Communicating automata

Dietrich Kuske[1] *and Anca Muscholl*[2]

[1]TU Ilmenau, Institut für Theoretische Informatik

[2] LaBRI, Université Bordeaux and CNRS

2010 Mathematics Subject Classification: Primary: 68Q68 Secondary: 68Q45, 68Q60

Key words: asynchronous peer-to-peer communication, communicating automata, message-passing systems, model checking, realizability

Contents

1 Introduction 1030
2 Communicating Automata 1032

I Reachability problems 1035
3 Symbolic representations 1036
4 Faulty channel systems 1038

II Specifications and Model-Checking 1040
5 Sequential specifications 1040
6 Partial order specifications 1042
6.1 Message sequence charts 1042
6.2 Monadic second-order logic 1045
6.3 Propositional dynamic logic 1047
7 Model-checking 1048
7.1 Channel bounds 1048
7.2 Bounded CFMs 1049
7.3 Model-checking with channel bounds 1050

III Realizability 1053
8 Realizability with unrestricted channels 1054
8.1 Local acceptance and sequential specifications 1054

8.2 MSO specifications 1055
8.3 PDL specifications 1056
9 Channel bounds, Mazurkiewicz traces and realizability 1057
9.1 Mazurkiewicz traces 1058
9.2 Bounds and traces 1059
9.3 Realizability with bounded channels 1061
References 1064

1 Introduction

Communicating automata, or equivalently, message-passing systems, are a key computational model for asynchronous, concurrent systems. In this simple model, a fixed number of processes cooperate via asynchronous message passing over unbounded, peer-to-peer channels. This model is very popular for modeling and reasoning about communication protocols; for instance the ITU specification and description language SDL (Specification and Description Language) relies on it to define the semantics of communication. Automata-based verification of properties of communicating automata is, nevertheless, quite challenging, since these machines are Turing-equivalent [22], subsuming, e.g., Post tag systems [74]. An immediate consequence is that every non-trivial property is undecidable, so that it is essential to look for approximate or semi-algorithmic verification techniques. This survey covers some of the automated verification techniques for the analysis of communicating automata, with particular emphasis on methods relying on partial-order semantics.

We consider two core problems about asynchronously communicating processes: the *model-checking* problem and a simpler variant of synthesis, called the *realizability* problem. The model-checking problem asks if all behaviors of the model (automaton) are correct with respect to some given property, usually described in a logical formalism such as, e.g., temporal logics [61]. The realizability problem is a simple variant of the more general synthesis question [27], which asks to compute a model which is correct w.r.t. a given specification. It should be noted that the realizability problem is already quite challenging for asynchronous systems, since it amounts to distributing the specification on a given process architecture, which limits the way information is exchanged. In contrast, general synthesis immediately leads to questions about distributed games (also see Chapter 33) that are either undecidable [73], or are still open, even on simpler models with rendez-vous synchronization [68].

Semi-algorithmic verification techniques mostly fall into two categories: *under-* and *over-approximating* methods. Under-approximating methods restrict the behavior of a system, while over-approximating ones consider relaxations. As an example, ignoring the order of messages in the channels is an over-approximation that turns communicating automata into Petri nets. Bounding the capacity of channels is an under-approximation that turns communicating automata into finite automata. Approaches using a finitary representation of possibly infinite sets of configurations (called symbolic representations) are another example of under-approximation. In the case of communicating automata, symbolic representations may be based on finite automata or some tractable, extended automata models [10, 11, 20].

An interesting instance of over-approximation is the setting where communication channels can lose messages. Lossy channel systems are a particular instance of well-structured transition systems [1, 37], and this implies that their reachability problem is decidable [2], albeit of non-primitive recursive complexity [75]. On the other hand, more general properties, such as liveness, are undecidable [1].

The above mentioned approaches are based on *symbolic representations of reachable configurations* of communicating automata. In this survey, we focus on analysis based on partial orders. The *partial-order* approach is event-based and emphasizes the causally-ordered executions of communicating automata. Its main advantage is that the realizability problem can be stated in a natural way, since the specifications and behavior of communicating automata are both captured by the same kind of partial orders, namely *message sequence charts*.

The partial-order approach provides solutions for both the model-checking and the realizability problem, under the assumption of some bounds on channels. A communicating automaton has universal channel bound B [49] if every execution requires only channels of size B. An existential channel bound B is a more relaxed condition, it means that any execution can be re-scheduled in such a way that it requires only channels of size B [46, 43]. A very simple example illustrating the idea of existential bounds is a pair of processes, a producer and a consumer, where the producer loops sending messages to the consumer, and the consumer receives them. Since there is no control on the relative speed of the two processes, there is no *a priori* bound on the number of transiting messages. However, if we are interested in, e.g., control-state reachability, then it suffices to consider only schedules where the messages are consumed without delay, so $B = 1$. Since the above system is existentially 1-bounded, the restriction to 1-bounded schedules suffices to check the required property. An even stronger result is an effective solution for the realizability problem for communicating automata with existential channel bounds w.r.t. logical properties [43] (for universal bounds, see [49]). From a language-theoretic perspective the result for the realizability problem can be stated as a Kleene-Büchi theorem for languages of message sequence charts.

Overview. Section 2 introduces the automaton model and the questions we consider in this survey. In Part I we summarize results on symbolic representations and lossy channel systems. In Part II we introduce various specification frameworks, ranging from logics to message sequence charts. Then we discuss solutions for the model-checking problem. In Part III we present solutions for the realizability problem w.r.t. the specifications introduced in Part II.

Several recent research directions are not included in this chapter — in particular verification methods for asynchronously communicating processes with additional storage capabilities (e.g., pushdown) [8, 55, 50, 30, 31, 28, 29]. Very recently, the same roadmap as the approach based on existential channel bounds presented in this chapter was followed in [19], but for communicating automata with mailbox semantics, where every process has a single incoming mailbox.

Acknowledgments: We would like to thank Benedikt Bollig, Blaise Genest, and Paul Gastin for many useful comments and discussions.

2 Communicating Automata

Communicating automata follow the simple paradigm of a network of automata cooperating asynchronously over point-to-point, FIFO communication channels. They arise naturally as models for peer-to-peer interaction, as occurring, e.g., in distributed protocols using asynchronous message passing. We consider systems described by means of a fixed *communication network*, consisting of a finite set of concurrent *processes* $\mathcal{P}$, together with a set of *channels* $\mathrm{Ch} \subseteq \{(p,q) \in \mathcal{P}^2 \mid p \neq q\}$, that stand for point-to-point links [22]. Following the classical definitions, we exclude multiple channels between a pair of processes, as well as self-linking channels. However, this restriction has no big impact on the kind of results we will present. In our model, processes act either by point-to-point communication or by local actions. Possible actions come from a (finite) *communication alphabet* Σ, that is parametrized by the network $(\mathcal{P}, \mathrm{Ch})$, a set Msg of message contents, and a set Act of local actions (for convenience we omit $\mathcal{P}, \mathrm{Ch}, \mathrm{Msg}, \mathrm{Act}$ from the notation Σ). A send action denoted by $p!q(m)$ means that process p sends a message with content $m \in \mathrm{Msg}$ to process q on channel $(p,q) \in \mathrm{Ch}$. A receive action denoted by $p?q(m)$ means that p receives a message from q with content $m \in \mathrm{Msg}$ on channel $(q,p) \in \mathrm{Ch}$. A local action denoted by c_p means that process p performs the action $c \in \mathrm{Act}$. The set of p-actions (equivalently, the p-local alphabet) equals $\Sigma_p = \{p!q(m), p?r(m), c_p \mid (p,q) \in \mathrm{Ch}, (r,p) \in \mathrm{Ch}, m \in \mathrm{Msg}, c \in \mathrm{Act}\}$, and we let $\Sigma = \bigcup_{p \in \mathcal{P}} \Sigma_p$ denote the set of all actions. As usual, Σ^* and Σ^ω denote the set of finite and infinite sequences over Σ, respectively, and $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$.

Definition 2.1. A *communicating automaton* (*CA* for short) is a triple $\mathcal{A} = \langle (\mathcal{A}_p)_{p \in \mathcal{P}}, \Sigma, F \rangle$ where

- each $\mathcal{A}_p = (S_p, \rightarrow_p, s_p^0)$ is a labeled transition system with state space S_p, transition relation $\rightarrow_p \subseteq S_p \times \Sigma_p \times S_p$, and initial state $s_p^0 \in S_p$;
- $F \subseteq \prod_{p \in \mathcal{P}} S_p$ is a set of *global* final states.

We call the product $S := \prod_{p \in \mathcal{P}} S_p$ the set of *global states.*

The *size* of a CA $\mathcal{A} = \langle (\mathcal{A}_p)_{p \in \mathcal{P}}, \Sigma, F \rangle$ with message contents Msg is defined as $\sum_{p \in \mathcal{P}} |S_p| + |\,\mathrm{Msg}\,| + |\,\mathrm{Act}\,|$.

The behavior of a CA is defined as the behavior of an infinite labeled transition system, considering the possible (local) transitions on the set of configurations of the CA. A *configuration* of the CA $\mathcal{A}$ consists of a global state, together with a word from Msg^* for each channel $(p,q) \in \mathrm{Ch}$. We write $C = \langle s, w \rangle$ for a configuration with global state $s \in S$ and channel contents $w \in (\mathrm{Msg}^*)^{\mathrm{Ch}}$, and let s_p and $w_{p,q}$ denote the p-component of s and the (p,q)-component of w, respectively. The set of all configurations of $\mathcal{A}$ is denoted $\mathcal{C}_{\mathcal{A}}$ (or simply $\mathcal{C}$ when there is no risk of confusion). The *initial configuration* of $\mathcal{A}$ is $C_0 = \langle s^0, \vec{\varepsilon} \rangle$ with $\vec{\varepsilon}_{p,q} = \varepsilon$ (the empty word) for all $(p,q) \in \mathrm{Ch}$. A configuration $C = \langle s, w \rangle$ is *final* if $s \in F$ is a final state of $\mathcal{A}$ (note that the channels need not be empty in a final configuration).

For two configurations $C = \langle s, w \rangle, C' = \langle s', w' \rangle$ and an action $a \in \Sigma_p$, we write $C \xrightarrow{a} C'$ if the following hold:

- $s_p \xrightarrow{a}_p s_p'$ is a transition of $\mathcal{A}_p$ (i.e., $(s_p, a, s_p') \in \rightarrow_p$), and $s_q' = s_q$ for all $q \neq p$,

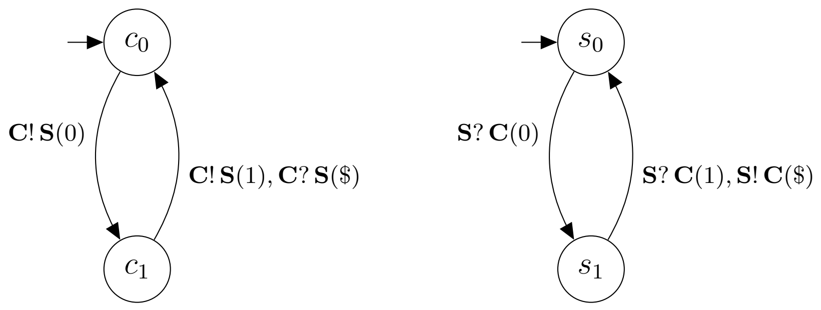
c_0
c_1
$\mathbf{C}!\,\mathbf{S}(0)$
$\mathbf{C}!\,\mathbf{S}(1), \mathbf{C}?\,\mathbf{S}(\$)$
s_0
s_1
$\mathbf{S}?\,\mathbf{C}(0)$
$\mathbf{S}?\,\mathbf{C}(1), \mathbf{S}!\,\mathbf{C}(\$)$

Figure 1. A CFM on two processes, $\mathbf{C}$ left, and $\mathbf{S}$ right.

- if $a = p!q(m)$ is a send action, then $w'_{p,q} = w_{p,q}m$ (message m is inserted into the channel from p to q) and $w'_{r,s} = w_{r,s}$ for all $(r, s) \neq (p, q)$ (all other channels are unchanged).
- if $a = p?q(m)$ is a receive action, then $w_{q,p} = mw'_{q,p}$ (message m is deleted from the channel from q to p) and $w'_{r,s} = w_{r,s}$ for all $(r, s) \neq (q, p)$ (all other channels are unchanged).
- if $a = c_p$ is a local action, then $w = w'$.

We say that C' is a *successor* of C (and write $C \longrightarrow C'$) if there exists some $a \in \Sigma$ with $C \xrightarrow{a} C'$. As usual, we write $\xrightarrow{*}$ for the reflexive-transitive closure of $\longrightarrow$. For a set $X \subseteq \mathcal{C}$ of configurations, we write $\mathrm{post}(X)$ for the set of successors of configurations from X and $\mathrm{post}^*(X)$ for the set of all *reachable* configurations from X:

$$\begin{aligned}\mathrm{post}(X) &:= \{C' \in \mathcal{C} \mid \exists C \in X \,:\, C \longrightarrow C'\}\,,\\ \mathrm{post}^*(X) &:= \{C' \in \mathcal{C} \mid \exists C \in X \,:\, C \xrightarrow{*} C'\}\,.\end{aligned}$$

The *reachability set* of a CA $\mathcal{A}$, denoted $\mathrm{Reach}(\mathcal{A})$, is the set

$$\mathrm{Reach}(\mathcal{A}) = \mathrm{post}^*(\{C_0\})$$

(recall that C_0 is the initial configuration of $\mathcal{A}$).

A CA is *deadlock-free* if from every reachable configuration it can reach a final configuration.

Example 2.1. The CA in Figure 1 describes the communication between two (finite-state) processes $\mathbf{C}$ and $\mathbf{S}$, connected through one channel in each direction (process $\mathbf{C}$ on the left, and $\mathbf{S}$ on the right). The set of message contents is $\mathrm{Msg} = \{0, 1, \$\}$. From the initial configuration $\langle(c_0, s_0), (\varepsilon, \varepsilon)\rangle$ (say, $(\mathbf{C}, \mathbf{S})$ is the first channel) the configurations $\langle(c_1, s_0), (010, \varepsilon)\rangle$ and $\langle(c_0, s_0), (101, \$)\rangle$ are reachable, but not $\langle(c_0, s_0), (0101, \$)\rangle$. For instance, $\langle(c_0, s_0), (\varepsilon, \varepsilon)\rangle \xrightarrow{\mathbf{C}!\,\mathbf{S}(0)} \langle(c_1, s_0), (0, \varepsilon)\rangle \xrightarrow{\mathbf{C}!\,\mathbf{S}(1)} \langle(c_0, s_0), (01, \varepsilon)\rangle$.

A CA $\mathcal{A}$ is called *deterministic* if for every local state $s \in S_p$ the following hold:

- $s \xrightarrow{a}_p s_1$ and $s \xrightarrow{a}_p s_2$ implies $s_1 = s_2$ for every $a \in \Sigma_p$,
- $s \xrightarrow{p!q(m_1)}_p s_1$ and $s \xrightarrow{p!q(m_2)}_p s_2$ (for some s_1, s_2) implies $m_1 = m_2$.

$$\begin{tikzcd}
\langle (c_0,s_0),((10)^*(01)^*,\varepsilon)\rangle \arrow[r, bend left, "\mathbf{C}!\,\mathbf{S}(0)"] \arrow[d, bend left, "\mathbf{S}?\,\mathbf{C}(0)"] & \langle (c_1,s_0),((10)^*(01)^*0,\varepsilon)\rangle \arrow[l, bend left, "\mathbf{C}!\,\mathbf{S}(1)"] \arrow[d, bend left, "\mathbf{S}?\,\mathbf{C}(0)"] \\
\langle (c_0,s_1),(1(01)^*,\varepsilon)\rangle \arrow[u, bend left, "\mathbf{S}?\,\mathbf{C}(1)"] \arrow[r, bend left, "\mathbf{C}!\,\mathbf{S}(0)"] \arrow[d, "\mathbf{S}!\,\mathbf{C}(\$)"] & \langle (c_1,s_1),((10)^*,\varepsilon)\rangle \arrow[u, bend left, "\mathbf{S}?\,\mathbf{C}(1)"] \arrow[l, bend left, "\mathbf{C}!\,\mathbf{S}(1)"] \arrow[d, "\mathbf{S}!\,\mathbf{C}(\$)"] \\
\langle (c_0,s_0),(1(01)^*,\$)\rangle \arrow[r, bend left, "\mathbf{C}!\,\mathbf{S}(0)"] & \langle (c_1,s_0),((10)^*,\$)\rangle \arrow[l, bend left, "\mathbf{C}!\,\mathbf{S}(1)"] \arrow[uul, "\mathbf{C}?\,\mathbf{S}(\$)"']
\end{tikzcd}$$

Figure 2. An abstract view of the transition system.

The notion of determinism used above originates from [49]. The second condition in this definition is motivated by viewing message contents as control information that has to be chosen deterministically by the sending process.

Definition 2.2. An *execution* or *run* of a CA $\mathcal{A}$ is a (finite or infinite) sequence of transitions: $\rho = C_1 \xrightarrow{a_1} C_2 \xrightarrow{a_2} C_3 \xrightarrow{a_3} \cdots$, with configurations $C_i \in \mathcal{C}_\mathcal{A}$ and actions $a_i \in \Sigma$. The *labeling* of the run ρ, denoted $\ell(\rho)$, is the sequence of actions $a_1 a_2 \cdots \in \Sigma^\infty$.

A finite run $\rho = C_1 \xrightarrow{a_1} C_2 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} C_n$ is *accepting* if $C_1 = \langle s^0, \vec{\varepsilon}\rangle$ is the initial configuration and $C_n \in F \times (\mathrm{Msg}^*)^{\mathrm{Ch}}$ is a final configuration.

The *language* of a CA $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^*$, is the set of labelings of accepting executions of $\mathcal{A}$:

$$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid w = \ell(\rho) \text{ for some accepting run } \rho\}\,.$$

Example 2.2. An abstract view of the transition system associated with the CA $\mathcal{A}$ from Example 2.1 is given in Figure 2. There, we grouped certain configurations and connected them by action-labeled edges such that the initial configuration belongs to the group at the top-left corner and every a-successor of every configuration of some group belongs to the a-successors of that group. One can easily verify that the groups of configurations form the least solution to this constraint system. Hence the reachability set $\mathrm{Reach}(\mathcal{A})$ is the union of all the configurations mentioned in that figure.

Assuming that all states of $\mathcal{A}$ are final, the language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$ consists of words over the alphabet $\Sigma = \{\mathbf{C}!\,\mathbf{S}(0), \mathbf{S}?\,\mathbf{C}(0), \mathbf{C}!\,\mathbf{S}(1), \mathbf{S}?\,\mathbf{C}(1)\, \mathbf{S}!\,\mathbf{C}(\$), \mathbf{C}?\,\mathbf{S}(\$)\}$ of the following form: the projections on $\Sigma_\mathbf{C}$ and on $\Sigma_\mathbf{S}$, resp., belong to the prefix closures of $\big((\mathbf{C}!\,\mathbf{S}(0)\ \mathbf{C}!\,\mathbf{S}(1))^*\ \mathbf{C}!\,\mathbf{S}(0)\ \mathbf{C}?\,\mathbf{S}(\$)\big)^*$ and $\big((\mathbf{S}?\,\mathbf{C}(0)\ \mathbf{S}?\,\mathbf{C}(1))^*\ \mathbf{S}?\,\mathbf{C}(0)\ \mathbf{S}!\,\mathbf{C}(\$)\big)^*$, respectively, and the words are well-formed w.r.t. the send/receives (cf. definition of "valid word" in Section 5). In particular, the process $\mathbf{S}$ sends at most one more message than process $\mathbf{C}$ receives.

Notice that we did not impose any restriction on the local automaton $\mathcal{A}_p$ in the definition of a CA $\mathcal{A} = \langle (A_p)_{p\in\mathcal{P}}, \Sigma, F\rangle$. In general, we might be interested in various kinds of (possibly infinite-state) automata, such as pushdown automata [55, 50]. However, a basic kind of CA is obtained by requiring that every $\mathcal{A}_p$ is a finite-state automaton, and then we call $\mathcal{A}$ a *communicating finite-state machine* (CFM for short). Most of the research done in the past decades on CAs focused on CFMs, and we will concentrate on them in the next sections.

Basic verification questions

One of the simplest properties that can be asked about a protocol is whether it avoids error states. The complement of such *safety* requirements is a reachability question. The *reachability problem* for CFMs in its most general form consists of computing the set $\mathrm{post}^*(\mathrm{Init})$ for some set of configurations Init. A more specific reachability question consists of asking whether $C_1 \xrightarrow{*} C_2$ for two configurations $C_1, C_2 \in \mathcal{C}$ of a CFM $\mathcal{A}$. Another instantiation of the reachability problem is control-state reachability:

> *Control-state reachability problem:* Given a CFM $\mathcal{A}$, a configuration C, and a global state $s \in S$, is some configuration with global state s reachable from C?

All instances of the reachability problem listed above are undecidable for CFMs [22], and in Part I and Section 7.1 of this survey we present different methods to tackle these problems.

Obviously, there are many more verification questions that one can ask about a given CFM. We give some examples below:

- *Termination:* Given a CFM $\mathcal{A}$ and a configuration C, are all runs of $\mathcal{A}$ from C finite?
- *Structural termination:* Given a CFM $\mathcal{A}$, are all runs of $\mathcal{A}$ from *every* configuration finite?
- *Absence of deadlocks:* Given a CFM, is it deadlock-free?
- *Boundedness:* Given a CFM $\mathcal{A}$ and a configuration C, are only finitely many configurations reachable from C?
- *Repeated reachability (Büchi acceptance):* Given a CFM $\mathcal{A}$, a configuration C, and a global state s, is there some run from C that meets s infinitely often?
- *Model-checking:* Given a CFM $\mathcal{A}$ and a property P, do all executions of $\mathcal{A}$ satisfy P?

The last of these questions is very general, as the term "property P" can be instantiated in very different ways. Here, we will concentrate on the case where P is given by some linear-time formula (see Part II for details). This survey does not discuss branching-time logical properties (see Chapter 38 for a detailed discussion about linear- versus branching-time logics).

Part I
Reachability problems

This part presents two methods to tackle the undecidability of the reachability problem for CFM: *Symbolic representations* of the reachability set help to accelerate the naive enumeration of all reachable configurations, and the *lossy channel* method allows us to compute an over-approximation of the reachability set. A third approach based on *channel bounds* will be used to compute under-approximations of the reachability set in Section 7.1.

3 Symbolic representations

The set of all reachable configurations of some CFM can be easily enumerated by a breadth-first search that requires handling finite sets of configurations. Eventually, every reachable configuration is computed. In this section, we discuss a technique that speeds up this process. There are two main ideas involved:

(1) Consider possibly infinite sets of configurations that are given symbolically through a finite description (see also Chapter 32 for more details related to symbolic representations of sets of numbers).
(2) Compute the effect of "meta-transitions" that correspond to sequences of actions of the CFM.

In the following, let $\mathcal{A} = \langle (\mathcal{A}_p)_{p\in\mathcal{P}}, \Sigma, F\rangle$ be some CFM with global state space S. Let us enumerate the channels in some fixed way, say as $\mathrm{ch}(1)$ to $\mathrm{ch}(k)$, such that a configuration $\langle s, w\rangle$ with $w = (w_{\mathrm{ch}(i)})_{i=1}^{k}$ is represented by the word $s\, w_{\mathrm{ch}(1)}\# \cdots w_{\mathrm{ch}(k)}\#$ (assuming that $\# \notin \mathrm{Msg} \cup S$).

One of the first papers that use finite automata for symbolic representations is [71]. There, the CFM $\mathcal{A}$ is said to have *recognizable channels* if the set of encodings of reachable configurations $\{sw \in S(\mathrm{Msg}^*\,\#)^k \mid \langle s, w\rangle \in \mathrm{Reach}(\mathcal{A})\}$ is a regular language. For CFMs with recognizable channels, reachability is decidable [71]. This can be checked by two semi-algorithms: one is the plain enumeration of all reachable configurations, and the other one enumerates all regular languages $L \subseteq S(\mathrm{Msg}^*\,\#)^k$, looking for one that contains the initial configuration C_0 (i.e., with $s^0\#^k \in L$), excludes the target configuration, and is closed under the one-step transition relation. Notice, however, that recognizable channels are very restrictive: a CFM containing one process that sends the same (arbitrary) sequence of messages to two other processes does not have recognizable channels.

Queue-content Decision Diagrams [10] (QDD for short) describe (possibly infinite) sets of configurations of a CFM by finite automata. The main idea here is to accelerate the computation of the reachability set – which, however, might be unrepresentable by a QDD. To this end, one analyzes the effect of iterating a loop in the transition system associated with the CFM.

More precisely, a QDD is a finite automaton accepting sequences from $(\mathrm{Msg}^*\,\#)^k$. A set of configurations X is *QDD-representable* if all configurations from X have the same global state $s \in S$ and there is a QDD accepting the set $\{w \in (\mathrm{Msg}^*\,\#)^k \mid \langle s, w\rangle \in X\}$

(i.e., if this set is regular). Now let $\sigma \in \Sigma^*$, $s \in S$ be a global state, and X be a set of configurations with global state s. Then $\mathrm{post}^*_\sigma(X)$ is the set of channel contents w' such that the configuration $\langle s, w' \rangle$ is reachable from some configuration $\langle s, w \rangle \in X$ by a path whose label belongs to σ^*.

A sequence σ is *(effectively) QDD-preserving* if $\mathrm{post}^*_\sigma(X)$ is (effectively) QDD-representable whenever X is a QDD-representable set of configurations. It is *non-counting* w.r.t. channel (p, q) if one of the following conditions holds:

- σ contains no letter $p!q$,
- the message alphabet of channel (p, q) is unary and σ contains the same number of occurrences of $p!q$ as of occurrences of $q?p$.

Theorem 3.1 ([11]). *The following assertions are equivalent for any sequence $\sigma \in \Sigma^*$:*

(1) σ is QDD-preserving,
(2) σ is effectively QDD-preserving,
(3) σ is non-counting w.r.t. all but at most one channel.

Example 3.1. We consider again the CFM from Example 2.1 (page 1033). The sequence $\sigma = \mathbf{C}!\,\mathbf{S}(0)\;\mathbf{C}!\,\mathbf{S}(1)$ is non-counting w.r.t. channel $(\mathbf{S}, \mathbf{C})$ (but not w.r.t. channel $(\mathbf{C}, \mathbf{S})$), so the theorem above can be applied with $X = \{C_0\}$, yielding $\mathrm{post}^*_\sigma(X) = \{((01)^n, \varepsilon) \mid n \geqslant 0\}$. The sequence $\sigma' = \mathbf{C}!\,\mathbf{S}(0)\;\mathbf{S}?\,\mathbf{C}(0)\;\mathbf{S}!\,\mathbf{C}(\$)\;\mathbf{C}?\,\mathbf{S}(\$)\;\mathbf{C}!\,\mathbf{S}(0)\;\mathbf{C}!\,\mathbf{S}(1)$ is non-counting w.r.t. $(\mathbf{S}, \mathbf{C})$ and $\mathrm{post}^*_{\sigma'}(X) = \{(01, \varepsilon), (1001, \varepsilon)\}$.

A semi-algorithm for model-checking LTL properties (with atomic propositions referring to global states), based on the QDD-representation, was proposed in [11].

Symbolic representations going beyond regular ones were introduced in [20]: *Constrained Queue-content Decision Diagrams* (CQDD for short). A CQDD consists of a restricted finite automaton $\mathcal{B}$, together with a Presburger constraint π. An accepting run in a CQDD is an accepting run of the automaton $\mathcal{B}$ whose Parikh image satisfies π. A word $\sigma \in \Sigma^*$ is (effectively) CQDD-preserving, if $\mathrm{post}^*_\sigma(X)$ is (effectively) CQDD-representable whenever X is a CQDD-representable set of configurations. It should be noted that CQDDs and QDDs are incomparable: CQDDs add power via the Presburger constraints, but lose on the level of the regular set, since only a restricted form of automaton is allowed. The following theorem shows that CQDDs can be used for a semi-algorithm computing the reachability set from any given initial configuration (since any finite set is CQDD-representable).

Theorem 3.2 ([20]). *Every sequence $\sigma \in \Sigma^*$ is effectively CQDD-preserving.*

As mentioned before, the above results allow speeding up the enumeration of all reachable configurations of a CFM. Suppose that we want to know whether some configuration with global state s is reachable from the initial configuration C_0. The trivial semi-algorithm mentioned at the beginning of this section performs a breath-first search in the transition system, thereby enumerating all sequences of actions that can be performed from the initial configuration. The improved semi-algorithm handles at once all sequences of the form $\rho_1\,\sigma^*\,\rho_2$ with $\rho_1, \sigma, \rho_2 \in \Sigma^*$ and computes the set $\mathrm{post}_{\rho_2}(\mathrm{post}^*_\sigma(\mathrm{post}_{\rho_1}(C_0)))$.

4 Faulty channel systems

In this section, we discuss a technique that allows us to compute a superset of the reachability set of a CFM. The idea is to add transitions to the labeled transition system derived from the CFM. This can be done in many more or less natural ways. The first possibility is to ignore the order of messages in the channels. The CFM thus becomes a Petri net, whose reachability problem is decidable [62, 51, 57] (albeit of high complexity, since no primitive recursive algorithm is known so far). However, the main problem with this approximation technique is that it is too coarse for realistic modeling.

The additional transitions considered here can either lose messages from or introduce messages into channels. Such a model is interesting in its own right, since it allows us to model imperfect channels. *Lossy* machines are CFMs where channels can lose an arbitrary number of messages, at any time. For CFMs with *insertion errors*, new messages can be inserted in channels, at any time. Although these two models have a different flavor, the techniques used to manipulate them are quite similar. However, we will see a minor qualitative difference between the results obtained for these two models (cf. Remark 4.6 below).

Lossy CFMs (or *lossy channel systems*) represent a special instance of a more general class of infinite-state systems, namely *well-structured transition systems* (WSTS for short). WSTS were considered independently by Finkel and Schnoebelen [36, 37], and by Abdulla and Jonsson [1, 2]. The basic idea behind a WSTS $\langle \mathcal{S}, \rightarrow \rangle$ is to endow the set of configurations $\mathcal{S}$ with a *well quasi-order* (wqo for short) in order to manipulate certain infinite subsets of $\mathcal{S}$ symbolically. A wqo $\preceq$ on $\mathcal{S}$ is a quasi-order without infinite antichains and without infinite decreasing chains. A transition system $\langle \mathcal{S}, \rightarrow, \preceq \rangle$ is a WSTS if it satisfies the following *monotonicity* property: for every $s \rightarrow s'$ and every $s_1 \in \mathcal{S}$ with $s \preceq s_1$, it is required that some $s_1' \in \mathcal{S}$ exists with $s_1 \rightarrow s_1'$ and $s' \preceq s_1'$. Monotonicity appears with different flavors in [37]. One variant consists in requiring $s_1 \rightarrow^+ s_1'$ in the definition above. This is then called *transitive monotonicity*.

Two properties are crucial for WSTS. The first one is that every subset $X \subseteq \mathcal{S}$ has a *finite* set of minimal elements, denoted $\min(X)$. The second property is that the predecessor relation preserves *upward-closed*[1] sets, i.e., $\mathrm{pre}(X) := \{s \mid \exists s' \in X\colon s \rightarrow s'\}$ is upward-closed whenever X is upward-closed. As a consequence, reachability of upward-closed sets X of states can be decided by a backward algorithm that computes the set $\mathrm{pre}^*(X) = \bigcup_{n \geqslant 0} \mathrm{pre}^n(X)$ as least fixpoint of the operation $Y \mapsto \mathrm{pre}(Y) \cup X$. Intersecting the result with the set of initial configurations solves the reachability problem for WSTS.

Theorem 4.1 ([2]). *Let $\langle \mathcal{S}, \rightarrow, \preceq \rangle$ be a WSTS such that $\prec$ is decidable and* $\min(\mathrm{pre}(X))$ *is computable from* $\min(X)$ *for every upward-closed set $X \subseteq \mathcal{S}$. Then, for $s, s' \in \mathcal{S}$, it is decidable whether there exists $s'' \in \mathcal{S}$ with $s \rightarrow^* s''$ and $s' \preceq s''$.*

Termination for WSTS can be decided by a forward algorithm, computing the *finite reachability tree* $\mathrm{FRT}(s)$ from a state $s \in \mathcal{S}$: this is the prefix of the reachability tree built from state s, obtained by defining a node t' as a leaf if there is some node $t \preceq t'$ on the path from s to t'. Note that the tree $\mathrm{FRT}(s)$ is finite if $\langle \mathcal{S}, \rightarrow \rangle$ has finite branching.

[1] X is upward-closed if $X = \{s' \mid s \preceq s' \text{ for some } s \in X\}$.

Theorem 4.2 ([37]). *Termination is decidable for finitely branching WSTS $\langle \mathcal{S}, \to, \preceq \rangle$ with transitive monotonicity such that $\prec$ is decidable and $\{s' \in \mathcal{S} \mid s \to s'\}$ is computable from $s \in \mathcal{S}$.*

For lossy CFMs, the choice of a wqo is very natural. One starts with the subword ordering: let $x \preceq y$ if $x = x_1 \cdots x_n$ and $y = y_0 x_1 y_1 \cdots y_{n-1} x_n y_n$ for some $x_i, y_i \in \mathrm{Msg}^*$. This wqo extends to configurations of the CFM: for two configurations $C = \langle s, w \rangle$, $C' = \langle s', w' \rangle$, let $C \preceq C'$ if $s = s'$ and $w_{p,q} \preceq w'_{p,q}$ for all channels (p, q). This proves the following result:

Corollary 4.3. *Control-state reachability and termination for lossy CFMs are decidable problems.*

However, the complexity of both problems is non-primitive recursive [75]. A precise complexity characterization was obtained in [25] in terms of a hierarchy of recursive functions.

On the negative side, more complex properties, such as repeated reachability of a given global state, are undecidable for lossy CFMs [2]. Extensions of this undecidability result were obtained in [63], by considering lossy counter machines. These are usual counter machines (with zero tests) where the counters can be decremented spontaneously. The main result in the latter paper is that it is undecidable whether a lossy counter machine has an infinite run from some initial configuration:

Theorem 4.4 ([63]). *Structural termination and boundedness for lossy counter machines are undecidable problems.*

Through a simulation of lossy counter machines by lossy CFMs, the above result extends to the latter model (and yields undecidability for two other problems):

Corollary 4.5 ([2, 63]). *The following questions about lossy CFMs are undecidable: structural termination, boundedness, and repeated reachability.*

A different picture arises when the source of faults are insertion errors, as considered, e.g., in [23, 21]. As defined there, insertion CFMs (called insertion channel machines) are CFMs where channels can acquire additional contents spontaneously. In addition, in [21], such machines are endowed with tests for channel emptiness and for non-occurrence of a given content.

Remark 4.6. Notice that for an insertion CFM, its reachability set (defined as in [23]) is recognizable, since it is upward-closed w.r.t. the wqo from the proof of Cor. 4.3. The same holds for lossy CFMs, since the reachability set here is downward-closed and therefore the complement of an upward-closed set. However, an essential difference is that a finite automaton accepting the reachability set of an insertion CFM is effectively computable, whereas this is not the case for lossy CFMs. For the first assertion one can use a similar argument as in the fixpoint computation of the backward reachability algorithm mentioned above [23]. For lossy CFMs the reachability set cannot be effectively computable —

otherwise the boundedness problem for lossy CFMs would be decidable, contradicting Theorem 4.4 (just compute a finite automaton for the reachability set and check it for finiteness).

Together with the wqo from the proof of Cor. 4.3, insertion CFMs are WSTS. From Thm. 4.1, it follows that their control-state reachability problem is decidable as well. As for lossy CFMs, this problem is not primitive recursive. In contrast, insertion CFMs behave better w.r.t. the termination problem:

Theorem 4.7 ([21]). *The (structural) termination problem for insertion CFMs has non-elementary, yet primitive recursive complexity.*

Further reading. The paper [23] also considers channels with duplication errors. Systems mixing lossy channels and error-free ones are considered in [24], providing a (polynomial) characterization of architectures with decidable reachability problem. Adding probabilities to lossy CFMs has been considered in several papers. For instance, non-deterministic CFMs with probabilistic lossiness were shown in [9] to have a decidable reachability problem, and so do some instances of the repeated reachability problem (e.g., the "almost surely" instance). On the other hand, repeated reachability with positive probability remains undecidable, by a reduction from the boundedness problem for lossy CFMs.

Part II
Specifications and Model-Checking

In this part, we first introduce formalisms to describe properties of runs of CFMs from a language-theoretic viewpoint. We start with sequential specifications, and show that model-checking CFMs against such specifications is undecidable. We then introduce message sequence charts, a graphical way of presenting the causal dependencies in a run of a CFM, and logics like monadic second-order logic and propositional dynamic logic that express properties of message sequence charts, i.e., causal properties of runs. In Section 7, we will investigate the model-checking problem for such logics. The results in this part support the "rule of thumb" that the only way to avoid undecidability of the model-checking problem is to use specifications that are compatible with the causal structure.

5 Sequential specifications

We start with some notation first. We fix a communication alphabet Σ and call a word over Σ *valid* if it can potentially be executed from some configuration with empty channels. To make this notion strict, let $\pi^!_{p,q}, \pi^?_{p,q} : \Sigma^* \to \mathrm{Msg}^*$ be the homomorphisms defined by

$\pi^!_{p,q}(p!q(m)) = \pi^?_{p,q}(q?p(m)) = m$ and $\pi^!_{p,q}(a) = \pi^?_{p,q}(b) = \varepsilon$ for a not of type $p!q$ and b not of type $q?p$. So, $\pi^!_{p,q}, \pi^?_{p,q}$ are the projections on sends and receives, respectively, on the channel from p to q. A word $u \in \Sigma^*$ is *valid* if $\pi^?_{p,q}(v)$ is a prefix of $\pi^!_{p,q}(v)$ for any prefix v of u and any channel (p,q). In other words, on any channel the sequence of sent messages is consistent with the received sequence, and a receive never precedes its matching send.

A simple formalism to describe properties of words is a finite automaton. Another one would be the linear-time temporal logic eLTL that extends LTL [61] by referring to matching sends and receives. The syntax of the logic eLTL is:

$$\varphi ::= \text{true} \mid \sigma \mid \mathsf{X}\varphi \mid \mathsf{X}_{\text{msg}}\varphi \mid \varphi \mathsf{U} \varphi \mid \varphi \vee \varphi \mid \neg\varphi\,,$$

where $\sigma \in \Sigma$.

Let $u = a_1 a_2 \cdots a_n$ be a valid word with $a_i \in \Sigma$ and let $1 \leqslant i \leqslant n$. Then the satisfaction relation is defined inductively:

$$\begin{aligned}
u, i \models \sigma \quad &\Leftrightarrow \quad a_i = \sigma \\
u, i \models \mathsf{X}\varphi \quad &\Leftrightarrow \quad u, i+1 \models \varphi \text{ and } i < n \\
u, i \models \mathsf{X}_{\text{msg}}\varphi \quad &\Leftrightarrow \quad a_i = p!q(m) \text{ for some } (p,q) \in \text{Ch}, \text{ and there exists } i < j \leqslant n \text{ with} \\
&\qquad u, j \models \varphi, a_j = q?p(m), \text{ and } |\pi^!_{p,q}(a_1 \cdots a_i)| = |\pi^?_{p,q}(a_1 \cdots a_j)| \\
u, i \models \varphi_1 \mathsf{U} \varphi_2 \quad &\Leftrightarrow \quad \text{there exists } i \leqslant k \leqslant n \text{ with } u, k \models \varphi_2 \\
&\qquad \text{and } u, j \models \varphi_1 \text{ for all } i \leqslant j < k
\end{aligned}$$

($u, i \models \varphi_1 \vee \varphi_2$ and $u, i \models \neg\varphi$ are defined in the obvious way.) The modalities X (next) and U (until) are standard, while X_{msg} refers to the matching receive of the current send event. We write $u \models \varphi$ for $u, 1 \models \varphi$.

Example 5.1. Unsurprisingly, the following example shows that the additional X_{msg} operator can express non-regular properties, even restricted to valid words over Σ. We assume that $\mathcal{P} = \{p, q, r\}$, $\text{Ch} = \{(p,q), (p,r), (r,q)\}$ and that Msg is a singleton, therefore omitted. Process r is actually not needed in this example, but plays a role in Section 6.2.

Consider the eLTL formula $\varphi = \text{true}\, \mathsf{U}\, (\mathsf{X}\, c_p \wedge \mathsf{X}_{\text{msg}}\, \mathsf{X}\, c_q)$ expressing that there is a matching pair of send and receive events that are immediately followed by the local actions c_p and c_q, respectively. Let L be a regular language that contains all valid words satisfying φ. Notice that words of the form

$$u = (p!q)^{x_1}\, c_p\, (p!q)^{x_2}\, (p!r\; r?p\; r!q\; q?r)\, (q?p)^{y_1}\, c_q\, (q?p)^{y_2}$$

with $x_i, y_i \geqslant 0$, are valid if and only if $x_1 + x_2 \geqslant y_1 + y_2$. A valid word u as above belongs to L if and only if $x_1 = y_1 > 0$, in this case the matching send/receive pair is on channel (p,q). Provided y_1 and y_2 are sufficiently large, a pumping argument allows us to find $h < y_1$ such that

$$v = (p!q)^{x_1}\, c_p\, (p!q)^{x_2}\, (p!r\; r?p\; r!q\; q?r)\, (q?p)^{y_1 - h}\, c_q\, (q?p)^{h + y_2}$$

also belongs to L. But the valid word v does not satisfy φ. Hence, any regular language that contains all valid words satisfying φ also contains some valid word not satisfying φ.

Unsurprisingly, model-checking problem CFMs against sequential specifications is undecidable:

Proposition 5.1. *Given a CFM $\mathcal{A}$ and an LTL-formula φ (i.e., an eLTL-formula not using the modality $\mathsf{X}_{\mathrm{msg}}$), it is undecidable whether all words $u \in \mathcal{L}(\mathcal{A})$ satisfy φ.*

An obvious reduction from Post's correspondence problem yields a fixed CFM with an undecidable model-checking problem: Let $\mathcal{P} = \{p, q, r, s\}$, $\mathrm{Ch} = \{(p,q),(r,s)\}$, and $\mathrm{Msg} = \{a, b, 1, 2, 3, 4, 5, \$\}$. The CFM $\mathcal{A}$ we want to model-check is the "universal" CFM on this architecture, and its language is the set of all valid words. Now let $\mathcal{I} = (u_i, v_i)_{1 \leqslant i \leqslant 5}$ be five pairs of words over the alphabet $A = \{a, b\}$, i.e., an instance of Post's correspondence problem. Then the language

$$[\bigcup_{1\leqslant i\leqslant 5} p!q(i) r!s(u_i)]^+ \, p!q(\$)\, r!s(\$)\; [\bigcup_{1\leqslant i\leqslant 5} q?p(i) s?r(v_i)]^+ \, q?p(\$)\, s?r(\$)$$

can be expressed by an LTL formula φ (here, $r!s(a_1 a_2 \cdots a_n)$ means the sequence of sends $r!s(a_1) \cdots r!s(a_n)$, and $s?r(a_1 \cdots a_n)$ is to be understood similarly).

Suppose u belongs to the language of φ and is valid. Then $\pi^{!}_{p,q}(u) = \pi^{?}_{p,q}(u) = i_1 i_2 \cdots i_n \$$ implies that $i_1, \ldots, i_n$ is a solution for $\mathcal{I}$. Conversely, any solution to $\mathcal{I}$ gives a valid word satisfying φ. Hence, all words from the language of $\mathcal{A}$ satisfy $\neg\varphi$ if and only if $\mathcal{I}$ has no solution. But this is undecidable [69].

6 Partial order specifications

A CFM is meant to model a network of asynchronously evolving processes. In particular, distinct processes can act at the same time. For instance, the processes p and p' can, at the very same moment, send messages m and m' to processes q and q'.

Recall that from a CFM, we defined a labeled infinite transition system such that executions of the CFM are considered as paths in this transition system. This forces us to linearly order the two actions $p!q(m)$ and $p'!q'(m')$ from above, giving rise to two different paths in the transition system.

Although simple, the interleaving semantics is unsatisfactory for at least two reasons: one is the undecidability of the model-checking problem, and the other is that many causal properties, e.g., races, are hard to formulate in the interleaving semantics. In this section, we model computations of CFMs by suitable partial orders, called message sequence charts [2]. Then we propose two logical formalisms that can describe causal properties of such objects.

6.1 Message sequence charts

Message sequence charts are Σ-labeled posets $\langle E, \leqslant, \lambda \rangle$, with E a set of events, $\leqslant$ a partial order on E, and $\lambda : E \to \Sigma$ a labeling function. We write $P(e)$ for the process

[2]The MSC partial order may be better known as Lamport's *happens-before* relation [56].

on which an event e is located. That is, we let $P(e) = p$ if $\lambda(e) \in \Sigma_p$, call e a p-event, and let $E_p = P^{-1}(p)$ be the set of all p-events. An event labeled by some $p!q(m)$ (resp., $q?p(m)$) is called an event of *type* $p!q$ (resp., $q?p$).

We need to define two relations $\leqslant_P$ and $<_{\mathrm{msg}}$ on events:

- $e \leqslant_P f$ if $P(e) = P(f)$ and $e \leqslant f$.
- $e <_{\mathrm{msg}} f$ if for some $(p, q) \in \mathrm{Ch}$, the two following conditions hold:
  (1) e is an event of type $p!q$ and f of type $q?p$.
  (2) The number of events $e' \leqslant e$ of type $p!q$ equals the number of events $f' \leqslant f$ of type $q?p$.

The meaning of the two relations $\leqslant_P$ and $<_{\mathrm{msg}}$ is as expected: the relation $\leqslant_P$ describes the order of the events executed by each process, whereas $<_{\mathrm{msg}}$ describes matching pairs of send and receive events (under the assumption that channels are FIFO). By $\lessdot_P$ we denote the immediate process successor relation: $e \lessdot_P f$ if $e <_P f$ and $e \leqslant_P g <_P f$ implies $e = g$.

Definition 6.1. A *prefix message sequence chart* $M = \langle E, \leqslant, \lambda \rangle$ is a finite, Σ-labeled poset – up to isomorphism – satisfying the conditions below:

(1) $\leqslant = (\leqslant_P \cup <_{\mathrm{msg}})^*$.
(2) The set of p-events $E_p \subseteq E$ is linearly ordered for any process $p \in \mathcal{P}$.
(3) For any event f of type $q?p$, there exists an event e and a message content $m \in \mathrm{Msg}$ such that $e <_{\mathrm{msg}} f$, $\lambda(e) = p!q(m)$, and $\lambda(f) = q?p(m)$.

A *message sequence chart* (MSC for short) is a prefix message sequence chart satisfying, in addition,

(4) for any send event e there exists a receive event f such that $e <_{\mathrm{msg}} f$.

The relation $<_{\mathrm{msg}}$ matches sends and receives. By the third requirement, matching send- and receive-events handle the same message content. It can be understood as saying "any message received has been sent". Prefix MSCs do not necessarily satisfy the dual condition "any message sent will be received" – this additional requirement is condition (4) that defines MSCs.

Example 6.1. An example prefix MSC is shown in the left half of Figure 3; this prefix MSC is an execution of the CFM of Example 2.1, page 1033. All events of process **C** are drawn as circles on the *process line* **C** (and similarly for **S**). For simplicity, not all event labels are shown in the picture. Vertical edges denote the relation $\lessdot_P$ (oriented downwards), and diagonal arrows denote the relation $<_{\mathrm{msg}}$; the label of such an arrow denotes the message contents transmitted (e.g., the action performed by the source event of the first diagonal edge is $\mathbf{C}!\,\mathbf{S}(0)$). The partial order is therefore the reflexive and transitive closure of all the arrows. The last two events of type $\mathbf{C}!\,\mathbf{S}$ are unmatched, since the messages sent are not received.

A *linearization* of a Σ-labeled partial order $\langle E, \leqslant, \lambda \rangle$ is the sequence of labels of a total order that extends $\leqslant$. Thus, a linearization is a word over the alphabet Σ, and the set $\mathrm{Lin}(M)$ of linearizations of a prefix MSC M is a subset of Σ^*. For a set (or language) of partial orders X, we write $\mathrm{Lin}(X) = \bigcup_{M \in X} \mathrm{Lin}(M)$.

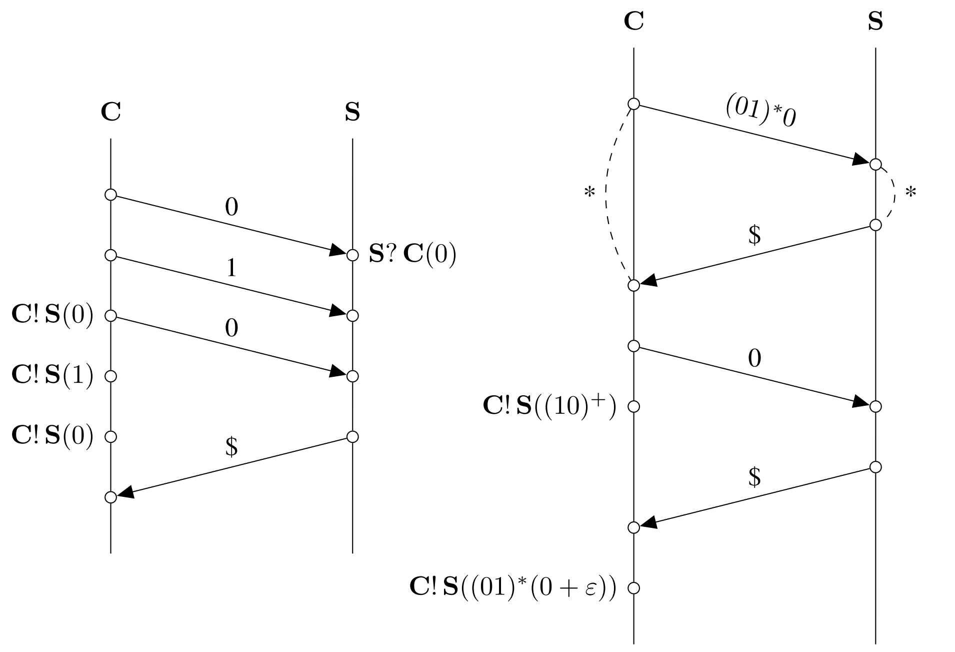

Figure 3. A prefix message sequence chart (left) and a symbolic representation of the set of executions of the CFM in Example 2.1 (right).

The relation between the MSC partial order and its linearizations is very tight, due to the FIFO channels:

Lemma 6.1. *Any linearization of a prefix MSC is a valid word, and any valid word u is the linearization of some unique prefix MSC that we denote* $\mathrm{msc}(u)$.

Let $\mathcal{A} = \langle (\mathcal{A}_p)_{p\in\mathcal{P}}, \Sigma, F\rangle$ be a CFM and let $\mathcal{M}_{\mathrm{prf}}(\mathcal{A})$ denote the set of prefix MSCs that admit at least one linearization in $\mathcal{L}(\mathcal{A})$, i.e.,

$$\mathcal{M}_{\mathrm{prf}}(\mathcal{A}) = \{\mathrm{msc}(u) \mid u \in \mathcal{L}(\mathcal{A})\}\,.$$

Since a CFM cannot distinguish between different linearizations of an MSC, note that $\mathcal{M}_{\mathrm{prf}}(\mathcal{A})$ equals the set of prefix MSCs M such that *all* their linearizations are in $\mathcal{L}(\mathcal{A})$.

Next we describe a more direct characterization of the set $\mathcal{M}_{\mathrm{prf}}(\mathcal{A})$ of prefix MSCs associated with accepting runs of the CFM $\mathcal{A} = \langle (\mathcal{A}_p)_{p\in\mathcal{P}}, \Sigma, F\rangle$. Let $M = \langle E, \leqslant, \lambda\rangle$ be a prefix MSC. A *run of $\mathcal{A}$ on M* is a mapping $\rho : E \to \bigcup_{p\in\mathcal{P}} S_p$ labeling events of M by local states, such that for any $p \in \mathcal{P}$ and any $e, f \in E_p$ we have

- if $e <_P f$, then $\rho(e) \xrightarrow{\lambda(f)}_p \rho(f)$ and
- if $f = \min(E_p, \leqslant_P)$, then $s_p^0 \xrightarrow{\lambda(f)}_p \rho(f)$.

For $p \in \mathcal{P}$, let $s_p = \rho(\max(E_p, \leqslant_P))$ if $E_p \neq \emptyset$ and the initial state s_p^0 otherwise. Then the run ρ is *accepting* if the global state $(s_p)_{p\in\mathcal{P}}$ belongs to the set of accepting states F.

Now one can show that $M \in \mathcal{M}_{\text{prf}}(\mathcal{A})$ (i.e., $M = \text{msc}(u)$ for some $u \in \mathcal{L}(\mathcal{A})$) if and only if $\mathcal{A}$ admits some accepting run on M.

Example 6.2. The right part of Figure 3 is another abstract view of the executions of the CFM in Example 2.1, this time in a more succinct way using prefix MSCs. The message labeled $(01)^*0$ stands for any sequence of message arrows from **C** to **S** with contents $0, 1, \ldots, 0$. The event **C**! **S**$((10)^*)$ stands for a sequence of sends **C**! **S** with contents $1, 0, \ldots, 1, 0$, similarly for **C**! **S**$((01)^*(0 + \varepsilon))$. The upper half of this picture can be iterated. Recall that process **C** and **S** alternate between states c_0, c_1, and s_0, s_1 respectively. With this picture it is rather easy to check that the set of reachable configurations with state (c_0, s_0) has channel contents in one of the following sets (cf. also Figure 2):

- $((01)^*, \varepsilon)$: **C** and **S** both in upper half, either **S** before sending \$, or both at the end of upper half,
- $((10)^*1, \$)$: **C** and **S** both in lower half, after **S** sends \$ (and before **C** receives it),
- $((10)^+(01)^*, \varepsilon)$: **C** and **S** both in lower half, after **C** receives \$.

Because of the close correspondence between the sets $\mathcal{M}_{\text{prf}}(\mathcal{A})$ and $\mathcal{L}(\mathcal{A})$, and since the elements of the former make the matching relation visible, it is convenient to consider $\mathcal{M}_{\text{prf}}(\mathcal{A})$ as semantics of the CFM $\mathcal{A}$. The literature looking at CFMs from this point of view usually considers only prefix MSCs where all messages are received. In other words, the semantics of CFMs is usually defined as set of MSCs, i.e., prefix MSCs where all send events are matched by some receive event:

$$\mathcal{M}(\mathcal{A}) = \{M \in \mathcal{M}_{\text{prf}}(\mathcal{A}) \mid M \text{ is an MSC}\}\,.$$

In the sense of Lemma 6.1, MSCs from $\mathcal{M}_{\text{prf}}(\mathcal{A})$ correspond to valid words from $\mathcal{L}(\mathcal{A})$ whose execution in $\mathcal{A}$ ends with empty buffers. We call a word over Σ *complete* if, for any channel $(p, q) \in \text{Ch}$, the number of occurrences of events of type $p!q$ equals that of type $q?p$. Then Lemma 6.1 remains correct if we replace “prefix MSC” by “MSC” and “valid word” by “valid and complete word”. Let $\mathcal{L}_{\text{cpl}}(\mathcal{A})$ denote the set of words from $\mathcal{L}(\mathcal{A})$ that are complete.

Then the semantics $\mathcal{M}(\mathcal{A})$ of the CFM $\mathcal{A}$ satisfies

$$\mathcal{M}(\mathcal{A}) = \{\text{msc}(u) \mid u \in \mathcal{L}_{\text{cpl}}(\mathcal{A})\}\,.$$

6.2 Monadic second-order logic

We fix a set Var of *individual variables* and a set VAR of *set variables*. We will use the convention that lower-case variables belong to Var and upper-case variables to VAR. The set MSO of *monadic second-order* (or MSO) formulas is given by the following grammar:

$$\begin{aligned}\varphi ::= \sigma(x) \mid x \lessdot_P y \mid x <_{\text{msg}} y \mid x \leqslant y \mid x = y \mid x \in X \mid \\ \exists x\, \varphi \mid \exists X \varphi \mid \neg\varphi \mid \varphi \vee \varphi\,,\end{aligned}$$

where $\sigma \in \Sigma$ is an action, x, y are individual variables from Var, and $X \in \text{VAR}$ is a set variable. A formula is *first-order* if it does not contain any subformula $\exists X\, \psi$ for

$X \in \mathrm{VAR}$, and it is *existential* if it is of the form $\exists X_1 \exists X_2 \cdots \exists X_n\, \varphi$ where φ is first-order. We write FO and EMSO for the sets of first-order and existential MSO formulas.

Let $M = \langle E, \leqslant, \lambda \rangle$ be an MSC, $f\colon \mathrm{Var} \to E$ and $g\colon \mathrm{VAR} \to 2^E$ be interpretations of variables, and φ a formula from MSO. We then define $M, f, g \models \varphi$ (read "the MSC M satisfies the formula φ under the interpretations of variables f and g") by induction:

$$\begin{aligned}
&M, f, g \models \sigma(x) \Leftrightarrow \lambda(f(x)) = \sigma\,,\\
&M, f, g \models x \lessdot_P y \Leftrightarrow f(x) \lessdot_P f(y)\,,\\
&M, f, g \models x \in X \Leftrightarrow f(x) \in g(X)\,,\\
&M, f, g \models \exists x\, \varphi \Leftrightarrow \text{ there exists } v \in E \text{ with } M, f[x/v], g \models \varphi\,,
\end{aligned}$$

and similarly for the formulas $x <_{\mathrm{msg}} y$, $x \leqslant y$, $x = y$, and $\exists X\, \varphi$. Here, $f[x/v]$ denotes the function $\mathrm{Var} \to E$ that sends x to v and agrees with f for all other variables. The semantics of $\neg\varphi$ and $\varphi_1 \vee \varphi_2$ are standard. For a subset R of $\{\lessdot_P, <_{\mathrm{msg}}, \leqslant\}$, let $\mathrm{MSO}(R)$ be the set of those MSO-formulas that do not mention relations from $\{\lessdot_P, <_{\mathrm{msg}}, \leqslant\} \setminus R$.

As a first example, consider the following formula

$$\mathrm{Succ} = x < y \wedge \bigvee_{p \in \mathcal{P}} \big(\Sigma_p(x) \wedge \Sigma_p(y) \wedge \forall z (\Sigma_p(z) \to z \leqslant x \vee y \leqslant z)\big)$$

from $\mathrm{MSO}(\leqslant)$ with two free individual variables x and y, where $\Sigma_p(x)$ is an abbreviation of $\bigvee_{\sigma \in \Sigma_p} \sigma(x)$. Then we have $M, f, g \models \mathrm{Succ}$ if and only if $f(x) \lessdot_P f(y)$. Hence any formula from $\mathrm{MSO}(\{\lessdot_P\} \cup R)$ can be translated into an equivalent formula from $\mathrm{MSO}(\{\leqslant\} \cup R)$, independently of what the set R is. Note that the same holds if we only consider first order or existential formulas. But such an argument cannot work for $<_{\mathrm{msg}}$; more precisely, there is a formula from $\mathrm{MSO}(\lessdot_P, <_{\mathrm{msg}})$ (and therefore from $\mathrm{MSO}(\leqslant, <_{\mathrm{msg}})$) without equivalent counterpart in $\mathrm{MSO}(\leqslant)$. Such an example is the eLTL-formula from Example 5.1, which we rephrase in $\mathrm{MSO}(\lessdot_P, <_{\mathrm{msg}})$ as

$$\varphi = \exists x, x', y, y' : x \lessdot_P x' \wedge y \lessdot_P y' \wedge x <_{\mathrm{msg}} y \wedge c_p(x') \wedge c_q(y')\,.$$

Suppose that ψ is an equivalent formula from $\mathrm{MSO}(\leqslant)$, and let u be a valid and complete word as in Example 5.1. Then $\mathrm{msc}(u)$ is linearly ordered by $\leqslant$, so we can interpret ψ directly on the word u. Let L be the *regular* set of all words over Σ that satisfy ψ. Then L contains, in particular, all valid and complete words u such that $\mathrm{msc}(u)$ is linearly ordered and satisfies φ. Arguments as in Example 5.1 show that L also contains some valid and complete word v with $\mathrm{msc}(v)$ linear and $\mathrm{msc}(v) \not\models \varphi$. But $v \in L$ and $\mathrm{msc}(v)$ linear imply $\mathrm{msc}(v) \models \psi$, so ψ and φ cannot be equivalent.

Next consider the following formula $\mathrm{Leq}(x, y)$ from $\mathrm{MSO}(\lessdot_P, <_{\mathrm{msg}})$:

$$\exists X\colon \big(y \in X \wedge \forall z' (z' \in X \leftrightarrow (z' = x \vee \exists z \in X : z <_{\mathrm{msg}} z' \vee z \lessdot_P z'))\big)\,.$$

The formula $\forall z'(\ldots)$ above holds in an MSC $M = \langle E, \leqslant, \lambda \rangle$ (with respect to f and g) if and only if $g(X)$ is the set of nodes $e \in E$ where $f(x) \leqslant e$. Hence $\mathrm{Leq}(x, y)$ holds if and only if $f(x) \leqslant f(y)$. As a consequence, any formula φ from MSO can be translated into an equivalent formula $\overline{\varphi}$ from $\mathrm{MSO}(\lessdot_P, <_{\mathrm{msg}})$. Since the above formula uses an additional set variable, $\overline{\varphi}$ is in general not first order or existential, even if φ is first order or existential, respectively.

6.3 Propositional dynamic logic

In this section, we introduce a temporal logic to describe causal properties. This logic is adapted from classical propositional dynamic logic (PDL for short) [38]. Like the logic TLC [7], which is interpreted over Mazurkiewicz traces (cf. Section 9.1), our variant of PDL is interpreted directly on MSCs, and extends the logic TLC$^-$ considered in [72].

The logic PDL has three types of formulas: local formulas express properties of single events in an MSC, path expressions express properties of sequences of events, and global formulas express properties of MSCs as a whole. As example, the local formula $p!q(m)$ is true if and only if the current event sends the message m from p to q. The path expression $(\mathrm{msg};\mathrm{proc})^*$ holds true of a sequence of events $e_0, e_1, e_2, e_3, \ldots, e_{2n}$ if and only if $e_{2i} <_{\mathrm{msg}} e_{2i+1} \lessdot_P e_{2i+2}$ for all $0 \leqslant i < n$. That is, $e_0 < e_{2n}$ via a path that alternates between message arcs and process successor arcs. Then the local formula $\langle(\mathrm{msg};\mathrm{proc})^*\rangle\, p!q(m)$ holds if, from the current event, there is a path as described, leading to an event that sends message content m from p to q. Finally, the global formula $\mathsf{E}(q!p(n) \wedge \langle(\mathrm{msg};\mathrm{proc})^*\rangle\, p!q(m))$ holds if and only if the MSC contains some event labeled $q!p(n)$ where the previous formula holds.

Path expressions π and *local formulas* α are defined inductively (with $\sigma \in \Sigma$ below):

$$\pi ::= \mathrm{proc} \mid \mathrm{proc}^{-1} \mid \mathrm{msg} \mid \mathrm{msg}^{-1} \mid \{\alpha\} \mid \pi;\pi \mid \pi + \pi \mid \pi^*\,,$$
$$\alpha ::= \mathrm{true} \mid \sigma \mid \alpha \vee \alpha \mid \neg\alpha \mid \langle\pi\rangle\,\alpha\,.$$

Local formulas express properties of single events in MSCs. To define the semantics of local formulas, let $M = \langle E, \leqslant, \lambda\rangle$ be an MSC and $e \in E$ an event from M with $P(e) = p$. Then $M, e \models \sigma$ if and only if $\lambda(e) = \sigma$; $M, e \models \alpha_1 \vee \alpha_2$ and $M, e \models \neg\alpha$ are defined canonically.

The semantics of modalities $\langle\pi\rangle\,\alpha$ indexed by a path expression π is to reach via a program (path) π an event where the local formula α is satisfied:

$$\begin{aligned}
&M, e \models \langle\mathrm{proc}\rangle\,\alpha \Leftrightarrow \text{there exists } e' \in E \text{ with } e \lessdot_P e' \text{ and } M, e' \models \alpha\,,\\
&M, e \models \langle\mathrm{proc}^{-1}\rangle\,\alpha \Leftrightarrow \text{there exists } e' \in E \text{ with } e' \lessdot_P e \text{ and } M, e' \models \alpha\,,\\
&M, e \models \langle\mathrm{msg}\rangle\,\alpha \Leftrightarrow \text{there exists } e' \in E \text{ with } e <_{\mathrm{msg}} e' \text{ and } M, e' \models \alpha\,,\\
&M, e \models \langle\mathrm{msg}^{-1}\rangle\,\alpha \Leftrightarrow \text{there exists } e' \in E \text{ with } e' <_{\mathrm{msg}} e \text{ and } M, e' \models \alpha\,,\\
&M, e \models \langle\{\alpha\}\rangle\,\beta \Leftrightarrow M, e \models \alpha \text{ and } M, e \models \beta\,,\\
&M, e \models \langle\pi_1;\pi_2\rangle\,\alpha \Leftrightarrow M, e \models \langle\pi_1\rangle\,\langle\pi_2\rangle\,\alpha\,,\\
&M, e \models \langle\pi_1 + \pi_2\rangle\,\alpha \Leftrightarrow M, e \models \langle\pi_1\rangle\,\alpha \vee \langle\pi_2\rangle\,\alpha\,,\\
&M, e \models \langle\pi^*\rangle\,\alpha \Leftrightarrow \text{there exists } n \geqslant 0 \text{ with } M, e \models (\langle\pi\rangle)^n\alpha\,.
\end{aligned}$$

Global formulas are Boolean combinations of properties of the form “there exists an event satisfying the local formula α”. Their syntax is given by:

$$\varphi ::= \quad \mathsf{E}\alpha \mid \mathsf{A}\alpha \mid \varphi \vee \varphi \mid \varphi \wedge \varphi\,,$$

where α ranges over the set of local formulas. The semantics is defined by:

$$\begin{aligned}
&M \models \mathsf{E}\alpha \Leftrightarrow \text{ there exists an event } e \text{ with } M, e \models \alpha\,,\\
&M \models \mathsf{A}\alpha \Leftrightarrow M, e \models \alpha \text{ for all events } e\,.
\end{aligned}$$

The definitions of $M \models \varphi_1 \vee \varphi_2$ and $M \models \varphi_1 \wedge \varphi_2$ are obvious.

Semantically, a local formula of the form $\langle(\{\alpha\};(\mathrm{proc}+\mathrm{msg}))^*\rangle\beta$ corresponds to the until construct $\alpha\,\mathcal{U}\beta$ in the temporal logic TLC$^-$ [72], i.e., TLC$^-$ is a semantic fragment of PDL. In TLC$^-$, however, one cannot express properties such as "there is an odd number of messages from p to q", which is expressible in PDL by the global formula below.

Example 6.3. We use the usual abbreviations for Boolean and rational expressions, as well as $p!q$ for $\bigvee_{m\in\mathrm{Msg}} p!q(m)$. The local formula $p!q \wedge \neg\left\langle(\mathrm{proc}^{-1})^+\right\rangle p!q$ is satisfied by the first event on process p of type $p!q$. Paths on process p with an even number of events of type $p!q$ (ending with such an event, and not counting the first event) are described by the path formula $[((\mathrm{proc}\,;\{\neg p!q\})^*;\mathrm{proc}\,;\{p!q\})^2]^*$. The global formula

$$\mathsf{A}\left((p!q \wedge \neg\left\langle(\mathrm{proc}^{-1})^+\right\rangle p!q) \to \left\langle[((\mathrm{proc}\,;\{\neg p!q\})^*;\mathrm{proc}\,;\{p!q\})^2]^*\right\rangle \neg\left\langle\mathrm{proc}^+\right\rangle p!q\right)$$

says that the number of messages from p to q is odd.

7 Model-checking

Since reachability is already undecidable for asynchronously communicating processes, model-checking any non-trivial property requires some restrictions. Given that the main reason for undecidability are the unbounded FIFO channels, we focus in this section on two variants of channel bounds. Notice that channel bounds are reasonably justified, since any concrete implementation of a communication protocol needs to use channels of some given size. However, we can relax channel bounds, from a universal to an existential version. The universal version is pessimistic and considers only those executions that, independently of the scheduling of events, use channels of a given bound. The existential version of channel bounds is optimistic and considers only those executions that can be *rescheduled* in order to use channels of a given bound. Although these two variants look rather similar, it is important to stress that certain communication networks guarantee that *every* CFM is existentially bounded (cf. Example 7.3), which is not the case for universal bounds.

7.1 Channel bounds

Let B be a fixed positive integer (we usually refer to B as the *bound*). A valid word u is *B-bounded* if every prefix v of u contains at most B events of type $p!q$ that are unmatched in v:

$$|\pi^!_{p,q}(v)| - |\pi^?_{p,q}(v)| \leqslant B\,.$$

We write $\Sigma^*|_B$ for the set of valid B-bounded words. It is not hard to see that this set is regular.

We explain now what "rescheduling a run" means. Let $\sim$ be the least equivalence relation on the set of *valid* words such that $u \sim v$ whenever $u = xaby$ and $v = xbay$ with $x, y \in \Sigma^*$, $a \in \Sigma_p$ and $b \in \Sigma_q$ for two distinct processes p and q. Note that, since u, v

are both valid, actions a, b cannot be a matching send/receive pair. We say that u can be rescheduled into v if $u \sim v$. For a valid word u, let $[u]_\sim$ denote its equivalence class with respect to $\sim$, i.e., the set of valid words that can be rescheduled to u.

A valid word u is *universally B-bounded* if $[u]_\sim \subseteq \Sigma^*|_B$ and it is *existentially B-bounded* if $[u]_\sim \cap \Sigma^*|_B \neq \emptyset$.

Example 7.1. The word $u = \mathbf{C}!\,\mathbf{S}(0)\ \mathbf{C}!\,\mathbf{S}(1)\ \mathbf{S}?\,\mathbf{C}(0)\ \mathbf{S}?\,\mathbf{C}(1)\ \mathbf{S}!\,\mathbf{C}(1)\ \mathbf{C}?\,\mathbf{S}(1)$ is valid, complete, and 2-bounded, but not 1-bounded because of the prefix $\mathbf{C}!\,\mathbf{S}(0)\ \mathbf{C}!\,\mathbf{S}(1)$ that consists of sends, only. But we have

$$\begin{aligned} u &= \mathbf{C}!\,\mathbf{S}(0)\ \mathbf{C}!\,\mathbf{S}(1)\ \mathbf{S}?\,\mathbf{C}(0)\ \mathbf{S}?\,\mathbf{C}(1)\ \mathbf{S}!\,\mathbf{C}(1)\ \mathbf{C}?\,\mathbf{S}(1) \\ &\sim \mathbf{C}!\,\mathbf{S}(0)\ \mathbf{S}?\,\mathbf{C}(0)\ \mathbf{C}!\,\mathbf{S}(1)\ \mathbf{S}?\,\mathbf{C}(1)\ \mathbf{S}!\,\mathbf{C}(1)\ \mathbf{C}?\,\mathbf{S}(1) \end{aligned}$$

and this word is 1-bounded. Consequently, u is existentially 1-bounded. Since u is not 1-bounded, it cannot be universally 1-bounded, but the reader is invited to check that u is universally 2-bounded.

7.2 Bounded CFMs

As a preparation for model-checking, we first consider the control-state reachability problem relative to channel bounds:

- *Universally bounded control-state reachability problem*: Given a CFM $\mathcal{A}$, a bound B, and a global state $s \in S$, is there some universally B-bounded valid word that leads from the initial configuration to some configuration with global state s?
- The *existentially bounded control-state reachability problem* is similar, one requires the valid word to be existentially B-bounded.

To solve the second of these problems, we simply restrict the transition system associated with the CFM $\mathcal{A}$ to those configurations $\langle s, w\rangle$ with $|w_{p,q}| \leqslant B$ for all $(p,q) \in \mathrm{Ch}$ (i.e., to configurations that hold at most B messages in each channel). The result is a finite automaton and in this automaton, some configuration with global state s can be reached if and only if the existentially bounded control-state reachability problem has an affirmative answer. Since the size of the finite automaton is exponential in B and $|\mathcal{P}|$, this is a PSPACE-algorithm (with B given in unary).

The universally bounded control-state reachability problem is slightly more involved. Here, one first observes that the set of universally B-bounded valid words is regular. Hence, it suffices to take the finite automaton from the previous paragraph, intersect its language with the language of universally B-bounded valid words, and check this intersection for emptiness. Thus, we have:

Proposition 7.1. *The universally and the existentially bounded control-state reachability problem are* PSPACE*-complete (with B given in unary).*

PSPACE-hardness is easily shown by reducing from the halting problem of linearly space bounded Turing machines: to simulate such a machine, one associates a process with each tape cell. At each instant, exactly one process is active, corresponding to the position of the head. This process can take over activity to one of its neighbors, thereby simulating the moves of the head. This CFM can be defined in such a way that its language consists of universally 1-bounded words, only.

Definition 7.1. A CFM $\mathcal{A}$ is *universally B-bounded* if all words from its language $\mathcal{L}_{\mathrm{cpl}}(\mathcal{A})$ are universally B-bounded: $\mathcal{L}_{\mathrm{cpl}}(\mathcal{A}) \subseteq \Sigma^*|_B$. Similarly, a CFM $\mathcal{A}$ is *existentially B-bounded* if all words from its language $\mathcal{L}_{\mathrm{cpl}}(\mathcal{A})$ are existentially B-bounded. We call $\mathcal{A}$ *universally* or *existentially bounded* if it is universally or existentially B-bounded for some bound B.

Example 7.2. The CFM in Example 2.1 (page 1033) is not universally bounded; however, it is existentially 1-bounded. For instance, $(\mathbf{C}!\,\mathbf{S}(0)\;\;\mathbf{C}!\,\mathbf{S}(1))^k(\mathbf{S}?\,\mathbf{C}(0)\;\;\mathbf{S}?\,\mathbf{C}(1))^k$ is not $2k-1$-bounded (and hence not universally $2k-1$-bounded), but can be rescheduled into the 1-bounded word $(\mathbf{C}!\,\mathbf{S}(0)\;\;\mathbf{S}?\,\mathbf{C}(0)\;\;\mathbf{C}!\,\mathbf{S}(1)\;\;\mathbf{S}?\,\mathbf{C}(1))^k$.

Example 7.3. Assuming that the undirected graph underlying the communication network is a forest, every CFM over such a network is existentially 1-bounded [50], but not universally bounded, in general. Actually, a CFM may have universal channel bounds only if the network has suitable cycles. The precise characterization is related to the notion of *loop-connected* MSC-graphs [67].

Remark 7.2. The above definition of channel bounds for CFMs is based on accepting runs that end in a configuration with all channels empty. Alternatively, we can define universal/existential bounds over all possible runs and/or by allowing arbitrary channel contents in final configurations. This makes only a difference at the level of deciding such bounds, but not in the expressivity results presented later.

The next result shows that we can decide whether a deadlock-free CFM is existentially (or universally) bounded for a given bound B. Without the assumption of deadlock-freeness, or if the bound is unknown, the problem remains undecidable.

Proposition 7.3 ([44]). *(1) Given a bound B and a deadlock-free CFM $\mathcal{A}$, it is decidable whether $\mathcal{A}$ is existentially (resp., universally) B-bounded. This problem is* PSPACE*-complete if the bound B is given in unary.*

(2) The following problems are undecidable:
- *Is a deterministic CFM existentially (resp., universally) B-bounded (for a given bound B)?*
- *Is a deterministic and deadlock-free CFM existentially (resp., universally) bounded?*

7.3 Model-checking with channel bounds

With universally bounded channels a decision procedure for the model-checking problem of CFMs w.r.t., say, regular specifications is straightforward, since such a machine is equivalent to a (exponentially larger) finite automaton. We can make this statement more precise, by observing that many natural decision problems about universally B-bounded CFMs are PSPACE-complete. This observation could be declared as another “rule of thumb" that holds as well for models like 1-safe Petri nets [35]. As already mentioned, PSPACE-hardness follows from the simulation of linearly bounded automata by a universally 1-bounded CFM. For the upper bound, let us consider model-checking a universally B-bounded CFM $\mathcal{A}$ against an eLTL property; see Section 5. The idea is to consider words over the extended alphabet $\Sigma \times \{0, \dots, B-1\}$ whose projection to Σ^* belongs to the language $\mathcal{L}_{\mathrm{cpl}}(\mathcal{A})$ of $\mathcal{A}$ and whose second component indexes actions of any given type modulo B. Then, in the eLTL formula φ, replace every occurrence of $a \in \Sigma$ by $\bigvee_{i \in \{0,\dots,B-1\}} (a, i)$ and $\mathsf{X}_{\mathrm{msg}}\psi$ by

$$\bigvee_{\substack{0 \leqslant i < B \\ (p,q) \in \mathrm{Ch}}} (p!q, i) \wedge (\neg(q?p, i)) \mathsf{U}((q?p, i) \wedge \psi)\,.$$

Then a valid B-bounded word satisfies φ if and only if its "indexed version" satisfies this new formula. Hence we reduced model-checking of universally bounded CFMs against eLTL-formulas to classical model-checking of LTL (without the operator $\mathsf{X}_{\mathrm{msg}}$).

Proposition 7.4. *Given a bound B in unary, a universally B-bounded CFM $\mathcal{A}$ and an eLTL-formula φ, it is* PSPACE*-complete to decide whether all words $u \in \mathcal{L}(\mathcal{A})$ satisfy φ.*

Similarly, given B in unary, a universally B-bounded CFM $\mathcal{A}$ and a CTL-formula φ, it is PSPACE*-complete to decide whether the global initial state of $\mathcal{A}$ satisfies φ.*

The upper bound for CTL can be shown by simulating universally B-bounded CFMs by 1-safe Petri nets and applying [35]. The simulation uses a place for each local state of $\mathcal{A}$, plus $|\,\mathrm{Ch}\,| \cdot |\,\mathrm{Msg}\,| \cdot B$ places, one for each tuple from $\mathrm{Ch} \times \mathrm{Msg} \times \{0, \dots, B-1\}$. The set of transitions of the Petri net equals the set of local transitions of the CFM $\mathcal{A}$, and each transition of $\mathcal{A}$ is simulated as expected.

We turn now to the model-checking problem for existentially B-bounded CFMs. As already shown in Section 5, model-checking CFMs against sequential specifications such as LTL, is undecidable. From the proof of Proposition 5.1, we can see that undecidability already applies to existentially 1-bounded CFMs. We can also observe that no *regular* set of words describes the same set of MSCs as the eLTL property from Proposition 5.1. This is a key observation that will make model-checking existentially bounded CFMs possible using representatives:

Definition 7.2. Let $K \subseteq \Sigma^*$. A language $R \subseteq \Sigma^*|_B$ is a *set of B-representatives for K* if

$$\{\mathrm{msc}(w) \mid w \in K \text{ valid}\} = \{\mathrm{msc}(v) \mid v \in R\}\,.$$

The next proposition describes a class of regular properties for which model-checking existentially B-bounded CFMs is decidable:

Proposition 7.5. *Given bounds B, B', an existentially B-bounded CFM $\mathcal{A}$ and a regular set R of B'-representatives for a property $K \subseteq \Sigma^*$, it is decidable whether $\mathcal{L}_{\mathrm{cpl}}(\mathcal{A}) \cap K \neq \emptyset$. The complexity is* PSPACE *if R is described by an automaton, and B, B' are in a unary encoding.*

For the proof of the proposition above, we note first that an existentially B-bounded CFM is also existentially B'-bounded for $B' \geqslant B$. So we can assume that $B' \leqslant B$. Let $\mathcal{B}$ be a finite automaton obtained from $\mathcal{A}$ and B' that accepts the language $L = \mathcal{L}_{\mathrm{cpl}}(\mathcal{A}) \cap \Sigma^*|_{B'}$. Notice that for every $u, v \in \Sigma^*|_{B'}$ with $\mathrm{msc}(u) = \mathrm{msc}(v)$, we have $u \in L$ if and only if $v \in L$. Therefore, $\mathcal{L}_{\mathrm{cpl}}(\mathcal{A}) \cap K \neq \emptyset$ is equivalent to $L \cap R \neq \emptyset$. Since $\mathcal{B}$'s size is exponential in $\mathcal{A}$ and B', the claim follows.

A prime example of properties satisfying the assumption of Proposition 7.5 is the class of partial order properties, as shown by the two results below:

Proposition 7.6 ([60]). *Given a bound B and a sentence φ from* MSO*, the language*

$$\{w \in \Sigma^*|_B \textit{ complete} \mid \mathrm{msc}(w) \vDash \varphi\}$$

is effectively regular.

Model-checking a CFM $\mathcal{A}$ against a formula φ of some *partial-order* logics is the question whether all $M \in \mathcal{M}(\mathcal{A})$ satisfy φ.

Corollary 7.7 ([43]). *Let the bound B be given. Model-checking existentially B-bounded CFMs against* MSO *is decidable.*

The decidable problem above is known to be of non-elementary complexity. The situation changes to the better if we consider the temporal logic PDL.

Proposition 7.8 ([16, 65]). *Given a bound B and a PDL sentence φ, one can construct a finite automaton of size exponential in φ, B, and the number of channels, whose language is a set of B-representatives of $\{u \in \Sigma^*$ valid and complete $\mid \mathrm{msc}(u) \models \varphi\}$.*

The original proof from [16] (for a restricted version of the logic) goes via realizations; see Section 8.3. An alternative proof in [65] translates PDL-formulas into automata that walk along paths in an MSC, and then uses a translation of such automata into alternating, 2-way automata on words (only this second step uses the bound B). Alternatively, Proposition 7.8 can be shown by translating a PDL sentence φ over MSCs into an ETL sentence $\tilde{\varphi}$ over $\Sigma^*|_B$. The logic ETL is an extension of LTL by regular operators described by finite automata [80]. Besides atomic statements $a \in \Sigma \cup \{p!q, q?p \mid (p,q) \in \mathrm{Ch}\}$ and the Boolean connectives, it uses operators of the form $\mathcal{B}(\varphi_1, \ldots, \varphi_n)$, where $\mathcal{B}$ is an automaton over the alphabet $\{a_1, \ldots, a_n\}$ and φ_i are ETL formulas. Formulas of this logic are evaluated over positions i in sequences $w \in \Sigma^*$. Then $w, i \models \mathcal{B}(\varphi_1, \ldots, \varphi_n)$ if the automaton $\mathcal{B}$ accepts some word $a_{i_1} a_{i_2} \cdots a_{i_m}$ such that $w, i+j \models \varphi_{i_j}$ for all $1 \leqslant j \leqslant m$. For the simulation we also need the 2-way version of ETL from [53]. The latter paper shows how to translate ETL formulas into Büchi automata with an exponential blow-up.

Corollary 7.9 ([16, 65]). *For given bound B, model-checking existentially B-bounded CFMs against PDL is decidable in* PSPACE.

Further reading. The paper [65] considers non-terminating CFMs and infinite MSCs. It then makes sense to also consider formulas of the form $\langle \pi^\omega \rangle \top$ that hold for some event if it is the starting point of some infinite path that can be split into infinitely many subpaths each conforming to the path expression π. A comprehensive study of complexity of the model checking problem of CFMs against different temporal logics can be found in [66], including MSO-definable temporal logics, PDL, and more concrete logics.

Timed CFMs are considered in [26, 52, 3]. The papers [26, 52] study local timed automata with communication channels, whereas a timed extension of communicating automata based on event clocks is studied in [3]. The paper [26] defines the semantics of the automata in terms of timed MSCs and proposes a solution to an MSC-matching problem using the tool UPPAAL. A timed extension of Büchi's equivalence between automata and MSO is obtained in [3], where it is also shown that the logic is decidable over existentially bounded channels.

Systems of pushdown automata communicating via FIFO-channels are considered in [55, 50]. In that setting, bounding the channels does not suffice to make control-state reachability decidable. Instead, the paper [55] shows decidability for acyclic networks and [50] characterizes networks with a decidable existentially channel-bounded control-state reachability problem. Context- and phase-bounded analysis of CFMs were studied in several papers [55, 50, 18]. The notion of split-width is proposed and studied in [30, 31, 28, 29], in order to be able to model-check MSO properties on a model that is more general than CFMs, including communication and pushdown storage.

The same roadmap as the approach based on existential channel bounds presented in this chapter has been recently adopted for CFMs with mailbox semantics, where every process has a single mailbox for incoming messages [19]. The results obtained in [19] for the so-called synchronizability property (which is a restricted version of existentially bounded channels) are of very similar nature: it is decidable if a deadlock-free CFM is k-synchronizable. In contrast, it remains open to decide whether a deadlock-free CFM is k-synchronizable for *some* k in the mailbox semantics. In the CFM semantics that we use here (peer-to-peer), the very same question is undecidable (cf. Proposition 7.3).

Part III
Realizability

This part discusses the synthesis problem, that is, how to convert a specification into an "equivalent" CFM. As already mentioned, we assume that our systems are closed (i.e., without environment), and the problem we look at is called *realizability*. We present two solutions for realizability, one for unbounded channels (Section 8), and another one for universally- and existentially-bounded channels (Section 9.3). Both solutions use additional message contents, and we will see in Section 8.1 that this is indeed required. We formalize this variant of "equivalence" between specifications and realizations below.

Let Σ be some communication alphabet on the network $(\mathcal{P}, \mathrm{Ch})$ with message contents from Msg. Furthermore, let Σ' be the analogous communication alphabet with message contents from Msg' and let $\pi : \mathrm{Msg}' \to \mathrm{Msg}$ be a projection. This projection is extended naturally to a homomorphism $\Sigma'^* \to \Sigma^*$ by setting $\pi(p!q(m')) = p!q(\pi(m'))$ and similarly for receive actions.

Definition. A specification $K \subseteq \Sigma^*$ is *realizable* if there exists some CFM $\mathcal{A}$ with communication alphabet Σ' and a projection π from its message contents Msg' to Msg, such that $\pi(\mathcal{L}_{\mathrm{cpl}}(\mathcal{A})) = K$.

Realizability problem: Given a specification $K \subseteq \Sigma^*$, is K realizable? If so, produce a CFM that realizes K.

A set of MSCs X is called *realizable* if $\mathrm{Lin}(X)$ is.

8 Realizability with unrestricted channels

8.1 Local acceptance and sequential specifications

Let Σ be some communication alphabet and $K \subseteq \Sigma^*$ a language (the specification). The question in this section is whether K is the language of some CFM $\mathcal{A}$. This is a more restrictive question than the one in the definition at the beginning of Part III, since the message alphabet is already defined by the specification. However, we will see that this question is undecidable. The reason for undecidability is – once again – the incompatibility between the specification, which is sequential-like, and the concurrent target realization.

A necessary condition for $K \subseteq \Sigma^*$ being the language of a CFM is that K must be closed under rescheduling: $K \subseteq \Sigma^*$ is called *closed* if for all valid words $u \sim v$, we have $u \in K$ if and only if $v \in K$ (see page 1048 for the definition of the rescheduling relation $\sim$).

Note that the language of any CFM is the union of finitely many languages of CFMs with only one global accepting state. A slightly more general class is that of CFMs with local acceptance: A CFM $\mathcal{A} = \langle (\mathcal{A}_p)_{p\in\mathcal{P}}, \Sigma, F\rangle$ has *local acceptance* if there are sets $F_p \subseteq S_p$ for $p \in \mathcal{P}$ such that $F = \prod_{p\in\mathcal{P}} F_p$.

Proposition 8.1. *A regular specification $K \subseteq \Sigma^*$ equals $\mathcal{L}_{\mathrm{cpl}}(\mathcal{A})$ for some CFM $\mathcal{A}$ with local acceptance if and only if it satisfies the conditions below:*

(1) K is closed,

(2) K consists of valid and complete words, only, and

(3) every valid and complete word $v \in \Sigma^*$ such that $\pi_p(v) \in \pi_p(K)$, for every $p \in \mathcal{P}$, belongs to K, where $\pi_p : \Sigma^* \to \Sigma_p^*$ is the natural projection homomorphism.

In the situation of condition (3), [5, 6] use the terminology “the valid and complete word v is weakly implied by K”. For the proof of Proposition 8.1, let us assume the existence of some CFM $\mathcal{A}$ with local acceptance, such that $\mathcal{L}_{\mathrm{cpl}}(\mathcal{A}) = K$. Even without local acceptance, we immediately get (1) and (2). Now consider valid and complete words $v \in \Sigma^*$ and $v_p \in K$ such that $\pi_p(v) = \pi_p(v_p)$ for every process $p \in \mathcal{P}$. It is not difficult to construct an accepting run of $\mathcal{A}$ on v, from some accepting runs of $\mathcal{A}$ on the words v_p: combining the p-projection of a run on v_p, over $p \in \mathcal{P}$, to a run on v is possible, since the communication alphabet is the same, and acceptance is guaranteed by the local final states. Conversely, suppose K is regular and satisfies the conditions above. Then we can construct a CFM whose process p executes the words from the language $\pi_p(K)$ (for all $p \in \mathcal{P}$). This CFM has local acceptance and accepts K.

If $\mathcal{A}$ is a finite automaton over Σ, then to decide whether its language consists of valid and complete words only, it suffices to check three conditions: the first requires that $\mathcal{A}$ accepts no word uv with $|u|_{p!q} < |u|_{q?p}$. The second one says that $\mathcal{A}$ accepts no word $u\, p!q(m)\, v\, q?p(m')\, w$ with $m \neq m'$ and $|u|_{p!q} = |uv|_{q?p}$. The third one says that all words u accepted by $\mathcal{A}$ have as many receive actions as send actions. All these conditions can be decided, e.g., using a 1-counter automaton. If this test succeeds, we can decide easily whether the language of $\mathcal{A}$ is closed. Hence, two of the three conditions from Proposition 8.1 are decidable, but the last one is not:

Theorem 8.2 ([6]). *It is undecidable whether a regular (even LTL) specification $K \subseteq \Sigma^*$ equals $\mathcal{L}_{\mathrm{cpl}}(\mathcal{A})$ for some CFM $\mathcal{A}$ with local acceptance.*

In [6], the theorem is only stated for regular specifications, but one can check that the reduction also works for LTL specifications (where the additional modality $\mathsf{X}_{\mathrm{msg}}$ is not used). The proof relies on condition (3) from Proposition 8.1 and therefore on the local acceptance. It seems open whether the class of regular languages of CFMs is decidable. We should also stress that an important argument in the undecidability proof above is that we ask for a CFM with *language* K, as opposed to the realizability problem, where additional message contents are allowed. In Section 9.3 we will see that the realizability problem for regular languages is decidable.

8.2 MSO specifications

The theorem below is an extension of the well-known Büchi correspondence between regular and EMSO-definable sets of words (or trees, Mazurkiewicz traces, etc.), to MSCs without any channel restriction. This characterization should be compared with those provided in Section 9.3 for universally and existentially bounded CFMs. The main difference is that for automata with universal/existential channel bounds, one can use more expressive logics.

Theorem 8.3. *Let $K \subseteq \Sigma^*$ be a closed set of valid and complete words. Then the following are equivalent:*

(1) K is realizable.
(2) $\mathrm{msc}(K)$ is the language of some $\mathrm{EMSO}(\lessdot_P, <_{\mathrm{msg}})$ formula [17].
(3) $\mathrm{msc}(K)$ is the language of some $\mathrm{EMSO}^2(\leqslant, \lessdot_P, <_{\mathrm{msg}})$ formula, where EMSO^2 is the set of formulas[3] from EMSO that use only two names of first-order variables [12].

The proofs of the implications (1) → (2) and (1) → (3) are very similar to the "usual" proof, using the partial order definition of CFM runs given at the end of Section 6.1. The existence of the mapping $\rho\colon E \to \bigcup_{p\in\mathcal{P}} S_p$ is expressed by existential set quantifiers.

[3]Very recently, Bollig et al. showed that the restriction to two variables is not necessary [13].

Additional existential set quantifiers are used to express the message contents sent and received by the CFM. The message predicate $<_{\mathrm{msg}}$ is used for guaranteeing the consistency of these message contents. The local consistency of ρ is expressed using the successor predicate $\lessdot_P$.

The “usual” proof of the converse implication proceeds by induction on the structure of the formula. This requires the class of realizable specifications to be closed under complementation. But [17] shows that this is not the case! Therefore, these proofs proceed very differently. The implication (2) $\to$ (1) is inspired by W. Thomas’ graph acceptors [78]: first, it uses Hanf’s theorem [48] from model theory to transform the first-order kernel of the given formula into a specific normal form. Intuitively, a formula in this normal form is a positive Boolean combination of statements of the form “there are at most/at least k nodes of the MSC whose neighborhood of radius r looks as follows ...”. Then it provides an explicit CFM that computes, for every node of the MSC, its neighborhood of radius r. This handles the first-order kernel of the formula, while the outermost existential quantification is dealt with by a projection, as in Büchi’s proof.

The use of Hanf’s theorem also explains why this proof only works if we exclude the binary relation $\leqslant$ from formulas: with only $<_{\mathrm{msg}}$ and $\lessdot_P$, any node has at most three neighbors and therefore the size of neighborhoods of radius r is at most 3^r. Without such a finite bound, Hanf’s theorem is not applicable.

The proof of the implication (2) $\to$ (1) is not only effective, it even constructs a CFM whose size is elementary (although multiply exponential) in that of the formula [14, 15]. The reason is, again, the restriction of formulas to those mentioning only $<_{\mathrm{msg}}$ and $\lessdot_P$, but not $\leqslant$.

The proof of the implication (3) $\to$ (1) starts with a transformation of the first-order kernel into Scott normal form (cf. [47]). An involved construction allows us to transform such formulas into CFMs. As above, the outermost existential quantification is dealt with by a projection.

8.3 PDL specifications

In general, the interest in temporal logics is their rather good complexity compared with MSO. The next theorem complies with this observation, in that it provides an exponential-time solution for the realizability problem for restricted PDL specifications.

In the following, we consider the restriction rPDL of PDL that disallows that a path expression contains both, forward modalities proc or msg and backward modalities proc^{-1} or msg^{-1}. More formally *forward* and *backward path expressions* π_+ and π_- and *local formulas* α are defined by simultaneous induction:

$$\begin{aligned}
\pi_+ &::= \mathrm{proc} \mid \mathrm{msg} \mid \{\alpha\} \mid \pi_+;\pi_+ \mid \pi_+ + \pi_+ \mid \pi_+^* \,,\\
\pi_- &::= \mathrm{proc}^{-1} \mid \mathrm{msg}^{-1} \mid \{\alpha\} \mid \pi_-;\pi_- \mid \pi_- + \pi_- \mid \pi_-^* \,,\\
\alpha &::= \mathrm{true} \mid \sigma \mid \alpha \vee \alpha \mid \neg\alpha \mid \langle \pi_+ \rangle\, \alpha \mid \langle \pi_- \rangle\, \alpha \,,
\end{aligned}$$

where σ ranges over the alphabet Σ. Global formulas of rPDL are then defined analogously to those of PDL.

Theorem 8.4 ([16]). *rPDL specifications are realizable in exponential time. More precisely, from a global rPDL formula φ, one can compute in exponential time a CFM $\mathcal{A}$ realizing* $\mathrm{Lin}(\{M \text{ } MSC \mid M \models \varphi\})$.

The proof follows the inductive approach from [39, 40]: Let α_{n+1} be a local formula of PDL whose top-most local subformulas are $\alpha_1, \ldots, \alpha_n$. Then one builds a CFM that accepts MSCs whose events carry $n+1$ additional bits (one for each top-most subformula and the last one for the formula α_{n+1}). Let $M = (E, \leqslant, \lambda)$ be an MSC and, for $1 \leqslant i \leqslant n+1$, let $X_i \subseteq E$ be the set of events where bit i is set to 1. Furthermore let α'_{n+1} be obtained from α_{n+1} by replacing every top-most occurrence of α_i by the informal statement "bit i is set to 1" (for $1 \leqslant i \leqslant n$). Then the CFM accepts $(M, X_1, \ldots, X_n, X_{n+1})$ if and only if $X_{n+1} = \{e \in E \mid M, e \models \alpha'_{n+1}\}$. For example, if $\alpha_2 = \neg\alpha_1$, then the CFM accepts MSCs M with two additional bits per node that have to be distinct at every node. The crucial point is that this CFM does not depend on the formulas $\alpha_1, \ldots, \alpha_n$, but only on the outermost operator of α_{n+1}. The construction is rather obvious for the operators of the form $\sigma \in \Sigma$ (without subformulas), $\vee$ and $\langle\{.\}\rangle$. (with two subformulas), and $\neg.$, $\langle \mathrm{proc} \rangle$. and the like (with one subformula). But also the path formulas $\alpha_2 = \langle \pi_+ \rangle\, \alpha_1$ and $\langle \pi_- \rangle^{-1} \alpha_1$ cause no deep problem (we discuss the forward-path formula, only). At each and every event, one starts a copy of a finite automaton accepting the word language that is described by π_+. Depending on whether the event was claimed to satisfy or not to satisfy α_2 (i.e., belongs to X_2 or does not belong to X_2), this automaton is simulated along one or all outgoing edges and is required to accept or not to accept at an event satisfying α_1.[4]

Combining Theorems 8.3 and 8.4, we infer that any rPDL formula can be translated into an equivalent formula from $\mathrm{EMSO}(\lessdot_P, <_{\mathrm{msg}})$. We do not know whether this is also possible for full PDL, where path expressions can mix forward and backward modalities. The converse translation, from $\mathrm{EMSO}(\lessdot_P, <_{\mathrm{msg}})$ to rPDL, is not always possible: Since the class of realizable MSC-languages is not closed under complementation [17], Theorem 8.3 implies that $\mathrm{EMSO}(\lessdot_P, <_{\mathrm{msg}})$ is (semantically) not closed under complementation. But the set of PDL formulas is closed under negation (using the usual de Morgan rules). Hence, PDL is a proper fragment of $\mathrm{EMSO}(\lessdot_P, <_{\mathrm{msg}})$, but no nontrivial description of this fragment is known. We do not even know whether the set of formulas from $\mathrm{EMSO}(\lessdot_P, \mathrm{msg})$ that can be translated into PDL is decidable.

9 Channel bounds, Mazurkiewicz traces and realizability

This section presents a detailed study of CFMs with universal and existential channel bounds. The ultimate goal is to show a Büchi-like equivalence between automata and MSO in the setting of CFMs with channel bounds. In other words, one obtains a solution for the realizability problem for regular specifications (universal bounds), but also some non-regular ones (existential bounds). We start with a brief survey on the well-studied partial order model of Mazurkiewicz traces, then present the link between traces and

[4]The main complication in [16] stems from the more general setting of infinite runs considered there.

channel bounds, and finally conclude with the realizability theorems.

9.1 Mazurkiewicz traces

Mazurkiewicz traces, known for a long time in combinatorics as partially commutative monoids, were introduced in computer science in the late seventies by Mazurkiewicz [64] for describing the behavior of 1-safe Petri nets. Their essential feature is to express the semantics of a concurrent system by a (static) relation of independence between actions. Formally, a *trace alphabet* is a pair (A, I) consisting of an alphabet A of actions and a symmetric and irreflexive relation $I \subseteq A^2$. The relation I is referred to as the *independence relation*; its complement $D = A^2 \setminus I$ is the *dependence relation*.

A *Mazurkiewicz trace* is a finite A-labeled partial order $\langle E, \leqslant, \lambda\rangle$ – up to isomorphism – with the labeling $\lambda : E \to A$ satisfying both conditions below, for any events $e, f \in E$:

- if e is an immediate predecessor of f (denoted as $e \lessdot f$), then $(\lambda(e), \lambda(f)) \in D$,
- if e and f are incomparable, then $(\lambda(e), \lambda(f)) \in I$.

The study of automata and logics over Mazurkiewicz traces has yielded several elegant results, generalizing the situation of word languages. One prominent example is the Kleene-Büchi theorem stating the equivalence between automata, monadic second-order logic and regular expressions, which was generalized to languages of finite and infinite traces (see [70, 77] or [33] for a collection of surveys on trace theory). A second example is the equivalence between first-order logic and linear temporal logics over traces [76, 32]. However, the deepest result of trace theory is undoubtedly the construction of deterministic *Zielonka automata* (also known as asynchronous automata) from finite automata [81]. It is this particular result which will be used for constructing CFMs with bounded channels in Section 9.3. For sake of completeness we introduce these automata and Zielonka's theorem in the rest of this section.

A *Zielonka automaton* over a trace alphabet (A, I) is a product of local automata synchronizing over common actions. The trace alphabet is described by a distribution of the actions on processes, given by a function $\mathrm{dom} : A \to (2^{\mathcal{P}} \setminus \{\emptyset\})$ associating each action $a \in A$ with a (non-empty) set of processes $\mathrm{dom}(a)$. Moreover, $(a, b) \in I$ if and only if the respective domains are disjoint, i.e., $\mathrm{dom}(a) \cap \mathrm{dom}(b) = \emptyset$. A Zielonka automaton $\mathcal{A} = \langle (S_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in A}, s^0, F\rangle$ consists of a finite set of states S_p for each process p, a transition relation $\delta_a \subseteq (\prod_{p\in\mathrm{dom}(a)} S_p)^2$ for each letter, an initial state $s^0 \in \prod_{p\in\mathcal{P}} S_p$ and a set of final states $F \subseteq \prod_{p\in\mathcal{P}} S_p$. The language of such an automaton is defined as the language of the finite automaton $\mathcal{B}$ with set of states $S = \prod_{p\in\mathcal{P}} S_p$ and transition relation $\delta \subseteq S \times A \times S$ given by $(s, a, s') \in \delta$ if $((s_p)_{p\in\mathrm{dom}(a)}, (s'_p)_{p\in\mathrm{dom}(a)}) \in \delta_a$ and $s'_q = s_q$ for all $q \notin \mathrm{dom}(a)$. The automaton is deterministic if δ_a is a function from $\prod_{p\in\mathrm{dom}(a)} S_p$ to $\prod_{p\in\mathrm{dom}(a)} S_p$, for each a. The language $L = \mathcal{L}(\mathcal{A})$ of such an automaton is I-closed: $uabv \in L$ implies $ubav \in L$, for every $u, v \in A^*$, $(a, b) \in I$.

Theorem 9.1 ([81]). *A language $L \subseteq A^*$ is I-closed and regular if and only if it is accepted by a deterministic Zielonka automaton.*

In terms of complexity, the original construction of Zielonka has been improved stepwise over the years: currently the best upper bound on the overall number of local states of the Zielonka automaton is $O(2^{|\mathcal{P}|^4} \cdot |\mathcal{A}|^{|\mathcal{P}|^2})$, with $\mathcal{A}$ the minimal deterministic automaton of L [42]. Surprisingly, an exponential lower bound in the number of processes is known only for a subclass of Zielonka automata [42]. When L is given by a non-deterministic automaton $\mathcal{A}$, then [45] gives an asymptotically optimal construction that is simply exponential in both $|\mathcal{A}|$ and $|\mathcal{P}|$.

9.2 Bounds and traces

Recall the definition of a B-bounded valid and complete word from Section 7.1, as well as the fact that linearizations of MSCs are necessarily valid and complete. It therefore makes sense to say that an MSC M is existentially (resp., universally) B-bounded if and only if one (resp., all) linearizations of M are B-bounded.

This matches the definition of existentially or universally B-bounded valid and complete words: an MSC M is existentially or universally B-bounded if and only if $\mathrm{Lin}(M)$ consists of existentially or universally B-bounded words, only. A set of MSCs is existentially or universally B-bounded if and only each of its members is so.

We describe now encodings of universally (resp., existentially) B-bounded MSCs into traces, following [54] and [43]. The trace alphabet (A, I) is given by $A = \Sigma \times \{0, \ldots, B-1\}$ and the following dependence relation $D \subseteq A \times A$: let $(a, i) D (b, j)$ if either $a, b \in \Sigma_p$ for some $p \in \mathcal{P}$, or $i = j$ and $\{a, b\} = \{p!q(m), q?p(m)\}$ for some $(p, q) \in \mathrm{Ch}, m \in \mathrm{Msg}$. Clearly, $I = A^2 \setminus D$ is symmetric and irreflexive, and hence (A, I) is a trace alphabet.

Since the universally bounded case is easier, we start with this case as preparation. The encoding $\mathrm{tr}(M)$ of a *universally* B-bounded MSC $M = \langle E, \leqslant, \lambda \rangle$ is obtained similarly to the indexing of a word on page 1051: If $e \in E$, then $\lambda_B(e) = (\lambda(e), n)$ such that

- e is of type $q?p$ and $n = |\{e' < e \mid e' \text{ is of type } q?p\}| \bmod B$, or
- e is of type $p!q$ and $n = |\{e' < e \mid e' \text{ is of type } p!q\}| \bmod B$.

This yields a trace: if e is an immediate predecessor of f, then they belong to the same process or they are matching send and receive (in which case $\lambda_B(e) = (p!q(m), n)$ and $\lambda_B(f) = (q?p(m), n)$ are dependent). If $\lambda_B(e)$ and $\lambda_B(f)$ are dependent, then they belong to the same process or they equal $(p!q(m), n)$ and $(q?p(m), n)$ or vice versa. But then, universal B-boundedness of M implies that e and f are comparable.

For an *existentially* B-bounded MSC $M = \langle E, \leqslant, \lambda \rangle$ we need to introduce an additional binary relation rev_B on events. Let $(r, s') \in \mathrm{rev}_B$ if, for some channel $(p, q) \in \mathrm{Ch}$: event r has type $q?p$ and its matching send is event s, event $s' > s$ has type $p!q$ and $|\{s'' \mid s < s'' \leqslant s' \text{ and } s'' \text{ has type } p!q\}| = B$. That is, the relation rev_B matches a receive event r with the B-th send of the same type after the matching send of r. Let $\preceq_B$ be the reflexive-transitive closure of $\leqslant \cup \mathrm{rev}_B$ and let $\mathrm{tr}(M) = \langle E, \preceq_B, \lambda_B \rangle$, with λ_B as above. It is easy to check that we have $\mathrm{rev}_B \subseteq \leqslant$ if and only if M is universally B-bounded, so the two definitions of $\mathrm{tr}(M)$ coincide in this case. Figure 4 shows an example for a trace $\mathrm{tr}(M)$ associated with an existentially 2-bounded MSC. The dashed edges are rev_2-edges.

The relation $\preceq_B$ is, in general, not a partial order, since it could contain a cycle.

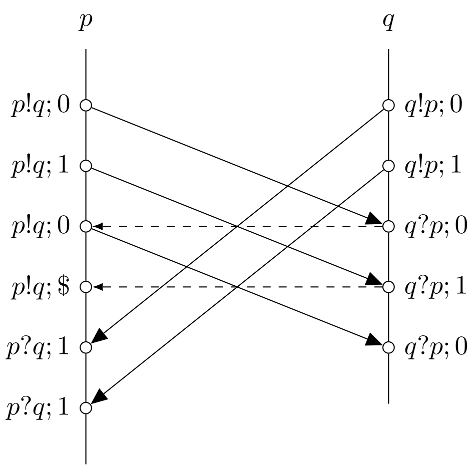

Figure 4. Trace $\mathrm{tr}(M)$ associated with an existentially 2-bounded MSC. Diagonal edges are messages, dashed edges are rev_2-edges. On each vertical line, events are ordered from top to bottom.

Lemma 9.2 ([59, 43]). *Let M be an MSC.*

- *M is existentially B-bounded if and only if $\preceq_B$ is acyclic, if and only if $\mathrm{tr}(M)$ is a trace over the alphabet (A, I).*
- *If M is existentially B-bounded, then $\mathrm{Lin}(M) \cap \Sigma^*|_B = \mathrm{Lin}(\mathrm{proj}_\Sigma(\mathrm{tr}(M)))$, where proj_Σ is the projection of the labeling on the first component of A.*
- *If M is universally B-bounded, then $M = \mathrm{proj}_\Sigma(\mathrm{tr}(M))$.*

We summarize the connection between languages of CFMs with channel bounds and trace languages in the proposition below. We say that $K \subseteq \Sigma^*$ is *B-closed*[5] if for all valid words $u, v \in \Sigma^*|_B$ with $u \sim v$, it holds that $u \in K$ if and only if $v \in K$. Note that K is closed if, and only if, it is B-closed for all $B \in \mathbb{N}$.

Closed regular languages of valid and complete words are those of universally bounded CFMs (cf. Theorem 9.5), whereas closures of B-closed regular languages of valid and complete words are those of existentially B-bounded CFMs (cf. Theorem 9.7).

It can be decided in polynomial space whether L is B-closed, for L given by an automaton and B in unary encoding.

Proposition 9.3 ([54, 43]). *Given a bound B, let $K \subseteq \Sigma^*|_B$ be a regular language of valid and complete words that is B-closed. Set $L = \mathrm{Lin}(\mathrm{tr}(\mathrm{msc}(K))) \subseteq A^*$.*

Then L is a regular and I-closed language over (A, I). Moreover, $K = \mathrm{proj}_\Sigma(L)$.

Remark 9.4. Let $\mathcal{A}$ be a universally B-bounded CFM. Then $\mathcal{L}_{\mathrm{cpl}}(\mathcal{A})$ is a closed and regular set of valid and complete B-bounded words. Hence, by the proposition above, there exists a regular and I-closed language L with $\mathcal{L}(\mathcal{A}) = \mathrm{proj}_\Sigma(L)$.

[5]The reader may compare this notion of B-closed with the definition of closed languages (see page 1054).

Next let $\mathcal{A}$ be an existentially B-bounded CFM. Then $\mathcal{L}_{\mathrm{cpl}}(\mathcal{A})$ is a closed set of valid and complete words, but $\mathcal{L}_{\mathrm{cpl}}(\mathcal{A})$ is in general neither regular nor a set of B-bounded words. However, the set $\mathcal{L}_{\mathrm{cpl}}(\mathcal{A}) \cap \Sigma^*|_B$ of B-bounded words from $\mathcal{L}_{\mathrm{cpl}}(\mathcal{A})$ is B-closed and regular. Hence, by the proposition above, there is a regular I-closed language L with $\mathcal{L}_{\mathrm{cpl}}(\mathcal{A}) \cap \Sigma^*|_B = \mathrm{proj}_\Sigma(L)$. Since $\mathcal{M}(\mathcal{A})$ is existentially B-bounded, we get $\mathcal{M}(\mathcal{A}) = \mathrm{msc}(\mathcal{L}(\mathcal{A})) = \mathrm{msc}(\mathrm{proj}_\Sigma(L))$.

9.3 Realizability with bounded channels

As already mentioned, universal or existential channel bounds give a robust framework for MSC languages, in which we can state the Büchi-like equivalence between automata and monadic second-order logic, cf. Theorems 9.5 and 9.7. In particular, the results show that realizability for specifications based on regular sets or on full monadic second-order logic (with any of the relations $\leqslant$, $\lessdot_P$, $<_{\mathrm{msg}}$) is decidable. For regular specifications, realizability can be solved rather efficiently in exponential time, see Corollary 9.6 and 9.8.

A language $K \subseteq \Sigma^*$ is *message deterministic* if, whenever $u\,p!q(m)$ and $u\,p!q(n)$ are prefixes of some elements of K, then $m = n$. Note that the language of every deterministic CFM is message deterministic. It follows that only message deterministic languages can be realized by deterministic CFMs. A language is *deterministically* realizable if it can be realized by a deterministic CFM.

Theorem 9.5 ([49]). *Given a bound B, let $K \subseteq \Sigma^*|_B$ be a closed set of valid and complete words. Then the following assertions are equivalent:*

(1) K is regular.
(2) $\mathrm{msc}(K)$ is the language of some formula of the logic $\mathrm{MSO}(\lessdot_P, <_{\mathrm{msg}})$ (or, equivalently, of $\mathrm{MSO}(\leqslant, <_{\mathrm{msg}})$, or of $\mathrm{MSO}(\leqslant)$).
(3) $\mathrm{msc}(K)$ is the language of some formula of the logic $\mathrm{EMSO}(\lessdot_P, <_{\mathrm{msg}})$ (or, equivalently, of $\mathrm{EMSO}(\leqslant, <_{\mathrm{msg}})$, or of $\mathrm{EMSO}(\leqslant)$).
(4) K is realizable.

In addition, the following are equivalent:

(1') K is regular and message deterministic.
(5) K is deterministically realizable.

Note that the realizing CFMs in (4) and (5) are necessarily universally B-bounded, since any word from K is universally B-bounded. Note also that the implication (4) $\to$ (3) for $\mathrm{EMSO}(\lessdot_P, <_{\mathrm{msg}})$ is a special instance of Theorem 8.3, which also implies (3) for $\mathrm{EMSO}(\leqslant, <_{\mathrm{msg}})$. For the logic $\mathrm{EMSO}(\leqslant)$, the implication (4) $\to$ (3) was shown in [54] using the fact that $\mathrm{msc}(K)$ consists of universally B-bounded MSCs, only. The implication (2) $\to$ (1) follows by translating the formula that talks about MSCs to an MSO formula talking about Mazurkiewicz traces, the equivalence between MSO and regular I-closed languages [79, 34], and the closure of regular languages under projections.

The proofs of the implications (1) $\to$ (4) and (1') $\to$ (5) use the encoding described in Section 9.2. The encoding was first presented in [54], where the Büchi characterization above is extended to infinite MSCs. The main idea is to use Proposition 9.3 in order to obtain a regular and I-closed language $L \subseteq A^*$ with $K = \mathrm{proj}_\Sigma(L)$. Then, Theorem 9.1 gives a deterministic Zielonka automaton $\mathcal{B}$ for L, which can be simulated by some CFM.

Corollary 9.6. *Let B be a fixed bound. Then any closed, regular specification $K \subseteq \Sigma^*|_B$ of valid and complete words is realizable in exponential time. If K is given by a DFA $\mathcal{B}$ and message deterministic, then one can construct a deterministic, universally B-bounded CFM $\mathcal{A}$ realizing K, of size polynomial in the size of $\mathcal{B}$ and exponential in $|\operatorname{Ch}|$ and B.*

The complexity stated in the corollary relies on the construction of Zielonka automata given in [42]; also see Section 9.1. Notice that we apply the construction to the "modulo B relabeling" of K (i.e., the word language $\operatorname{Lin}(\operatorname{tr}(\operatorname{msc}(K)))$), so we start with a DFA of size polynomial in $\mathcal{B}$ and exponential in Ch. Then we obtain a deterministic Zielonka automaton of size that is still polynomial in $\mathcal{B}$, and (simply) exponential in Ch and $|\overline{\mathcal{P}}| = |\operatorname{Ch}| \cdot B$; hence in both B and Ch. The resulting automaton includes the time-stamping needed in the CFM simulation.

The generalization of Theorem 9.5 to the existentially bounded case is less obvious. The reason is that although Proposition 9.3(2) offers a similar encoding by traces as in the universally bounded case, a CFM cannot easily simulate the corresponding Zielonka automaton. The reader might try to define a CFM that realizes the set of existentially B-bounded MSCs. Whereas this task is not very difficult to accomplish (even deterministically) in the universally bounded case, it becomes much more involved in the existentially bounded case; see [43].

Theorem 9.7 ([43]). *Given a bound B, let $K \subseteq \Sigma^*|_B$ be a B-closed set of valid and complete words. Then the following assertions are equivalent:*

(1) *K is regular.*
(2) $\operatorname{msc}(K)$ *is the language of some formula of one of the logics* $\operatorname{MSO}(\lessdot_P, <_{\mathrm{msg}})$, $\operatorname{MSO}(\leqslant, <_{\mathrm{msg}})$, *or* $\operatorname{MSO}(\leqslant)$, *respectively.*
(3) $\operatorname{msc}(K)$ *is the language of some formula of one of the logics* $\operatorname{EMSO}(\lessdot_P, <_{\mathrm{msg}})$, $\operatorname{EMSO}(\leqslant, <_{\mathrm{msg}})$, *or* $\operatorname{EMSO}(\leqslant)$, *respectively.*
(4) $\operatorname{msc}(K)$ *is realizable.*

Note that the realizing CFM in (4) is necessarily existentially B-bounded, since any word from K is existentially B-bounded. The main difference between this and Theorem 9.5 is that K is only B-closed (and not necessarily closed). Given this more general language K, we cannot construct a deterministic CFM as in Theorem 9.5, as the following example shows:

Example 9.1. Let $\mathcal{P} = \{p, q, r\}$ and consider the following existentially 1-bounded set X of MSCs. Let $M \in X$ if the following properties hold:

- Process p alternates between sends to q and to r (with fixed content m).
- Between every pair of consecutive receives from p, process q (resp., r) executes 0 or 1 local actions a_q (resp., a_r).
- Let u_q (resp., u_r) be the 0-1 sequence of local actions a_q (resp., a_r) executed by q (resp., r) between two consecutive receives from p. Then $u_q = u_r$.

It is fairly easy to see that X can be realized by a non-deterministic CFM where process p chooses between sending (to both q and r) a bit $b \in \{0, 1\}$, telling q and r to execute b local actions before the next receive. But any deterministic CFM fails to realize X.

Suppose, to get a contradiction, that a deterministic CFM $\mathcal{A}$ realizes X. Notice first that there is a unique infinite run ρ_0 on the process p, such that every finite run ρ of $\mathcal{A}$ is a prefix of ρ_0 when projected to process p. If ρ is large enough, then there exist two MSCs $M_1, M_2 \in X$ with the same projection on p, but different projections on q, that reach the same global state in $\mathcal{A}$ (with empty channels). For these two MSCs, the runs on p are identical and we can combine them to an accepting run on the MSC M that coincides with p, q on M_1 and with p, r on M_2. This yields $M \in X$, a contradiction.

The proofs of Theorems 9.5 and 9.7 are very similar except for the implication (1) $\rightarrow$ (4), i.e., the transformation of a regular language K into a CFM. It is shown again using a Zielonka automaton accepting the recognizable trace language corresponding to K. However, one faces two additional problems here. First, recall that the conversion of an existentially bounded MSC into a Mazurkiewicz trace requires rev_B-edges, which are in a sense "virtual", since not implied by the causal order of the MSC. Therefore, the simulation of a Zielonka automaton by the CFM is non-deterministic, since the information conveyed by the rev_B-edges in the runs of the Zielonka automaton has to be guessed in the CFM. Second, one needs to construct a CFM accepting the set of all existentially B-bounded MSCs. The test that the relation $\leqslant \cup \mathrm{rev}_B$ is acyclic is the most intricate part of the proof, and makes use of non-determinism, too. The main idea is to look for particular cycles of bounded length, and to check the non-existence of such cycles by guessing a suitable labeling of events and counting particular events.

A further difference between the proofs of Theorems 9.5 and 9.7 concerns the translation of a formula from $\mathrm{EMSO}(\leqslant, <_{\mathrm{msg}})$ into an equivalent one from $\mathrm{EMSO}(\leqslant)$. The problem occurs since one needs to express the set of existentially B-bounded MSCs by an $\mathrm{EMSO}(\leqslant)$-formula. For this, one can use the CFM realizing the set of complete and existentially B-bounded words, see [43].

Although the constructions in [43] are quite involved, the asymptotic complexity is the same as in the universally bounded case:

Corollary 9.8. *Let the bound B be fixed. Then for any B-closed, regular specification $K \subseteq \Sigma^*|_B$ of valid and complete words, the set $\mathrm{msc}(K)$ is realizable in exponential time. If K is given by a DFA $\mathcal{B}$ then one can construct a non-deterministic, existentially B-bounded CFM $\mathcal{A}$ realizing $\mathrm{msc}(K)$ of size polynomial in the size of $\mathcal{B}$ and exponential in $|\mathcal{P}|$ and B.*

Further reading. The negative result of Theorem 8.2 is counterbalanced by the following positive result (see [4] and [58] for the matching lower bound): it is decidable in exponential space whether a regular specification is the language of a deadlock-free CFM with local acceptance.

Throughout this section we considered only CFMs with finite behavior. We briefly mention some related results on CFMs with infinite behavior. Theorem 9.5 has been extended to universally bounded sets of infinite MSCs and CFMs with Büchi and Muller acceptance in [54]. Subsequently, [14] has generalized the logical characterization of arbitrary CFMs from [17] by showing that for infinite behaviors, Muller and Büchi acceptance are equivalent, and also equivalent to $\mathrm{EMSO}(\lessdot_P, <_{\mathrm{msg}}) + \exists^\infty$ that extends $\mathrm{EMSO}(\lessdot_P, <_{\mathrm{msg}})$ by the quantifier $\exists^\infty$ expressing that there exist infinitely many nodes with a given property. It is open whether EMSO and CFMs are equally expressive for infinite MCSs.

Theorems 9.5 and 9.7 state Büchi equivalences between automata and logics over universally and existentially bounded MSCs. We did not mention a third characterization in terms of CMSC-graphs that corresponds roughly to regular expressions over MSCs. In that setting one imposes restrictions on loops of such graphs, known resp. as local synchronization and global cooperation, in order to capture universally and existentially bounded "regular" sets of MSCs, respectively, see [67, 49, 43].

A different notion of realizability was considered in [41], where the specification talks only about local actions, so that a CFM realization amounts to synthesize the necessary communication over a fixed architecture. Several decidable notions of deadlock-free realizability are proposed in that paper.

References

[1] P. A. Abdulla, K. Cerans, B. Jonsson, and Y.-K. Tsay. General decidability theorems for infinite state systems. In *IEEE Symposium on Logic in Computer Science (LICS)*, pages 313–323. IEEE Computer Society Press, 1996. 1031, 1038

[2] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Inform. and Comput.*, 127(2):91–101, 1996. 1031, 1038, 1039

[3] S. Akshay, B. Bollig, and P. Gastin. Automata and logics for timed message sequence charts. In *27th International Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS)*, number 4855 in LNCS, pages 290–302. Springer, 2007. 1053

[4] R. Alur, K. Etessami, and M. Yannakakis. Realizability and verification of MSC graphs. In *28th International Colloquium on Automata, Languages and Programming (ICALP)*, number 2076 in LNCS, pages 797–808. Springer, 2001. 1063

[5] R. Alur, K. Etessami, and M. Yannakakis. Inference of message sequence charts. *IEEE Trans. Software Eng.*, 29(7):623–633, 2003. 1054

[6] R. Alur, K. Etessami, and M. Yannakakis. Realizability and verification of MSC graphs. *Theor. Comp. Science*, 331(1):97–114, 2005. 1054, 1055

[7] R. Alur, D. Peled, and W. Penczek. Model-checking of causality properties. In *10th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 90–100. IEEE Computer Society Press, 1995. 1047

[8] M. F. Atig, A. Bouajjani, and T. Touili. On the reachability analysis of acyclic networks of pushdown systems. In *19th International conference on concurrency theory (CONCUR)*, number 5201 in LNCS, pages 356–371. Springer, 2008. 1031

[9] C. Baier, N. Bertrand, and Ph. Schnoebelen. Verifying nondeterministic probabilistic channel systems against omega-regular linear-time properties. *ACM Trans. on Comp. Logic*, 9(1), 2007. 1040

[10] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. In *8th International Conference on Computer Aided Verification (CAV)*, number 1102 in LNCS, pages 1–12. Springer, 1996. 1031, 1036

[11] B. Boigelot, P. Godefroid, B. Willems, and P. Wolper. The power of QDDs. In *4th International Symposium on Static Analysis (SAS)*, number 1302 in LNCS. Springer, 1997. 1031, 1037

[12] B. Bollig, M. Fortin, and P. Gastin. Communicating finite-state machines and two-variable logic. In *35th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 96 of *LIPIcs*, pages 17:1–17:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. 1055

[13] B. Bollig, M. Fortin, and P. Gastin. It is easy to be wise after the event: Communicating finite-state machines capture first-order logic with "happened before". *CoRR*, abs/1804.10076, 2018. To appear in 29th International conference on concurrency theory (CONCUR), 2018. 1055

[14] B. Bollig and D. Kuske. Muller message-passing automata and logics. *Inform. and Comput.*, 206:1084–1094, 2008. 1056, 1063

[15] B. Bollig and D. Kuske. An optimal construction of Hanf sentences. *Journal of Applied Logic*, 10:179–186, 2012. 1056

[16] B. Bollig, D. Kuske, and I. Meinecke. Propositional Dynamic Logic for message-passing systems. *Logical Methods in Computer Science*, 6(3:16):1–31, 2010. 1052, 1057

[17] B. Bollig and M. Leucker. Message-passing automata are expressively equivalent to EMSO logic. *Theor. Comp. Science*, 358(2-3):150–172, 2006. 1055, 1056, 1057, 1063

[18] A. Bouajjani and M. Emmi. Bounded phase analysis of message-passing programs. *International Journal on Software Tools for Technology Transfer (STTT)*, 16(2):127–146, 2014. 1053

[19] A. Bouajjani, C. Enea, K. Ji, and S. Qadeer. On the completeness of verifying message passing programs under bounded asynchrony. In *30th International Conference on Computer Aided Verification (CAV)*, number 10981 in LNCS. Springer, 2018. 1031, 1053

[20] A. Bouajjani and P. Habermehl. Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations. *Theor. Comp. Science*, 221(1-2):211–250, 1999. 1031, 1037

[21] P. Bouyer, N. Markey, J. Ouaknine, Ph. Schnoebelen, and J. Worrell. On termination for faulty channel machines. In *25th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1 of *LIPIcs*, pages 121–132. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2008. 1039, 1040

[22] D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983. 1030, 1032, 1035

[23] G. Cécé, A. Finkel, and S. Iyer. Unreliable channels are easier to verify than perfect channels. *Inform. and Comput.*, 124:20–31, 1996. 1039, 1040

[24] P. Chambart and Ph. Schnoebelen. Mixing lossy and perfect Fifo channels. In *19th International conference on concurrency theory (CONCUR)*, number 5201 in LNCS, pages 340–355. Springer, 2008. 1040

[25] P. Chambart and Ph. Schnoebelen. The ordinal recursive complexity of lossy channel systems. In *23rd Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 205–216. IEEE Computer Society Press, 2008. 1039

[26] P. Chandrasekaran and M. Mukund. Matching scenarios with timing constraints. In *FORMATS'06*, number 4202 in LNCS, pages 98–112. Springer, 2006. 1053

[27] A. Church. Logic, arithmetics, and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35, 1962. 1030

[28] L. Clemente, F. Herbreteau, and G. Sutre. Decidable topologies for communicating automata with FIFO and bag channels. In *25th International conference on concurrency theory (CONCUR)*, number 8704 in LNCS, pages 281–296. Springer, 2014. 1031, 1053

[29] A. Cyriac. *Verification of Communicating Recursive Programs via Split-width*. PhD thesis, LSV, ENS Cachan, 2014. 1031, 1053

[30] A. Cyriac, P. Gastin, and K. N. Kumar. MSO decidability of multi-pushdown systems via split-width. In *23rd International conference on concurrency theory (CONCUR)*, number 7454 in LNCS, pages 547–561. Springer, 2012. 1031, 1053

[31] A. Cyriac, P. Gastin, and K. N. Kumar. Controllers for the verification of communicating multi-pushdown systems. In *25th International conference on concurrency theory (CONCUR)*, number 8704 in LNCS, pages 297–311. Springer, 2014. 1031, 1053

[32] V. Diekert and P. Gastin. Pure future local temporal logics are expressively complete for Mazurkiewicz traces. *Inform. and Comput.*, 204:1597–1619, 2006. 1058

[33] V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, Singapore, 1995. 1058

[34] W. Ebinger and A. Muscholl. Logical definability on infinite traces. *Theor. Comp. Science*, 154:67–84, 1996. 1061

[35] J. Esparza. Decidability and complexity of Petri net problems - an introduction. In W. Reisig and G. Rozenberg, editors, *Advanced Course on Petri Nets 1996*, number 1491 in LNCS, pages 374–428. Springer, 1998. 1051

[36] A. Finkel. A generalization of the procedure of Karp and Miller to well structured transition systems. In *14th International Colloquium on Automata, Languages and Programming (ICALP)*, number 267 in LNCS, pages 499–508. Springer, 1987. 1038

[37] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theor. Comp. Science*, 256(1-2):63–92, 2001. 1031, 1038, 1039

[38] M. Fischer and R. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Sys. Sci.*, 18:194–211, 1979. 1047

[39] P. Gastin and D. Kuske. Satisfiability and model checking for MSO-definable temporal logics are in PSPACE. In *14th International conference on concurrency theory (CONCUR)*, number 2761 in LNCS, pages 222–236. Springer, 2003. 1057

[40] P. Gastin and D. Kuske. Uniform satisfiability problem for local temporal logics over Mazurkiewicz traces. *Inform. and Comput.*, 208:797–816, 2010. 1057

[41] B. Genest. On implementation of global concurrent systems with local asynchronous controllers. In *16th International conference on concurrency theory (CONCUR)*, number 3653 in LNCS, pages 443–457. Springer, 2005. 1064

[42] B. Genest, H. Gimbert, A. Muscholl, and I. Walukiewicz. Optimal Zielonka-type construction of deterministic asynchronous automata. In *37th International Colloquium on Automata, Languages and Programming (ICALP)*, number 6199 in LNCS, pages 52–63. Springer, 2010. 1059, 1062

[43] B. Genest, D. Kuske, and A. Muscholl. A Kleene theorem and model checking algorithms for existentially bounded communicating automata. *Inform. and Comput.*, 204(6):920–956, 2006. 1031, 1052, 1059, 1060, 1062, 1063, 1064

[44] B. Genest, D. Kuske, and A. Muscholl. On communicating automata with bounded channels. *Fundam. Inform.*, 80(1-3):147–167, 2007. 1050

[45] B. Genest and A. Muscholl. Constructing exponential-size deterministic Zielonka automata. In *33rd International Colloquium on Automata, Languages and Programming (ICALP)*, number 4052 in LNCS, pages 565–576, 2006. 1059

[46] B. Genest, A. Muscholl, H. Seidl, and M. Zeitoun. Infinite-state high-level MSCs: Model-checking and realizability. *J. Comput. Syst. Sci.*, 72(4):617–647, 2006. 1031

[47] E. Grädel and M. Otto. On logics with two variables. *Theor. Comput. Sci.*, 224(1-2):73–113, 1999. 1056

[48] W. Hanf. The theory of models. In J. W. Addison, L. Henkin, and A. Tarski, editors, *Model-Theoretic Methods in the Study of Elementary Logic*. North-Holland, Amsterdam, 1965. 1056

[49] J. G. Henriksen, M. Mukund, K. N. Kumar, M. Sohoni, and P. Thiagarajan. A theory of regular MSC languages. *Inform. and Comput.*, 202(1):1–38, 2005. 1031, 1034, 1061, 1064

[50] A. Heußner, J. Leroux, A. Muscholl, and G. Sutre. Reachability analysis of communicating pushdown systems. *Logical Methods in Computer Science*, 8(3), 2012. 1031, 1035, 1050, 1053

[51] S. Kosaraju. Decidability of reachability in vector addition systems. In *14th Annual ACM Symposium on Theory of Computing (STOC)*, pages 267–281. ACM, 1982. 1038

[52] P. Krcal and W. Yi. Communicating timed automata: The more synchronous, the more difficult to verify. In *18th International Conference on Computer Aided Verification (CAV)*, number 4144 in LNCS, pages 249–262. Springer, 2006. 1053

[53] O. Kupferman, N. Piterman, and M. Y. Vardi. Extended temporal logic revisited. In *12th International conference on concurrency theory (CONCUR)*, number 2154 in LNCS, pages 519–535. Springer, 2001. 1052

[54] D. Kuske. Regular sets of infinite message sequence charts. *Inform. and Comput.*, 187:80–109, 2003. 1059, 1060, 1061, 1063

[55] S. La Torre, P. Madhusudan, and G. Parlato. Context-bounded analysis of concurrent queue systems. In *14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, number 4963 in LNCS, pages 299–314. Springer, 2008. 1031, 1035, 1053

[56] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978. 1042

[57] J. Leroux and S. Schmitz. Demystifying reachability in vector addition systems. In *30th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 56–67. IEEE Computer Society Press, 2015. 1038

[58] M. Lohrey. Realizability of high-level message sequence charts: closing the gaps. *Theor. Comp. Science*, 309(1-3):529–554, 2003. 1063

[59] M. Lohrey and A. Muscholl. Bounded MSC communication. *Inform. and Comput.*, 189(2):160–181, 2004. 1060

[60] P. Madhusudan and B. Meenakshi. Beyond message sequence graphs. In *21st International Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS)*, number 2245 in LNCS, pages 256–267. Springer, 2001. 1052

[61] Z. Manna and A. Pnueli. Verification of the concurrent programs: the temporal framework. In R. Boyer and J. Moore, editors, *The Correctness Problem in Computer Science*, pages 215–273. Academic Press, 1981. 1030, 1041

[62] E. W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM J. of Comput.*, 13:441–460, 1984. 1038

[63] R. Mayr. Undecidable problems in unreliable computations. *Theor. Comp. Science*, 297(1-3):337–354, 2003. 1039

[64] A. Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977. 1058

[65] R. Mennicke. Propositional dynamic logic with converse and repeat for message-passing systems. *Logical Methods in Computer Science*, 9(2:12):1–35, 2013. 1052

[66] R. Mennicke. *Model Checking Concurrent Systems using Temporal Logics*. PhD thesis, TU Ilmenau, 2015. 1053

[67] A. Muscholl and D. Peled. Message sequence graphs and decision problems on Mazurkiewicz traces. In *24th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, number 1672 in LNCS, pages 81–91. Springer, 1999. 1050, 1064

[68] A. Muscholl and I. Walukiewicz. Distributed synthesis for acyclic architectures. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 29 of *LIPIcs*, pages 639–651. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014. 1030

[69] T. Neary. Undecidability of binary tag systems and the Post correspondence problem for five pairs of words. In *32nd Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 30 of *LIPIcs*, pages 649–661. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. 1042

[70] E. Ochmanski. Regular behaviour of concurrent systems. *Bulletin of the EATCS*, 27:56–67, 1985. 1058

[71] J. K. Pachl. Reachability problems for communicating finite state machines. Research Report CS-82-12, Dep. of Comp. Sci., Univ. of Waterloo, 1982. see also `http://arxiv.org/abs/cs/0306121`. 1036

[72] D. Peled. Specification and verification of message sequence charts. In *Formal Techniques for Distributed System Development (FORTE/PSTV)*, number 183 in IFIP Conference Proceedings, pages 139–154. Kluwer, 2000. 1047, 1048

[73] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *31st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 746–757, 1990. 1030

[74] E. Post. Formal reductions of the combinatorial decision problem. *Amer. J. of Math.*, 65(2):197–215, 1943. 1030

[75] Ph. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Inform. Proc. Lett.*, 83(5):251–261, 2002. 1031, 1039

[76] P. S. Thiagarajan and I. Walukiewicz. An expressively complete linear time temporal logic for Mazurkiewicz traces. In *12th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 183–194. IEEE Computer Society Press, 1997. 1058

[77] W. Thomas. On logical definability of trace languages. In *Proceedings of a workshop of the ESPRIT BRA No 3166: Algebraic and Syntactic Methods in Computer Science (ASMICS)*, Report TUM-I9002, Technical University of Munich, pages 172–182, 1990. 1058

[78] W. Thomas. On logics, tilings, and automata. In *18th International Colloquium on Automata, Languages and Programming (ICALP)*, number 510 in LNCS, pages 441–454. Springer, 1991. 1056

[79] W. Thomas and W. Zielonka. Logical definability of trace languages. Unpublished manuscript, 1990. 1061

[80] M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Inf. and Comput.*, 115(1):1–37, 1994. 1052

[81] W. Zielonka. Notes on finite asynchronous automata. *RAIRO - Informatique Théorique et Applications*, 21:99–135, 1987. 1058

Index

B-bounded channel, 1048

automaton
- asynchronous, 1058
- communicating, 1032
- Zielonka, 1058

bounded channel, 1048

CFM, 1035
- insertion, 1039
- local acceptance, 1054
- lossy, 1038
- model checking, 1048

channel, 1032
- B-bounded, 1048
- bounded, 1048
- fifo, 1032

channel system
- lossy, 1038

channel system
- insertion, 1039

communicating automata, 1032
CQDD, 1037

decision diagram
- queue-content
  - constraint, 1037

decision diagram
- queue-content, 1036

linearization, 1043
logic
- monadic second-order, 1045
- propositional dynamic, 1047
- temporal
  - linear-time, 1041

LTL, 1041

machine
- communicating finite-state
  - lossy, 1038
- communicating finite-state, 1035
- counter
  - lossy, 1039

Mazurkiewicz trace, 1058
message sequence chart, 1043
model checking CFMs, 1048
MSC, 1043
MSO, 1045

network
- communication, 1032

PDL, 1047
- global formula, 1047
- local formula, 1047
- path expression, 1047

problem
- boundedness $\sim$ for CFMs, 1035
- model-checking $\sim$ for CFMs, 1035
- reachability $\sim$ for CFMs
  - control-state, 1035
- reachability $\sim$ for CFMs, 1035
- reachability $\sim$ for CFMs
  - existentially bounded, 1049
  - universally bounded, 1049

QDD, 1036

realizability, 1053
- specification
  - MSO, 1055
  - PDL, 1056
  - sequential, 1054

transition system
- well-structured, 1038

word
- valid, 1041

WSTS, 1038

Zielonka automaton, 1058