance condition AC. Only to simplify our presentation, in the following, we consider the acceptance condition $(S^\infty, \mathbb{N}^\infty)$ and say it is *satisfied*.

Let in the following v and w range over Σ^∞, σ and σ_i over Σ. Furthermore, we shall let x, y range over $\mathbb{N}^\infty$ and π, π_i range over $\mathbb{N}$.

We bring out the relation between channels and MSCs. For words $w = \sigma_1\sigma_2 \ldots \in \Sigma^\infty$ and $x = \pi_1\pi_2 \ldots \in \mathbb{N}^\infty$ of equal length, we call $\frac{w}{x} \in (\Sigma \times \mathbb{N})^\infty$ *well-formed* iff $\pi_i = \pi_j$ implies $\sigma_i = \Gamma^q_p(\lambda)$ and $\sigma_j = \Omega^q_p(\lambda)$ for any $p, q \in \mathcal{P}_N$, $\lambda \in \Lambda$. The set of well-formed words is denoted by WF, the set of finite well-formed words by fWF. A word $\frac{w}{x} \in$ WF determines an MSC $M^w_x := (\{E_p\}_{p \in \mathcal{P}_N}, \{\preceq_p\}_{p \in \mathcal{P}_N}, f, L)$ where

- $E_p = \{n \in \{1, \ldots, |w|\} \mid P(\sigma_n) = p\}$,
  $S = \{n \mid \sigma_n \in \Sigma_{\mathcal{S}}$ and $\exists m > n : \pi_m = \pi_n$ and $\sigma_m \in \Sigma_{\mathcal{R}}$ and $\forall i = n+1, \ldots, m-1 : \pi_i \neq \pi_n\}$,

$R = \{n \mid \sigma_n \in \Sigma_{\mathcal{R}}$ and $\exists m < n : \pi_m = \pi_n$ and $\sigma_m \in \Sigma_{\mathcal{S}}$ and $\forall i = m+1, \ldots, n-1 : \pi_i \neq \pi_n\}$, $G = \{n \mid n \notin S$ and $n \notin R\}$,

- $m \preceq_p n$ iff $m, n \in E_p$ and $m \leq n$,
- $f(n) = min\{m \mid m > n$ and $\pi_m = \pi_n\}$, and
- $L(n) = \sigma_n$.

Let C be a channel. Since for $w \in \mathcal{L}(C)$ with corresponding output run $\rho_{\mathbb{N}}(w)$, the word $\genfrac{}{}{0pt}{}{w}{\rho_{\mathbb{N}}(w)}$ is obviously well-formed (remember the further demand we made on a channel), we may assign to w and C an MSC $M_C^w := M^w_{\rho_{\mathbb{N}}(w)}$ satisfying $w \in Lin(M_C^w)$. As to a word language, we canonically link a channel C to an MSC language $\mathcal{M}(C) := \bigcup\{M_C^w \mid w \in \mathcal{L}(C)\}$.

An equivalence relation $\sim \subseteq \mathrm{WF} \times \mathrm{WF}$ will turn out to be important. Let first $\approx \subseteq f\mathrm{WF} \times f\mathrm{WF}$ be the least equivalence relation satisfying that if $\beta = \genfrac{}{}{0pt}{}{\sigma_1 \ldots \sigma_i \sigma_{i+1} \ldots \sigma_\ell}{\pi_1 \ldots \pi_i \pi_{i+1} \ldots \pi_\ell}$, $\beta' = \genfrac{}{}{0pt}{}{\sigma_1 \ldots \sigma_{i+1} \sigma_i \ldots \sigma_\ell}{\pi_1 \ldots \pi_{i+1} \pi_i \ldots \pi_\ell}$, $P(\sigma_i) \neq P(\sigma_{i+1})$, and $\pi_i \neq \pi_{i+1}$, then $\beta \approx \beta'$. For infinite well-formed words α, α', let furthermore $\alpha \trianglelefteq \alpha'$ iff for every finite prefix β of α, there is a prefix β' of α' and a β'' such that both $\beta' \approx \beta''$ and β is a prefix of β''. Finally, $\alpha \sim \alpha'$ iff either $\alpha, \alpha' \in f\mathrm{WF}$ and $\alpha \approx \alpha'$ or $\alpha \trianglelefteq \alpha'$ and $\alpha' \trianglelefteq \alpha$.

2.3. Message Passing Automata

We now propose the automata model for distributed systems. It allows to describe a set of MSCs in terms of a finite device and consists of components communicating with each other via channels. Dealing with one possibility of modelling such a system, it is proposed to be a kind of a product automaton [16]. Components synchronise in a local manner, i.e. only execute actions accompanying a corresponding activity on the part of the channel.

Definition 3 *A* message passing automaton *(MPA) is a family $\mathcal{A} = (\{\mathcal{A}_p\}_{p \in \mathcal{P}_N}, C, Q_{in})$ of so-called* local automata, *each of which is of the form $\mathcal{A}_p = (Q_p, \longrightarrow_p, F_p, \widehat{F}_p)$, together with a set Q_{in} of* initial states *and a channel $C = (S, \delta, \delta_{\mathbb{N}}, s_{in}, AC)$. Q_p denotes the nonempty finite set of* local states, *$\longrightarrow_p \subseteq Q_p \times \Sigma_p \times Q_p$ the* local transitions *(Σ_p contains the actions belonging to process p), and $F_p, \widehat{F}_p \subseteq Q_p$ denote sets of* local final states. *The set of initial states is a subset of $Q_{\mathcal{A}} = (\times_{p \in \mathcal{P}_N} Q_p) \times S$, the set of* global states.

Let $\overline{s}[p]$ denote the pth component of a tuple $\overline{s}$, and let $w \upharpoonright p$ yield the word we get by omitting in $w \in \Sigma^\infty$ exactly those actions which do not belong to process p. We define a transition relation $\Longrightarrow_{\mathcal{A}} \subseteq Q_{\mathcal{A}} \times \Sigma \times Q_{\mathcal{A}}$ according to $(\overline{s}, t) \overset{\sigma}{\Longrightarrow}_{\mathcal{A}} (\overline{s}', t')$ iff $\overline{s}[P(\sigma)] \overset{\sigma}{\longrightarrow}_{P(\sigma)} \overline{s}'[P(\sigma)]$, $\overline{s}[p] = \overline{s}'[p]$ for all $p \in \mathcal{P}_N - \{P(\sigma)\}$, and furthermore $\delta(t, \sigma) = t'$.

A *run* of $\mathcal{A}$ on a word $w = \sigma_1\sigma_2 \ldots \in \Sigma^\infty$ is a mapping $\xi : Prf(w) \to Q_{\mathcal{A}}$ from the set of finite prefixes of w to the set of global states such that $\xi(\epsilon) \in Q_{in}$ and $\xi(v) \overset{\sigma}{\Longrightarrow}_{\mathcal{A}} \xi(v\sigma)$ for all $v \in \Sigma^*$ and $\sigma \in \Sigma$ with $v\sigma \in Prf(w)$. ξ is called accepting iff for all $p \in \mathcal{P}_N$, it holds

- $\xi(v)[p] \in F_p$ for a $v \in Prf(w)$ with $v \upharpoonright p = w \upharpoonright p$, if $w \upharpoonright p$ is finite,
- $\xi(v)[p] \in \widehat{F}_p$ for infinitely many $v \in Prf(w)$, if $w \upharpoonright p$ is infinite,

and furthermore both $\rho(w)$ and $\rho_{\mathbb{N}}(w)$, the respective runs of channel C on w, are accepting. $\mathcal{A}$ defines a word language $\mathcal{L}(\mathcal{A}) := \{w \in \Sigma^\infty \mid$ there is an accepting run of $\mathcal{A}$ on $w\}$ and an MSC language $\mathcal{M}(\mathcal{A})$ in exactly the same way as a channel. And obviously, we have both $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(C)$ and $Lin(\mathcal{M}(\mathcal{A})) \subseteq Lin(\mathcal{M}(C))$.

3. A global logic for MSCs

The logic LTML$^-$is given by

$$\varphi ::= \top \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle\sigma\rangle\varphi \mid \langle\tau \in S\rangle\varphi \mid \Diamond\varphi$$

where $\sigma \in \Sigma$ and $\tau \in \Sigma_{\mathcal{S}}$.

LTML$^-$ is LTL$^-$ over Mazurkiewicz traces but enriched by an operator $\langle . \in S\rangle$ specifying that a sending event is eventually answered by a corresponding receiving event. Due to lack of space, we only suggest that we could likewise employ operators $\langle . \in R\rangle$ and $\langle . \in G\rangle$ with the obvious meaning. An LTML$^-$-formula is inductively interpreted over an MSC M wrt. one of its configurations $c \in Conf(M)$—like LTrL for Mazurkiewicz traces [17]. While the Boolean operators are defined as usual, let

- $M, c \models \langle\sigma\rangle\varphi$ iff $\exists c' \in Conf(M) : c \overset{\sigma}{\longrightarrow}_M c'$ and $M, c' \models \varphi$,
- $M, c \models \langle\sigma \in S\rangle\varphi$ iff $\exists c' \in Conf(M)$, $c' = c \uplus \{e\}$: $c \overset{\sigma}{\longrightarrow}_M c'$ and $e \in S$ and $M, c' \models \varphi$,
- $M, c \models \Diamond\varphi$ iff $\exists c' \in Conf(M) : c \subseteq c'$ and $M, c' \models \varphi$.

Instead of $M, \emptyset \models \varphi$, we also write $M \models \varphi$.

In principle, we may also make use of full LTML permitting until-formulas $\varphi\mathcal{U}\psi$ such that

- $M, c \models \varphi\mathcal{U}\psi$ iff $\exists c' \in Conf(M) : c \subseteq c'$ and $M, c' \models \psi$ and for all $c'' \in Conf(M)$ with $c \subseteq c'' \subset c'$, it holds $M, c'' \models \varphi$.

But for readability, we only provide a decision procedure for LTML$^-$. In order to get an impression how to adapt the procedure to full LTML, see [4].

We will freely use abbreviations $\bot = \neg\top$, $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$, $\Box\varphi = \neg\Diamond\neg\varphi$, and so on.

To shortly give a feeling in which way our logic may be employed for specification, let us consider the formula

$$\Box(\langle \mathrm{S}_1^2(\texttt{request})\rangle\top \rightarrow \Diamond\langle \mathrm{S}_2^1(\texttt{reply}) \in S\rangle\top).$$

It requires that every request of client process 1 is finally and successfully answered by server process 2.

Looking towards a decision procedure, it is advisable to additionally interpret LTML$^-$-formulas over words $\genfrac{}{}{0pt}{}{w}{x} \in$ WF interpreting $\top$, negation, and disjunction as usual and furthermore (let $Y \subseteq \mathcal{P}_N$ and $Z \subseteq \mathbb{N}$ be finite sets)

- $\genfrac{}{}{0pt}{}{w}{x} \models_W \langle\sigma\rangle\varphi$ iff $\genfrac{}{}{0pt}{}{w}{x} \models_W \langle\sigma\rangle^{\emptyset}\varphi$,
- $\genfrac{}{}{0pt}{}{w}{x} \models_W \langle\sigma\rangle^{Z}\varphi$ iff $\genfrac{}{}{0pt}{}{w}{x} = \genfrac{}{}{0pt}{}{\sigma_1\ldots\sigma_i\sigma v}{\pi_1\ldots\pi_i\pi y}$ and
  - $\genfrac{}{}{0pt}{}{w}{x} \sim \genfrac{}{}{0pt}{}{\sigma\sigma_1\ldots\sigma_i v}{\pi\pi_1\ldots\pi_i y}$,
  - $\sigma \in \Sigma_{\mathcal{R}}$ implies $\pi \notin Z$, and
  - $\genfrac{}{}{0pt}{}{\sigma_1\ldots\sigma_i v}{\pi_1\ldots\pi_i y} \models_W \varphi$,
- $\genfrac{}{}{0pt}{}{w}{x} \models_W \langle \mathrm{S}_p^q(\lambda) \in S\rangle\varphi$ iff $\genfrac{}{}{0pt}{}{w}{x} = \genfrac{}{}{0pt}{}{\sigma_1\ldots\sigma_i\sigma v}{\pi_1\ldots\pi_i\pi y}$, $\sigma = \mathrm{S}_p^q(\lambda)$, and
  - $\genfrac{}{}{0pt}{}{w}{x} \sim \genfrac{}{}{0pt}{}{\sigma\sigma_1\ldots\sigma_i v}{\pi\pi_1\ldots\pi_i y}$,
  - $\genfrac{}{}{0pt}{}{v}{y} \models_W ev(\mathrm{R}_p^q(\lambda), \pi)$, and
  - $\genfrac{}{}{0pt}{}{\sigma_1\ldots\sigma_i v}{\pi_1\ldots\pi_i y} \models_W \varphi$,
- $\genfrac{}{}{0pt}{}{w}{x} \models_W ev(\sigma, \pi)$ iff $\genfrac{}{}{0pt}{}{w}{x} = \genfrac{}{}{0pt}{}{\sigma_1\ldots\sigma_i\sigma v}{\pi_1\ldots\pi_i\pi y}$ and π is not contained in $\{\pi_1, \ldots, \pi_i\}$,
- $\genfrac{}{}{0pt}{}{w}{x} \models_W \Diamond\varphi$ iff $\genfrac{}{}{0pt}{}{w}{x} \models_W \Diamond_{\emptyset}^{\emptyset}\varphi$,
- $\genfrac{}{}{0pt}{}{w}{x} \models_W \Diamond_Y^Z\varphi$ iff $\exists\, \genfrac{}{}{0pt}{}{w'}{x'} \in$ WF: $\genfrac{}{}{0pt}{}{w}{x} \sim \genfrac{}{}{0pt}{}{w'}{x'}$, $\genfrac{}{}{0pt}{}{w'}{x'} = \genfrac{}{}{0pt}{}{\sigma_1\ldots\sigma_i v}{\pi_1\ldots\pi_i y}$, and
  - $\{P(\sigma_1), \ldots, P(\sigma_i)\} \cap Y = \emptyset$,
  - $\forall k \in \{1, \ldots, i\}$: $\sigma_k \in \Sigma_{\mathcal{R}}$ implies $\pi_k \notin Z$, and
  - $\genfrac{}{}{0pt}{}{v}{y} \models_W \varphi$.

We will illustrate the extended modalities introduced here later on. By the way, note that WF is suffix-closed, i.e., every suffix of a word contained in WF is again in WF. The following proposition is a consequence of well-known results in Mazurkiewicz trace theory.

Proposition 1 *Given $\varphi \in$ LTML$^-$ and a word $\genfrac{}{}{0pt}{}{w}{x} \in$ WF, $\genfrac{}{}{0pt}{}{w}{x} \models_W \varphi$ iff $M_x^w \models \varphi$.*

4. Alternating Büchi Automata

In a nutshell, we will go into the so-called alternating automaton, the instrument for our decision procedure, which extends nondeterministic automata by universal choices. We recall the notion of alternating automata along the lines of [19] where alternating Büchi automata are used for model checking LTL over words.

For a finite set X of variables, let $\mathcal{B}^+(X)$ be the set of *positive Boolean formulas* over X, the smallest set such that $X \subseteq \mathcal{B}^+(X)$ and for $\varphi, \psi \in \mathcal{B}^+(X)$, $\varphi \wedge \psi \in \mathcal{B}^+(X)$ as well as $\varphi \vee \psi \in \mathcal{B}^+(X)$.

A set $Y \subseteq X$ is a *model* of a formula $\varphi \in \mathcal{B}^+(X)$ iff φ evaluates to $true$ when the variables in Y are assigned to $true$ and the members of $X \backslash Y$ to $false$.

An *alternating Büchi automaton* over Σ is a tuple $\mathcal{A} = (Q, \Sigma, \chi, q_{in}, F, \widehat{F})$ where Q is its finite nonempty set of *states*, $q_{in} \in Q$ is the *initial state*, $F, \widehat{F} \subseteq Q$ are the sets of *final states* and $\chi : Q \times \Sigma \rightarrow \mathcal{B}^+(Q)$ is the *transition function.*

A *run* of $\mathcal{A}$ on a word $w = \sigma(0)\sigma(1)\ldots \in \Sigma^\infty$ is a pair (t, T) where t is a complete tree of height $|w|$ and T : $nodes(t) \rightarrow Q$ a labelling function such that the root of t is labelled by q_{in} and for every internal node $v \in nodes(t)$, it holds $\{T(v') \mid v' \in children(v)\} \models \chi(T(v), \sigma(height(v)))$. Let thereby $height(v) = 0$ if v is root, $height(v) = height(parent(v)) + 1$, otherwise. (t, T) is *accepting* iff for every leaf node v, it holds $T(v) \in F$ and every infinite branch of t hits an element of $\widehat{F}$ infinitely often. The language $\mathcal{L}(\mathcal{A})$ is determined by all words for which an accepting run of $\mathcal{A}$ exists.

5. A decision procedure

Given a formula and a finite channel, we now construct an alternating Büchi automaton which accepts those linearisations of MSCs satisfying the formula and corresponding to the channel.

5.1 The automaton

In general, given an LTML$^-$-formula φ, we will not be able to construct a finite automaton recognising $Lin(\{M \mid M \models \varphi\})$, since it is not a regular language. We rather build an alternating Büchi automaton $\mathcal{A}_\varphi^C$ wrt. a finite channel C, such that $\mathcal{L}(\mathcal{A}_\varphi^C) = \{w \mid w \in \mathcal{L}(C)$ and $M_{\rho_{\mathrm{IN}}(w)}^{w} \models \varphi\}$.

We now illustrate the ideas the procedure is based on. The states of the alternating Büchi automaton to be constructed will be formulas that are Boolean combined within a run to obligations we still have to meet wrt. $\models_W$. Faced with an obligation $\varphi_1 \vee \varphi_2$ and reading an action, the automaton employs the action to prove either φ_1 or φ_2 choosing in a run the respective successor node for $\varphi_1 \vee \varphi_2$. Negation is treated by means of the so-called dual of a formula, which is defined further below. At this point we should mention that the current channel state is carried along while rewriting the obligations. Reading an action, we simultaneously access the corresponding output value. In this way, we are able to unambiguously relate send and receive actions and consequently to reason about configurations. Let us turn towards the more involved cases of suitably handling the temporal operators $\langle\sigma\rangle^Z$ and $\Diamond_Y^Z$.

We first suppose to face a formula $\langle\tau\rangle^Z\varphi$ reading an action $\sigma = \mathrm{S}_p^q(\lambda)$ that yields the output π in return. It turns out that three cases have to be distinguished.

- Assume that $\tau = \sigma$. Indeed, we have realised τ in the current configuration, and the further obligation can be reduced to φ.

- According to the definition of $\models_W$, the automaton will rewrite the obligation to $\perp$ if though $P(\tau) = p$ but $\tau \neq \sigma$, because then a τ-step is no more possible in that configuration in which we wanted to see it.

- Consider the case that $P(\tau) = p$ does not hold. Even if it did not occur so far, a τ-step might be hypothetically possible further on in the configuration we just left. But in case that τ is a receive action, we have to take into account that a further action $\sigma' = \tau$ which we read afterwards must not correspond to σ as the suitable receive event. τ would have occurred too late, i.e. not in the original configuration. We therefore add π to the indexing set of the $\langle\tau\rangle^Z$-modality resulting in the further obligation $\langle\tau\rangle^{Z+\pi}\varphi'$ where φ' arises from inductively applying σ to φ (here $Z+\pi$ means $Z \cup \{\pi\}$).

Thus, in case of $\sigma \in \Sigma_{\mathcal{R}}$, we furthermore have to pay attention that the action the automaton reads conforms to the condition attached to Z.

Rewriting the other temporal obligations is quite similar. Supposed to face $\Diamond_Y^Z\varphi$, for example, and reading a send action σ together with π, we may employ σ to prove φ assuming that the future configuration in demand is at least partly reached. Further actions that depend on σ, i.e. a corresponding receive action or actions from the same process, necessarily have to prove the obligations emerging from φ, too. We therefore undertake the appropriate incrementation of the respective indexing sets resulting in $\Diamond_{Y+P(\sigma)}^{Z+\pi}\varphi'$. But reading σ, we could likewise decide either to be already completely in the required configuration or that σ does not have to be employed to prove φ.

The state space of our automaton consists of all subformulas obtained by transformations as described above combined with the current channel state. The *extended closure* of φ ($ecl_C(\varphi)$) is defined wrt. a channel C as the least set

- containing all subformulas as usual as well as $\perp$, $ev(\sigma, \pi)$ (where $\sigma \in \Sigma_S$ and π is an output value of C), and fail,

- being closed under positive Boolean combination,

- containing all those formulas we obtain from $\psi \in ecl_C(\varphi)$ when we
  - negate any of its subformulas identifying $\neg\neg\psi'$ with ψ', or
  - replace a subformula $\langle\sigma\rangle\psi'$ of ψ with $\langle\sigma\rangle^Z\psi'$ or a subformula $\Diamond\psi'$ with $\Diamond_Y^Z\psi'$ ($Y \subseteq \mathcal{P}_N$ and Z is a set of output values of C), or
  - replace any of its subformulas ψ' with a subformula of ψ'.

We assume all positive Boolean formulas to be in disjunctive normal form and reduced wrt. idempotence and commutation, and we maintain the following proposition (see also [4]).

Proposition 2 *For $\varphi \in$ LTML$^-$, $ecl_C(\varphi)$ is a finite set.*

The decision procedure will also make use of the so-called *dual* $\overline{\varphi}$ of a formula φ, which we obtain by pushing negation inwards as far as possible according to $\overline{\top} = \perp$, $\overline{\neg\varphi} = \varphi$, $\overline{\varphi \vee \psi} = \overline{\varphi} \wedge \overline{\psi}$, $\overline{\langle\sigma\rangle\varphi} = \neg\langle\sigma\rangle\varphi$, and so on.

Wrt. a formula $\varphi \in$ LTML$^-$ and a finite channel $C = (S, \delta, \delta_{\mathbb{N}}, s_{in}, AC)$, we now propose the core of our decision procedure. The rewrite function $\gamma : ecl_C(\varphi) \times S \times \Sigma \to \mathcal{B}^+(ecl_C(\varphi))$ implements the ideas we illustrated before and subsequently provides (a little modified) the transition function of the automaton in demand. We write $\gamma_s(\eta, \sigma)$ instead of $\gamma(\eta, s, \sigma)$, and for fixed s, $\gamma_s(\eta, \sigma)$ is inductively defined as shown in Figure 3.

Without loss of generality, we now assume an LTML$^-$-formula φ to satisfy $\varphi = \overline{\overline{\varphi}}$, i.e., negation is already pushed inwards as far as possible.

Definition 4 *Given an LTML$^-$-formula φ and a channel $C = (S, \delta, \delta_{\mathbb{N}}, s_{in}, AC)$, the alternating Büchi automaton $\mathcal{A}_\varphi^C$ is defined by its components as follows:*

- $Q = ecl_C(\varphi) \times S$.

- $\chi((\psi, s), \sigma) = \Theta_{\delta(s,\sigma)}(\gamma_s(\psi, \sigma))$ *where $\Theta_{s'}$ distributes s' among the atoms Boolean combined in its argument, i.e., $\bigvee\bigwedge\varphi_{ij}$ becomes $\bigvee\bigwedge(\varphi_{ij}, s')$.*

- $q_{in} = (\varphi, s_{in})$.

- *F contains a state (ψ, s) iff ψ does not constitute further obligations (for example $\psi = \neg\langle\sigma\rangle\top$). Furthermore, $\widehat{F} = \{(\neg\psi, s) \mid \neg\psi \in ecl_C(\varphi)\} \cup \{\top\}$.*

$$
\begin{array}{lll}
\eta, \sigma & \mapsto & \text{fail } \textit{if } \delta(s,\sigma) = \text{fail}, \textit{ otherwise}: \\
\eta, \sigma & \mapsto & \eta \textit{ if } \eta \in \{\text{fail}, \top, \bot\} \\
\neg\varphi, \sigma & \mapsto & \overline{\gamma_s(\varphi,\sigma)} \\
\varphi_1 \vee \varphi_2, \sigma & \mapsto & \gamma_s(\varphi_1,\sigma) \vee \gamma_s(\varphi_2,\sigma) \\
\langle \mathrm{S}_p^q(\lambda) \in S\rangle\varphi, \sigma & \mapsto & \begin{cases} \bot & \textit{if } P(\sigma) = p \textit{ and } \sigma \neq \mathrm{S}_p^q(\lambda) \\ \langle \mathrm{S}_p^q(\lambda) \in S\rangle \gamma_s(\varphi,\sigma) & \textit{if } P(\sigma) \neq p \\ \varphi \wedge ev(\mathrm{R}_p^q(\lambda), \delta_{\mathbb{N}}(s,\sigma)) & \textit{if } \sigma = \mathrm{S}_p^q(\lambda) \end{cases} \\
ev(\mathrm{R}_p^q(\lambda),\pi), \sigma & \mapsto & \begin{cases} \bot & \textit{if } \sigma = \mathrm{S}_p^q(\lambda) \textit{ and } \delta_{\mathbb{N}}(s,\sigma) = \pi \\ ev(\mathrm{R}_p^q(\lambda),\pi) & \textit{if } \delta_{\mathbb{N}}(s,\sigma) \neq \pi \\ \top & \textit{if } \sigma = \mathrm{R}_p^q(\lambda) \textit{ and } \delta_{\mathbb{N}}(s,\sigma) = \pi \end{cases} \\
\langle\tau\rangle\varphi, \sigma & \mapsto & \gamma_s(\langle\tau\rangle^{\emptyset}\varphi, \sigma) \\
\Diamond\varphi, \sigma & \mapsto & \gamma_s(\Diamond_{\emptyset}^{\emptyset}\varphi, \sigma) \\
\multicolumn{3}{l}{\quad \textit{let} \quad \pi = \delta_{\mathbb{N}}(s, \mathrm{S}_p^q(\lambda))} \\
\multicolumn{3}{l}{\qquad\qquad \sigma = \mathrm{S}_p^q(\lambda)} \\
\langle\tau\rangle^Z\varphi, \mathrm{S}_p^q(\lambda) & \mapsto & \begin{cases} \bot & \textit{if } P(\tau) = p \textit{ and } \tau \neq \sigma \\ \langle\tau\rangle^{Z+\pi}\gamma_s(\varphi,\sigma) & \textit{if } P(\tau) \neq p \\ \varphi & \textit{if } \tau = \sigma \end{cases} \\
\Diamond_Y^Z\varphi, \mathrm{S}_p^q(\lambda) & \mapsto & \begin{cases} \Diamond_Y^{Z+\pi}\gamma_s(\varphi,\sigma) \vee \gamma_s(\varphi,\sigma) & \textit{if } p \in Y \\ \Diamond_{Y+p}^{Z+\pi}\gamma_s(\varphi,\sigma) \vee \gamma_s(\varphi,\sigma) \vee \Diamond_Y^Z\varphi & \textit{if } p \notin Y \end{cases} \\
\multicolumn{3}{l}{\quad \textit{let} \quad \pi = \delta_{\mathbb{N}}(s, \mathrm{R}_p^q(\lambda))} \\
\multicolumn{3}{l}{\qquad\qquad \sigma = \mathrm{R}_p^q(\lambda)} \\
\langle\tau\rangle^Z\varphi, \mathrm{R}_p^q(\lambda) & \mapsto & \begin{cases} \bot & \textit{if } (P(\tau) = q \textit{ and } \tau \neq \sigma) \textit{ or } (\tau = \sigma \textit{ and } \pi \in Z) \\ \langle\tau\rangle^{Z}\gamma_s(\varphi,\sigma) & \textit{if } P(\tau) \neq q \\ \varphi & \textit{if } \tau = \sigma \textit{ and } \pi \notin Z \end{cases} \\
\Diamond_Y^Z\varphi, \mathrm{R}_p^q(\lambda) & \mapsto & \begin{cases} \Diamond_{Y+q}^{Z}\gamma_s(\varphi,\sigma) \vee \gamma_s(\varphi,\sigma) & \textit{if } q \in Y \textit{ or } \pi \in Z \\ \Diamond_{Y+q}^{Z}\gamma_s(\varphi,\sigma) \vee \gamma_s(\varphi,\sigma) \vee \Diamond_Y^Z\varphi & \textit{if } q \notin Y \textit{ and } \pi \notin Z \end{cases}
\end{array}
$$

Figure 3. The rewrite function

Theorem 1 finally records the correctness of our construction meeting decidability of the model checking problem and satisfiability wrt. a channel.

Theorem 1 *For a formula $\varphi \in \mathrm{LTML}^-$, a channel C, and the alternating Büchi automaton $\mathcal{A}_\varphi^C$,*

$$
\begin{array}{rcl}
\mathcal{L}(\mathcal{A}_\varphi^C) & = & \{w \mid w \in \mathcal{L}(C) \textit{ and } M^w_{\rho_{\mathbb{N}}(w)} \models \varphi\} \\
& = & \{w \mid w \in \mathcal{L}(C) \textit{ and } \rho^w_{\mathbb{N}}(w) \models_W \varphi\}.
\end{array}
$$

Proof The theorem is a final consequence of the correctness of our construction presented in [4]. Assumed a certain familiarness with the notion of Mazurkiewicz traces, we want to shortly go into the relation between them and MSCs. Wrt. Mazurkiewicz traces, the relation $\sim$ is defined wrt. a dependency relation which declares actions interdependent rather than pairs of actions and natural numbers. Within our framework, we must have a look at the channel when verifying interdependency of positions at a given configuration. This illustrates why we have to carry along the current channel state. □

Corollary 1 now justifies the application of partial order reduction methods.

Corollary 1 *$\mathcal{A}_\varphi^C$ is closed in the following sense. If $w, w' \in \mathcal{L}(C)$ and $\rho^w_{\mathbb{N}}(w) \sim \rho^{w'}_{\mathbb{N}}(w')$ then $w \in \mathcal{L}(\mathcal{A}_\varphi^C)$ iff $w' \in \mathcal{L}(\mathcal{A}_\varphi^C)$.*

Given an MPA $\mathcal{A} = (\{\mathcal{A}_p\}_{p \in \mathcal{P}_N}, C, Q_{in})$ and a formula $\varphi \in \mathrm{LTML}^-$, we are now able to decide satisfiability wrt. C by testing whether $\mathcal{L}(\mathcal{A}_\varphi^C) \neq \emptyset$ and to decide the model checking problem by testing whether $\mathcal{L}(\mathcal{A}) \subseteq \{w \mid w \in \mathcal{L}(C) \textit{ and } M^w_{\rho_{\mathbb{N}}(w)} \models \varphi\}$. The latter is the case iff $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}_\varphi^C)$ and iff $\mathcal{L}(\mathcal{A} \times \mathcal{A}_{\neg\varphi}^C)$ is empty. $\mathcal{A} \times \mathcal{A}_{\neg\varphi}^C$, which to construct is a simple task, is the product automaton accepting all those words accepted by both $\mathcal{A}$ and $\mathcal{A}_{\neg\varphi}^C$.

Our procedure for full LTML is nonelementary, which is unavoidable, but only in the number of nested until-formulas which are seldom in practice. Some fragments of LTML show more positive complexity. See [4] for a detailed analysis.

5.2 A note on the use of nondeterministic channels

In our presentation, we required channels to behave deterministically. For practical applications though, it is desirable to extend our approach towards nondeterministic channels. Unfortunately, one and the same word defined by a nondeterministic channel may stand for different MSCs. That way, the unambiguous mutual assignment of MSCs and their sets of linearisations, which our procedure is based on, gets lost. Nevertheless, we can solve this problem by likewise introducing linearisations as well-formed words in such a way as we regain aforesaid unambiguousness [5].

6. Conclusion

We presented a way to formally deal with message passing systems and have given a slightly adapted presentation of message sequence charts (MSCs) which may be used to describe scenarios of an underlying message passing system. We presented a model for channels employed for communication which is applicable for a broad range of different channel types. We introduced message passing automata as a finite description for sets of MSCs.

To specify message passing systems in the light of the powerful approach via temporal logic formulas, we proposed a version of the prominent linear time temporal logic (LTL) adapted towards MSCs.

To support automatic verification of specified systems, we developed a corresponding decision procedure which makes it possible to reason about finite state message passing systems using fixed but arbitrarily modelled deterministic channels.

Altogether, we presented a framework for modelling, specifying, and verifying message passing systems.

References

[1] R. Alur, G. Holzmann, and D. Peled. An analyzer for message sequence charts. *Software: Concepts and Tools*, 17:70–77, 1996. also appeared in TACAS'96, Tools and Algorithms for the Construction and Analysis of Systems, Passau, Germany, LNCS 1055, Springer-Verlag, 1996, 35-48.

[2] R. Alur and M. Yannakakis. Model checking of message sequence charts. In *Proc. 10th Intl. Conf. on Concurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 114–129. Springer Verlag, 1999.

[3] J. Araújo. Formalizing sequence diagrams. In L. Andrade, A. Moreira, A. Deshpande, and S. Kent, editors, *Proceedings of the OOPSLA'98 Workshop on Formalizing UML. Why? How?*, volume 33, 10 of *ACM SIGPLAN Notices*, New York, 1998. ACM Press.

[4] B. Bollig and M. Leucker. Deciding LTL over Mazurkiewicz Traces. In *Proceedings of the Symposium on Temporal Representation and Reasoning (TIME01)*. IEEE Computer Society Press, 2001.

[5] B. Bollig, M. Leucker, and T. Noll. Regular MSC Languages. Technical Report AIB-05-2001, RWTH Aachen, Apr. 2001.

[6] J. G. Henriksen, M. Mukund, K. N. Kumar, and P. S. Thiagarajan. Regular collections of message sequence charts. In *Proceedings of 25th International Symposium on Mathematical Foundations of Computer Science (MFCS'2000)*, volume 1893 of *Lecture Notes in Computer Science*, pages 405–414. Springer-Verlag, 2000.

[7] ITU-TS. ITU-TS Recommendation Z.120: Message Sequence Chart 1996 (MSC96). Technical report, ITU-TS, Geneva, 1996.

[8] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[9] L. Lamport and N. Lynch. Distributed computing: Models and methods. In J. van Leewen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 19, pages 1157–1199. The MIT Press, New York, NY, 1990.

[10] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer, New York, 1992.

[11] B. Meenakshi and R. Ramanujam. Reasoning about message passing in finite state environments. In *Proceedings of 27th International Colloquium on Automata, Languages and Programming (ICALP'2000)*, volume 1853 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.

[12] A. Muscholl, Z. Su, and D. Peled. Deciding properties for message sequence charts. In *Foundations of Software Science and Computation Structures (FoSSaCS'98)*, volume 1578 of *LNCS*, Lisbon, Portugal, 1998. Springer Verlag.

[13] D. Peled. Ten years of partial order reduction. In *CAV, Computer Aided Verification*, number 1427 in LNCS, pages 17–28, Vancouver, BC, Canada, 1998. Springer.

[14] A. P. Sistla, E. M. Clarke, N. Francez, and A. R. Meyer. Can message buffers be axiomatized in linear temporal logic? *Information and Control*, 63(1/2):88–112, Oct./Nov. 1984.

[15] A. P. Sistla and L. D. Zuck. Automatic temporal verification of buffer systems. In K. G. Larsen and A. Skou, editors, *Proceedings of Computer Aided Verification (CAV '91)*, volume 575 of *LNCS*, pages 59–69, Berlin, Germany, July 1992. Springer.

[16] P. S. Thiagarajan. PTL over product state spaces. Technical Report TCS-95-4, School of Mathematics, SPIC Science Foundation, 1995.

[17] P. S. Thiagarajan and I. Walukiewicz. An expressively complete linear time temporal logic for Mazurkiewicz traces. In *Proceedings, Twelth Annual IEEE Symposium on Logic in Computer Science*, pages 183–194, Warsaw, Poland, 29 June–2 July 1997. IEEE Computer Society Press.

[18] A. Valmari. A stubborn attack on state explosion. In E. M. Clarke and R. P. Kurshan, editors, *Proceedings of Computer-Aided Verification (CAV '90)*, volume 531 of *LNCS*, pages 156–165, Berlin, Germany, June 1991. Springer.

[19] M. Y. Vardi. *An Automata-Theoretic Approach to Linear Temporal Logic*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer-Verlag Inc., New York, NY, USA, 1996.