

PROPOSITIONAL DYNAMIC LOGIC FOR MESSAGE-PASSING SYSTEMS

BENEDIKT BOLLIG^a, DIETRICH KUSKE^b, AND INGMAR MEINECKE^c

^a LSV, ENS Cachan, CNRS, France
e-mail address: bollig@lsv.ens-cachan.fr

^b LaBRI Université de Bordeaux and CNRS, France
e-mail address: dietrich.kuske@tu-ilmenau.de

^c Institut für Informatik, Universität Leipzig, Germany
e-mail address: meinecke@informatik.uni-leipzig.de

ABSTRACT. We examine a bidirectional propositional dynamic logic (PDL) for finite and infinite message sequence charts (MSCs) extending LTL and TLC^- . By this kind of multi-modal logic we can express properties both in the entire future and in the past of an event. Path expressions strengthen the classical until operator of temporal logic. For every formula defining an MSC language, we construct a communicating finite-state machine (CFM) accepting the same language. The CFM obtained has size exponential in the size of the formula. This synthesis problem is solved in full generality, i.e., also for MSCs with unbounded channels. The model checking problem for CFMs and HMSCs turns out to be in PSPACE for existentially bounded MSCs. Finally, we show that, for PDL with intersection, the semantics of a formula cannot be captured by a CFM anymore.

1. INTRODUCTION

To make a system accessible to formal analysis and verification techniques, we require it to be modeled mathematically. In this regard, automata-based models have been widely used to describe the behavior of a system under consideration. A natural model for finite processes that exchange messages via FIFO-channels are communicating finite-state machines (CFMs) [BZ83]. In a CFM, each process is modeled as a finite automaton that performs send and receive actions and, in doing so, exchanges messages with other processes via order-preserving communication channels. One single run of a CFM can be described by a message sequence chart (MSC). MSCs are an important common notation in telecommunication and are defined by an ITU standard [ITU96]. An MSC has both a formal definition and a comprehensible visualization.

1998 ACM Subject Classification: F.3.1.

Key words and phrases: message sequence charts, communicating finite-state machines, propositional dynamic logic.

^a Work was partly supported by the projects ANR-06-SETI-003 DOTS and ARCUS Île de France-Inde.

^c Ingmar Meinecke was supported by the German Research Foundation (DFG).

Once we have an automata model \mathcal{A} of a system defining a set $L(\mathcal{A})$ of possible executions, the next task might be to check if it satisfies a requirements specification φ , which represents a set $L(\varphi)$ of (desired) behaviors. Verification now amounts to the *model-checking question*: do all possible behaviors of \mathcal{A} satisfy φ , i.e., do we have $L(\mathcal{A}) \subseteq L(\varphi)$? Many concrete instances of that problem have been considered in the literature [CGP00]. The original, and most popular, ones are finite automata, Kripke structures, or Büchi automata as system model, and temporal logics such as LTL [Pnu77] and CTL [CE81] as specification language. It is well-known that, for all these choices, the corresponding model-checking problem is decidable.

When we move to the setting of CFMs, which, due to a priori unbounded channels, induce infinite-state systems, model checking becomes undecidable. Meenakshi and Rammanujam [MR04] showed undecidability even for very restrictive temporal logics (their results transfer easily from Lamport diagrams to MSCs). One solution is to put a bound on the channel capacity. In other words, the domain of behaviors is restricted to *existentially B -bounded* MSCs, which can be executed without exceeding a fixed channel bound B . In [Pel00, MM01, GMSZ02, GKM06], the model-checking problem was indeed tackled successfully for several logics by using this restriction and following the automata-theoretic approach: (1) a formula φ from a temporal logic or monadic second-order logic is translated into a machine model $\mathcal{A}_{\varphi,B}$ that recognizes those models of φ that are existentially B -bounded; (2) it is checked whether every existentially B -bounded behavior of the system model is contained in the language of $\mathcal{A}_{\varphi,B}$.

But on the other hand, we may apply temporal logic in the early stages of system development and start with specifying formulas to exemplify the intended interaction of the system to be. If so, we would like to synthesize a system model from a formula that captures precisely those behaviors that satisfy the formula. In other words, we ask whether a temporal-logic formula is realizable, i.e., whether the derived system is consistent and shows any reasonable behavior at all. Once a system is synthesized directly from its specification, it can be assumed to be correct a priori, provided the translation preserves the semantics of the specification.

Though the assumption of bounded channels leads to the decidability of the model checking problem, it does not seem natural to restrict the channel size of the desired system in advance, especially when one is interested in the synthesis of a system from a specification. Despite the complexity of MSCs, we will provide in this paper a linear-time temporal logic for message-passing systems and solve its realizability problem in its full generality, i.e., under the assumption of a priori unbounded channels.

Results from [BL06, BK08] suggest to use an existential fragment of monadic second-order logic (EMSO) as a specification language. A formula from that fragment can be translated into a CFM that *precisely* recognizes the models of the formula. This result holds without channel restriction. In this paper, we basically follow the approach from [BL06, BK08], but we propose a new logic: propositional dynamic logic (PDL) for MSCs. Our logic will prove useful for verification, as it is closed under negation and allows us to express interesting properties in an easy and intuitive manner. Like EMSO, but unlike full monadic second-order logic, every PDL formula φ can be effectively translated into a CFM \mathcal{A}_{φ} whose language is the set of models of φ . This synthesis step is independent of any channel bound B and would not become simpler if we took some B into account. The size of the resulting CFM is exponential in the size of φ and in the number of processes. Note that, by [BL06, BK08], EMSO is expressively equivalent to CFMs. Moreover, the set of

CFM languages is not closed under complementation. As, on the other hand, PDL does not impose any restriction on the use of negation, we obtain that PDL is a proper fragment of EMSO although this is not obvious.

The model checking problem of CFMs against PDL formulas can be decided in polynomial space for existentially B -bounded MSCs, following a standard procedure and using the translation of φ into a CFM \mathcal{A}_φ . Our PSPACE algorithm meets the lower bound that is imposed by the complexity of LTL model checking for finite-state systems. We also show PSPACE completeness of the model checking problem of high-level MSCs (HMSCs) against PDL formulas (where the bound B is given implicitly by the HMSC). HMSCs are more abstract and restrictive than CFMs, but can likewise be used as a model of a system.

The final technical section considers an enriched logic: iPDL (PDL with intersection). This extension seems natural to strengthen the expressive power of the formulas. But adapting a proof technique from colored grids, we show that iPDL is too strong for CFMs, i.e., there is an iPDL formula φ such that no CFM accepts precisely the models of φ .

Related Work. For MSCs, there exist a few attempts to define suitable temporal logics. Meenakshi and Ramanujam obtained exponential-time decision procedures for several temporal logics over Lamport diagrams (which are similar to MSCs) [MR00, MR04]. Peled [Pel00] considered the fragment TLC^- of the temporal logic TLC that was introduced in [APP95]. Like their logics, our logic is interpreted directly over MSCs, not over linearizations; it combines elements from [MR04] (global next operator, past operators) and [Pel00] (global next operator, existential interpretation of the until-operator). In particular, however, it is inspired by *dynamic* LTL as introduced by Henriksen and Thiagarajan first for words [HT99]. There, standard LTL is extended by indexing the until operator with a regular expression to make it more expressive. The same authors applied dynamic LTL also to Mazurkiewicz traces but reasoned only about the future of an event in the same process [HT97]. In contrast, we might argue about the whole future of an event rather than about one single process. Moreover, we provide past operators to judge about events that have already been executed. We call our logic PDL because it is essentially the original propositional dynamic logic as first defined by Fischer and Ladner [FL79] but here in the framework of MSCs. Although PDL can be seen as an extension of Peled’s TLC^- , our decision procedure is rather different. Instead of translating a PDL formula φ into a CFM directly, we use an inductive method inspired by [GK03, GK07, GK10]. As TLC^- is a fragment of PDL, we actually generalize the model checking result from [Pel00].

Outline. In Section 2, we define message sequence charts, the logic PDL, and CFMs. We continue, in Section 3, with several useful constructions for CFMs. Sections 4 and 5 deal with the translation of PDL formulas into CFMs. The model checking problem is tackled in Section 6 before we conclude, in Section 7, with the result that PDL with intersection (iPDL) cannot be implemented in terms of CFMs.

A preliminary version of this paper appeared as [BKM07].

2. DEFINITIONS

The communication framework used in our paper is based on sequential processes that exchange messages asynchronously over point-to-point, error-free FIFO channels. Let \mathcal{P} be a finite set of process identities which we fix throughout this paper. Furthermore, let $\text{Ch} =$

$\{(p, q) \in \mathcal{P}^2 \mid p \neq q\}$ denote the set of *channels*. Processes act by either sending a message, that is denoted by $p!q$ meaning that process p sends to process q , or by receiving a message, that is denoted by $p?q$, meaning that process p receives from process q . For any process $p \in \mathcal{P}$, we define a local alphabet (set of event types on p) $\Sigma_p = \{p!q, p?q \mid q \in \mathcal{P} \setminus \{p\}\}$, and we set $\Sigma = \bigcup_{p \in \mathcal{P}} \Sigma_p$.

2.1. Message sequence charts. A message sequence chart depicts processes as vertical lines, which are interpreted as top-down time axes. Moreover, an arrow from one line to a second corresponds to the communication events of sending and receiving a message. Formally, message sequence charts are special labeled partial orders. To define them, we need the following definitions: A Σ -labeled partial order is a triple $M = (V, \leq, \lambda)$ where (V, \leq) is a partially ordered set and $\lambda : V \rightarrow \Sigma$ is a mapping. For $v \in V$ with $\lambda(v) = p\theta q$ where $\theta \in \{!, ?\}$, let $P(v) = p$ denote the process that v is located at. We set $V_p = P^{-1}(p)$. We define two binary relations *proc* and *msg* on V :

- $(v, v') \in \text{proc}$ iff $P(v) = P(v')$, $v < v'$, and, for any $u \in V$ with $P(v) = P(u)$ and $v \leq u < v'$, we have $v = u$. The idea is that $(v, v') \in \text{proc}$ whenever v and v' are two consecutive events of the same process.
- $(v, v') \in \text{msg}$ iff there is a channel (p, q) with $\lambda(v) = p!q$, $\lambda(v') = q?p$, and

$$|\{u \mid \lambda(u) = p!q, u \leq v\}| = |\{u \mid \lambda(u) = q?p, u \leq v'\}|.$$

Here, the idea is that v is a send event and v' is the matching receive event. Since we model reliable FIFO-channels, this means that, for some i and some channel (p, q) , v is the i^{th} send and v' the i^{th} receive event on channel (p, q) .

Definition 2.1. A *message sequence chart* or *MSC* for short is a Σ -labeled partial order (V, \leq, λ) such that

- $\leq = (\text{proc} \cup \text{msg})^*$,
- $\{u \in V \mid u \leq v\}$ is finite for any $v \in V$,
- V_p is linearly ordered for any $p \in \mathcal{P}$, and
- $|\lambda^{-1}(p!q)| = |\lambda^{-1}(q?p)|$ for any $(p, q) \in \text{Ch}$.

We refer to the elements of V as *events* or *nodes*.

If (V, \leq, λ) is an MSC, then *proc* and *msg* are even injective partial functions, so $v' = \text{proc}(v)$ as well as $v = \text{proc}^{-1}(v')$ are equivalent notions for $(v, v') \in \text{proc}$; $\text{msg}(v)$ and $\text{msg}^{-1}(v)$ are to be understood similarly.

An example MSC with three processes is pictured as a diagram in Figure 1(b) on page 7. The processes are visualized as vertical lines going downwards and messages as horizontal directed edges between process lines.

2.2. Propositional dynamic logic. *Path expressions* π and *local formulas* α are defined by simultaneous induction. This induction is described by the following rules

$$\begin{aligned} \pi &::= \text{proc} \mid \text{msg} \mid \{\alpha\} \mid \pi; \pi \mid \pi + \pi \mid \pi^* \\ \alpha &::= \# \mid \sigma \mid \alpha \vee \alpha \mid \neg \alpha \mid \langle \pi \rangle \alpha \mid \langle \pi \rangle^{-1} \alpha \end{aligned}$$

where σ ranges over the alphabet Σ .

Local formulas express properties of single nodes in MSCs. To define the semantics of local formulas, let therefore $M = (V, \leq, \lambda)$ be an MSC and v a node from M . Then we define

$$\begin{aligned} M, v \models \sigma &\iff \lambda(v) = \sigma \text{ for } \sigma \in \Sigma \\ M, v \models \alpha_1 \vee \alpha_2 &\iff M, v \models \alpha_1 \text{ or } M, v \models \alpha_2 \\ M, v \models \neg\alpha &\iff M, v \not\models \alpha \end{aligned}$$

The idea of forward-path modalities $\langle \pi \rangle \alpha$ indexed by a path expression π is to perform a program π and then to check whether α is satisfied. Thereby, π is a rational expression over proc and msg describing paths in a MSC but allows also for tests $\{\alpha\}$ in following the paths defined by π . Formally, the semantics of *forward-path* formulas $\langle \pi \rangle \alpha$ is given by

$$\begin{aligned} M, v \models \langle \text{proc} \rangle \alpha &\iff \text{there exists } v' \in V \text{ with } (v, v') \in \text{proc} \text{ and } M, v' \models \alpha \\ M, v \models \langle \text{msg} \rangle \alpha &\iff \text{there exists } v' \in V \text{ with } (v, v') \in \text{msg} \text{ and } M, v' \models \alpha \\ M, v \models \langle \{\alpha\} \rangle \beta &\iff M, v \models \alpha \text{ and } M, v \models \beta \\ M, v \models \langle \pi_1; \pi_2 \rangle \alpha &\iff M, v \models \langle \pi_1 \rangle \langle \pi_2 \rangle \alpha \\ M, v \models \langle \pi_1 + \pi_2 \rangle \alpha &\iff M, v \models \langle \pi_1 \rangle \alpha \vee \langle \pi_2 \rangle \alpha \\ M, v \models \langle \pi^* \rangle \alpha &\iff \text{there exists } n \geq 0 \text{ with } M, v \models (\langle \pi \rangle)^n \alpha \end{aligned}$$

The semantics of *backward-path* formulas $\langle \pi \rangle^{-1} \alpha$ is defined similarly:

$$\begin{aligned} M, v \models \langle \text{proc} \rangle^{-1} \alpha &\iff \text{there exists } v' \in V \text{ with } (v', v) \in \text{proc} \text{ and } M, v' \models \alpha \\ M, v \models \langle \text{msg} \rangle^{-1} \alpha &\iff \text{there exists } v' \in V \text{ with } (v', v) \in \text{msg} \text{ and } M, v' \models \alpha \\ M, v \models \langle \{\alpha\} \rangle^{-1} \beta &\iff M, v \models \alpha \text{ and } M, v \models \beta \\ M, v \models \langle \pi_1; \pi_2 \rangle^{-1} \alpha &\iff M, v \models \langle \pi_1 \rangle^{-1} \langle \pi_2 \rangle^{-1} \alpha \\ M, v \models \langle \pi_1 + \pi_2 \rangle^{-1} \alpha &\iff M, v \models \langle \pi_1 \rangle^{-1} \alpha \vee \langle \pi_2 \rangle^{-1} \alpha \\ M, v \models \langle \pi^* \rangle^{-1} \alpha &\iff \text{there exists } n \geq 0 \text{ with } M, v \models (\langle \pi \rangle^{-1})^n \alpha \end{aligned}$$

Semantically, a local formula of the form $\langle \{\alpha\}; (\text{proc} + \text{msg})^* \rangle \beta$ corresponds to the until construct $\alpha \mathcal{U} \beta$ in Peled's TLC^- [Pel00]. In TLC^- , however, one cannot express properties such as “there is an even number of messages from p to q ”, which is easily expressible in PDL.

Global properties of an MSC are Boolean combinations of properties of the form “there exists a node satisfying the local formula α ”. These global properties are expressed by *global formulas* φ whose syntax is given by

$$\varphi ::= E\alpha \mid A\alpha \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$$

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

$$\begin{aligned} M \models E\alpha &\iff \text{there exists a node } v \text{ with } M, v \models \alpha \\ M \models A\alpha &\iff M, v \models \alpha \text{ for all nodes } v \\ M \models \varphi_1 \vee \varphi_2 &\iff M \models \varphi_1 \text{ or } M \models \varphi_2 \\ M \models \varphi_1 \wedge \varphi_2 &\iff M \models \varphi_1 \text{ and } M \models \varphi_2 \end{aligned}$$

Note that our syntax of global formulas does not allow explicit negation. But since we allow existential and universal quantification as well as disjunction and conjunction, the expressible properties are closed under negation.

Example 2.2. For $i \in \mathcal{P}$, we put $P_i = \bigvee_{j \in \mathcal{P}, j \neq i} (i!j \vee i?j)$, i.e., $M, v \models P_i$ iff $P(v) = i$ for every MSC $M = (V, \leq, \lambda)$ and $v \in V$. Now the global formula

$$\varphi_{i,j}^{\overline{=2}} = A(P_i \longrightarrow (\langle \text{proc}^*; \text{msg}; \text{proc}^*; \text{msg} \rangle P_j))$$

states that process j can always be reached from process i with exactly two messages (using an intermediate process in between).

Definition 2.3. The set of subformulas $\text{sub}(\alpha)$ of a local formula α and the set of subformulas $\text{sub}(\pi)$ of a path expression π are defined by synchronous induction as follows:

$$\begin{aligned} \text{sub}(\text{proc}) &= \text{sub}(\text{msg}) = \emptyset \\ \text{sub}(\{\alpha\}) &= \text{sub}(\alpha) \\ \text{sub}(\pi_1; \pi_2) &= \text{sub}(\pi_1 + \pi_2) = \text{sub}(\pi_1) \cup \text{sub}(\pi_2) \\ \text{sub}(\pi^*) &= \text{sub}(\pi) \end{aligned}$$

and

$$\begin{aligned} \text{sub}(\sigma) &= \{\sigma\} \text{ for } \sigma \in \Sigma \\ \text{sub}(\neg\alpha) &= \{\neg\alpha\} \cup \text{sub}(\alpha) \\ \text{sub}(\alpha \vee \beta) &= \{\alpha \vee \beta\} \cup \text{sub}(\alpha) \cup \text{sub}(\beta) \\ \text{sub}(\langle \pi \rangle \alpha) &= \{\langle \pi \rangle \alpha\} \cup \text{sub}(\pi) \cup \text{sub}(\alpha) \\ \text{sub}(\langle \pi \rangle^{-1} \alpha) &= \{\langle \pi \rangle^{-1} \alpha\} \cup \text{sub}(\pi) \cup \text{sub}(\alpha) \end{aligned}$$

Thus, in addition to the obvious definition, a subformula of a path expression is any of the local formulas occurring in the path expression as well as any subformula of these local formulas. In particular, contrary to what one might expect, a rather long local formula like $\varphi = \langle \text{proc}; \{\sigma\}; \text{proc}; \{\sigma\}; \text{proc}; \{\sigma\}; \text{proc}; \{\sigma\} \rangle \sigma$ has only two subformulas, namely φ itself and σ . The number of subformulas of α is bounded by the length of α , but the length of α cannot be bounded in terms of the number of subformulas.

Note that a path expression π is a regular expression over the following alphabet $\{\text{proc}, \text{msg}, \{\alpha_1\}, \dots, \{\alpha_n\}\}$ for some local formulas α_i . The *size* $s(\pi)$ of π is defined by $s(\{\alpha\}) = s(\text{proc}) = s(\text{msg}) = 1$, $s(\pi_1 + \pi_2) = s(\pi_1; \pi_2) = s(\pi_1) + s(\pi_2)$ and $s(\pi^*) = s(\pi)$ (i.e., it is the number of occurrences of $\{\alpha\}$, msg , and proc in the regular expression π). Note that the size of the path expression $\{\alpha\}$ is 1, independent from the concrete form of the local formula α .

2.3. Communicating finite-state machines. One formalism to describe (asynchronous) communication protocols are *communicating finite-state machines* (CFM for short) [BZ83]. They form a basic model for distributed algorithms based on asynchronous message passing between concurrent processes. Thus, the basic actions performed are just sending and receiving of messages (i.e., letters from Σ).

A CFM \mathcal{A} consists of a collection of finite automata \mathcal{A}_p , one for each process $p \in \mathcal{P}$. The automaton \mathcal{A}_p performs the actions of process p , i.e., the send events $p!q$ and the receive events $p?q$ for all $q \neq p$. Moreover, the single automata synchronize by control messages

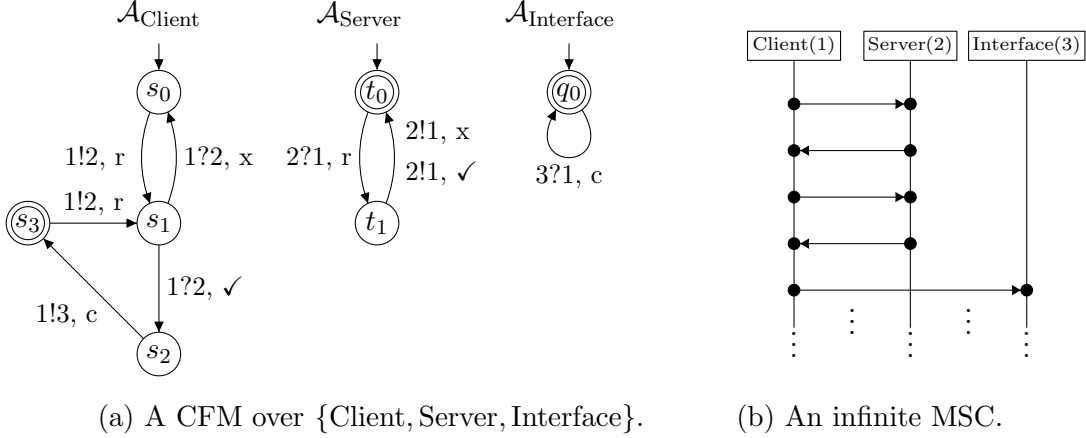


Figure 1: A CFM and an infinite MSC accepted by it.

from some finite set C . Whenever \mathcal{A}_p sends a message to \mathcal{A}_q , then \mathcal{A}_p and \mathcal{A}_q share some common control message $c \in C$. The final states of a CFM are defined globally and the local components of a final state have either to be repeated infinitely often by the process or the process terminates in such a local state.

We extend the alphabet Σ for later purposes to $\Sigma \times \{0, 1\}^n$ for some $n \in \mathbb{N}$ – the classical model is obtained by setting $n = 0$ in the below definition (in which case we write $\mathcal{A} = (C, (\mathcal{A}_p)_{p \in \mathcal{P}}, F)$.)

Definition 2.4. A *communicating finite-state machine* (or, simply, *CFM*) is a structure $\mathcal{A} = (C, n, (\mathcal{A}_p)_{p \in \mathcal{P}}, F)$ with $n \in \mathbb{N}$ where

- C is a finite set of *message contents* or *control messages*,
- $\mathcal{A}_p = (S_p, \rightarrow_p, \iota_p)$ is a finite labeled transition system over the alphabet $\Sigma_p \times \{0, 1\}^n \times C$ for any $p \in \mathcal{P}$ (i.e., $\rightarrow_p \subseteq S_p \times (\Sigma_p \times \{0, 1\}^n \times C) \times S_p$) with initial state $\iota_p \in S_p$,
- $F \subseteq \prod_{p \in \mathcal{P}} S_p$ is a set of global final states.

Now let \mathcal{A} be a CFM as above, $M = (V, \leq, \lambda)$ be an MSC, and $c : V \rightarrow \{0, 1\}^n$. A *run* of \mathcal{A} on (M, c) is a pair (ρ, μ) of mappings $\rho : V \rightarrow \bigcup_{p \in \mathcal{P}} S_p$ and $\mu : V \rightarrow C$ such that, for any $v \in V$,

- (1) $\mu(v) = \mu(\text{msg}(v))$ if $\text{msg}(v)$ is defined,
- (2) $(\rho(\text{proc}^{-1}(v)), \lambda(v), c(v), \mu(v), \rho(v)) \in \rightarrow_{P(v)}$ if $\text{proc}^{-1}(v)$ is defined, and $(\iota_p, \lambda(v), c(v), \mu(v), \rho(v)) \in \rightarrow_{P(v)}$ otherwise.

In order to define when the run (ρ, μ) is accepting, we will use Büchi-conditions on each process. For this, one is usually interested in the set of states that appear infinitely often. But since, even in an infinite MSC, some of the processes may execute only finitely many events, the set of states appearing infinitely often is here generalized to the set of states that appear *cofinally*: Let $\text{cofin}_\rho(p) = \{s \in S_p \mid \forall v \in V_p \exists v' \in V_p : v \leq v' \wedge \rho(v') = s\}$. Then the run (ρ, μ) is *accepting* if there is some $(s_p)_{p \in \mathcal{P}} \in F$ such that $s_p \in \text{cofin}_\rho(p)$ for all $p \in \mathcal{P}$. The *language* of \mathcal{A} is the set $L(\mathcal{A})$ of all pairs (M, c) that admit an accepting run.

Example 2.5. Consider the CFM illustrated in Figure 1(a). A client (process 1) communicates with a server (process 2) sending requests (message content r) to receive permission

to send a message to the interface (process 3). If the server refuses permission (message content x), the request is repeated. But if permission is given (message content \checkmark), then the client sends a message c to the interface. Now the client can start again to send requests to the server. Here, the only accepting state is (s_3, t_0, q_0) . Thus the client can either stop after sending a message to the interface (and all other processes also stop) or the client has to send infinitely many messages to the interface, i.e., every request of the client is eventually followed by a communication with the interface.

The MSC pictured in Figure 1(b) is one possible behavior of the CFM. Moreover, any MSC M accepted by this CFM satisfies the formula $\varphi_{2,3}^2$ from Example 2.2.

3. CONSTRUCTIONS OF CFMS

In this section, we present some particular CFMs and constructions of CFMs. The purpose is twofold: the results will be used later, and the reader shall become acquainted with the computational power of CFMs. Hence, some readers might choose to skip the details of this section in a first reading.

3.1. Intersection. Here, we show that the intersection of languages accepted by CFMs can again be accepted by a CFM. Since the acceptance by a CFM is defined in terms of a Büchi-condition, we can adopt the flag construction [Cho74] from the theory of word automata with Büchi-acceptance condition (cf. proof of Lemma 1.2 in [Tho90]). The additional problem we face here is the interplay between different processes.

The basic idea of our construction is as follows (for two CFMs \mathcal{A}^1 and \mathcal{A}^2): each local process guesses an accepting global state $f^1 = (f_p^1)_{p \in \mathcal{P}}$ of \mathcal{A}^1 and $f^2 = (f_p^2)_{p \in \mathcal{P}}$ of \mathcal{A}^2 . Then, locally, process p simulates both CFMs \mathcal{A}^1 and \mathcal{A}^2 and checks that f_p^1 and f_p^2 are visited infinitely often (here, one relies on Choueka's flag construction). Hence, the set of local states of process p equals $F^1 \times F^2 \times S_p^1 \times S_p^2 \times \{0, 1, 2\}$. A global state is accepting if all the guesses locally made coincide and if the local processes accept according to the flag construction.

Recall that the set of accepting states F^1 is a set of tuples, its maximal size is therefore $\prod_{p \in \mathcal{P}} |S_p^1|$. Thus, the intersection of two CFMs with s local states per process can result in a CFM with $s^{2|\mathcal{P}|} \cdot s^2 \cdot 3 = s^{O(|\mathcal{P}|)}$ many local states per process.

Now suppose that F^1 and F^2 are direct products, i.e., $F^1 = \prod_{p \in \mathcal{P}} F_p^1$ for some sets $F_p^1 \subseteq S_p^1$ and, similarly, $F^2 = \prod_{p \in \mathcal{P}} F_p^2$ for some sets $F_p^2 \subseteq S_p^2$. Then, in the above construction, it is not necessary for the local guesses to coincide – which makes them superfluous (cf. Proof of Lemma 3.2 below). Thus, in this case, the set of local states of process p will just be $S_p^1 \times S_p^2 \times \{0, 1, 2\}$, in particular, it will not be exponential in the number of processes.

To use this simplification of the construction, we introduce the following notion.

Definition 3.1. Let $F \subseteq \prod_{p \in \mathcal{P}} S_p$. The *index of F* is the least number n such that there are sets $F_p^i \subseteq S_p$ for $p \in \mathcal{P}$ and $1 \leq i \leq n$ with $F = \bigcup_{1 \leq i \leq n} \prod_{p \in \mathcal{P}} F_p^i$.

The *index of a CFM* is the index of its set of accepting states.

Clearly, the index of a CFM is bounded by $s^{|\mathcal{P}|}$ where s is the maximal size of a set of local states S_p . To see that it can indeed be quite large, let $\mathcal{P} = S_p = [n]$ for all $p \in \mathcal{P}$ (where we let $[n] = \{1, \dots, n\}$). Furthermore, let F be the set of all tuples $(s_p)_{p \in \mathcal{P}}$ such that

$\{s_p \mid p \in \mathcal{P}\} = [n]$, i.e., the set of surjections from $[n]$ onto $[n]$. Hence F contains $n!$ many elements. Any two of them differ in at least two positions. Hence the index of F equals its size and is exponential in n and therefore in $|\mathcal{P}|$. Despite this exponential example, we will encounter only small indices in our constructions.

Lemma 3.2. *For $1 \leq i \leq m$, let $\mathcal{A}^i = (C^i, n, (S_p^i, \rightarrow_p^i, \iota_p^i)_{p \in \mathcal{P}}, F^i)$ be CFMs of index 1. Then there exists a CFM \mathcal{A} of index 1 that accepts (M, c) with M an MSC and $c : V \rightarrow \{0, 1\}^n$ iff it is accepted by \mathcal{A}^i for all $i \in [m]$.*

The set of messages of \mathcal{A} is $\prod_{i \in [m]} C^i$ and the set of local states of process p is $\{0, 1, \dots, m\} \times \prod_{i \in [m]} S_p^i$.

Proof. Since F^i has index 1, there exist sets $F_p^i \subseteq S_p^i$ with $F^i = \prod_{p \in \mathcal{P}} F_p^i$.

The idea of the proof is that \mathcal{A} will simulate all the machines \mathcal{A}^i in parallel. In addition, it checks that, for each $p \in \mathcal{P}$ and $i \in [m]$, some state from F_p^i is assumed cofinally (i.e., infinitely often or, if process p executes only finitely many events, at the last event from p). Formally, we set $\iota_p = \begin{cases} (m, \iota_p^1, \dots, \iota_p^m) & \text{if } (\iota_p^1, \dots, \iota_p^m) \in \prod_{i \in [m]} F_p^i \text{ and} \\ (0, \iota_p^1, \dots, \iota_p^m) & \text{otherwise} \end{cases}$

$F = \prod_{p \in \mathcal{P}} (\{m\} \times \prod_{i \in [m]} S_p^i)$. Furthermore, $(a, (s_i)_{i \in [m]}) \xrightarrow{\sigma, c, (b_i)_{i \in [m]}}_p (a', (s'_i)_{i \in [m]})$ with $b_i \in C^i$ is a transition of \mathcal{A} iff

- (1) $s_i \xrightarrow{\sigma, c, b_i}_p^i s'_i$ is a transition of \mathcal{A}^i for all $i \in [m]$
- (2) $a' = \begin{cases} m & \text{if } s_i \in F_p^i \text{ for all } i \in [m] \\ 0 & \text{if } a = m \text{ and } s_i \notin F_p^i \text{ for some } i \in [m] \\ a + 1 & \text{if } a < m, s_{a+1} \in F_p^{a+1} \text{ and } s_i \notin F_p^i \text{ for some } i \in [m] \\ a & \text{otherwise.} \end{cases}$

Recall the classical flag construction for ω -word automata. There, the value of the counter a indicates that the composite machine waits for an accepting state of the simulated machine $a + 1$; a value m indicates that all simulated machines went through some accepting states. Here, we do the same. But, in addition, if all component states of the composite machine are accepting, then we set the counter value directly to m . This is useful when process p executes only finitely many events. Then, at its final event v , all the component machines have to be in some accepting state. For processes executing infinitely many events, this is of no importance. \square

Proposition 3.3. *For $i \in [m]$, let $\mathcal{A}^i = (C^i, n, (S_p^i, \rightarrow_p^i, \iota_p^i)_{p \in \mathcal{P}}, F^i)$ be a CFM of index ℓ_i . Then there exists a CFM \mathcal{A} of index $\prod_{1 \leq i \leq m} \ell_i$ that accepts (M, c) with M an MSC and $c : V \rightarrow \{0, 1\}^n$ iff it is accepted by \mathcal{A}^i for all $i \in [m]$.*

The set of messages of \mathcal{A} is $\prod_{i \in [m]} C^i$. Moreover, the set of local states of process p is $\{\iota_p\} \cup (\{0, 1, 2, \dots, m\} \times \prod_{i \in [m]} S_p^i \times \prod_{i \in [m]} [\ell_i])$.

Proof. Since the index of \mathcal{A}^i is ℓ_i , its language is the union of languages $L_1^i, \dots, L_{\ell_i}^i$ that can each be accepted by a CFM of index 1. The language in question is therefore given by $\bigcup_{j \in \prod_{i \in [m]} [\ell_i]} \bigcap_{i \in [m]} L_{j_i}^i$. By Lemma 3.2, the intersection $\bigcap_{i \in [m]} L_{j_i}^i$ can be accepted by a CFM of index 1 with set of local states $\{0, 1, 2, \dots, m\} \times \prod_{i \in [m]} S_p^i \times \{j\}$. The disjoint union of all these CFMs (together with new local initial states) accepts the language in question;

its set of local states equals $\{\iota_p\} \cup (\{0, 1, 2, \dots, m\} \times \prod_{i \in [m]} S_p^i \times \prod_{i \in [m]} [\ell_i])$ and its index is $\prod_{1 \leq i \leq m} \ell_i$ as claimed. \square

3.2. Infinitely running processes. For an MSC $M = (V, \leq, \lambda)$, let $\text{Inf}(M) \subseteq \text{Ch}$ denote the set of those channels (p, q) that are used infinitely often, i.e., $\text{Inf}(M) = \{(p, q) \in \text{Ch} \mid \lambda^{-1}(p!q) \text{ is infinite}\}$. From a set $I \subseteq \text{Ch}$, we want to construct a CFM of index 1 that checks whether $\text{Inf}(M) = I$.

Lemma 3.4. *Let $I \subseteq \text{Ch}$. There exists a CFM \mathcal{A}_1 of index 1 with three local states per process and one message that accepts an MSC M iff $\text{Inf}(M) \subseteq I$.*

Proof. The sets of local states are given by $S_p = \{0, 1, 2\}$ for any $p \in \mathcal{P}$, the state 0 is locally initial. The only control message is 1. Then we set $a \xrightarrow{\sigma, 1}_p b$ iff $(a = b \text{ and } \sigma \text{ uses a channel from } I) \text{ or } (a < b \text{ and } \sigma \text{ does not use a channel from } I) \text{ or } a = b = 1$. Then state 0 indicates that no channel of $\text{Ch} \setminus I$ has been used, 1 indicates that some channel from $\text{Ch} \setminus I$ has been used and that some channel will be used, and 2 denotes that some channel from $\text{Ch} \setminus I$ has been used but none will ever be used in the future. Hence, process p uses the channels from $\text{Ch} \setminus I$ only finitely often iff it can visit 0 or 2 cofinally. Setting $F = \prod_{p \in \mathcal{P}} \{0, 2\}$ therefore finishes the construction of the desired CFM. \square

Lemma 3.5. *Let $I \subseteq \text{Ch}$. There exists a CFM \mathcal{B}_1 of index 1 with $4^{|\mathcal{P}|}$ local states per process and one message that accepts an MSC M iff $I \subseteq \text{Inf}(M)$.*

Proof. For $p \in \mathcal{P}$ let $S_p = \{0, 1\}^{\Sigma_p}$ and set $C = \{1\}$. The locally initial state $\iota_p \in S_p$ maps all $\tau \in \Sigma_p$ to 0. Then we set $g \xrightarrow{\sigma, 1} g'$ for $g, g' \in S_p$ and $\sigma \in \Sigma_p$ iff $g'(\tau) = \begin{cases} g(\tau) & \text{if } \tau \neq \sigma \\ 1 - g(\tau) & \text{otherwise} \end{cases}$ for all $\tau \in \Sigma_p$. Thus, the local process p counts modulo 2 the number of occurrences of any local action. The channel (p, q) is used infinitely often iff the following two properties hold:

- Process p visits a state g_p with $g_p(p!q) = 0$ cofinally.
- Process q visits a state g_q with $g_q(q?p) = 1$ cofinally.

Therefore, a global state $(g_p)_{p \in \mathcal{P}}$ is final (i.e., belongs to F) iff, for any $(p, q) \in I$, we have $g_p(p!q) = 0$ and $g_q(q?p) = 1$. \square

Proposition 3.6. *Let $I \subseteq \text{Ch}$. There exists a CFM \mathcal{B} of index 1 with $3 \cdot 3 \cdot 4^{|\mathcal{P}|}$ local states per process and one message that accepts an MSC M iff $I = \text{Inf}(M)$.*

Proof. Follows immediately from Lemmas 3.4, 3.5, and 3.2. \square

3.3. The color language. In this section, we build a CFM that accepts some “black/white colored” MSCs. The aim is that whenever a coloring is accepted, then any infinite path in the MSC has infinitely many color changes (cf. Cor. 3.9). This language will be the crucial ingredient in our handling of forward-path formulas of the form $\langle \pi \rangle \alpha$ (cf. Section 4.2).

For the time being, we proceed as follows: first, we define a language Col whose elements are colored MSCs (M, c) . Prop. 3.7 shows that this language can be accepted by a CFM. Cor. 3.9 ensures that any infinite path in $(M, c) \in \text{Col}$ has infinitely many color changes. We do not prove the converse (which is actually false), but will see later that sufficiently many colorings with this property belong to Col (Lemma 4.14).

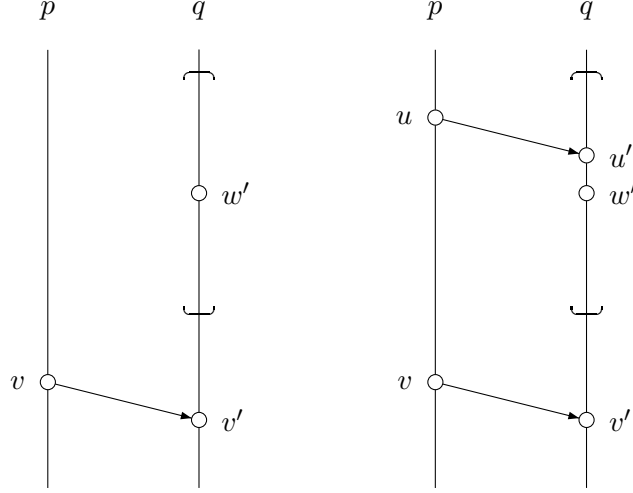


Figure 2: The second condition.

Let M be an MSC and $c : V \rightarrow \{0, 1\}$. On V , we define an equivalence relation \sim setting $u \sim w$ iff $P(u) = P(w)$ and, for all $v \in V$ with $(u \leq v \leq w$ or $w \leq v \leq u)$ and $P(u) = P(v)$, we have $c(u) = c(v) = c(w)$ (i.e., a \sim -equivalence class is a maximal monochromatic interval on a process line).

Let Col be the set of all pairs (M, c) with $c : V \rightarrow \{0, 1\}$ such that the following hold

- (1) if v is minimal on its process, then $c(v) = 1$,
- (2) if $(v, v') \in \text{msg}$ and $w' \leq v'$ with $P(w') = P(v')$, then there exists $(u, u') \in \text{msg}$ with $\lambda(u') = \lambda(v')$, $c(u) = c(u')$, and $u' \sim w'$ (implying $\lambda(u) = \lambda(v)$),
- (3) any equivalence class of \sim is finite.

Figure 2 visualizes the second condition, on the left, we have the precondition while the right diagram indicates the conclusion. More precisely, in the precondition, we have a message $(v, v') \in \text{msg}$ from process p to process q and some node w' preceding v' on the same process. Recall that equivalence classes of \sim are intervals on process lines. The borders of the equivalence class containing w' are indicated. Then, by the conclusion, there is a message $(u, u') \in \text{msg}$ from p to q such that u' belongs to the indicated equivalence class of \sim (that also contains w') and the colors of u and u' are the same (which is not indicated).

In general, there can be messages $(u, u') \in \text{msg}$ such that the colors of u and u' are different, i.e., $c(u) \neq c(u')$. The second condition ensures that there are “many” messages where the send and the receive event carry the same color.

Proposition 3.7. *There exists a CFM \mathcal{A}_{Col} that accepts the set Col . The CFM \mathcal{A}_{Col} has two messages and its number of local states is in $2^{O(|\mathcal{P}|)}$.*

Proof. Since the language Col consists of pairs (M, c) , any process p of a CFM with two messages executes a sequence of events from $((\Sigma_p \times \{0, 1\}) \times \{0, 1\})^\infty$ (with Γ^∞ the set of finite and infinite words over Γ) where $((\sigma, a), b)$ stands for a (σ, a) -labeled event that sends or receives b . Our automaton \mathcal{A}_{Col} will always send the current value of the mapping c , i.e., the set of control messages is $\{0, 1\}$ and we will only execute events from

$$\Gamma_p = \{((p!q, a), a) \mid q \in \mathcal{P} \setminus \{p\}, a \in \{0, 1\}\} \cup \{((p?q, a), b) \mid q \in \mathcal{P} \setminus \{p\}, a, b \in \{0, 1\}\}.$$

Having this in mind, consider for $p \in \mathcal{P}$, $B \subseteq \mathcal{P}$, and $i \in \{0, 1\}$ the language $L_{B,i}^p \subseteq \Gamma_p^*$ with $w \in L_{B,i}^p$ iff

- if $((p!q, a), a)$ occurs in w , then $a = i$,
- if $((p?q, a), b)$ occurs in w , then $a = i$ and $q \in B$,
- for all $q \in B$, the letter $((p?q, i), i)$ occurs in w .

Then $L_{B,i}^p$ is regular and can be accepted by a finite deterministic automaton $\mathcal{B}_{B,i}^p$ with $2^{|B|}$ many states. We build the p -component \mathcal{A}_p of \mathcal{A}_{Col} from the disjoint union of all these automata $\mathcal{B}_{B,i}^p$ – it therefore has

$$\sum_{B \subseteq \mathcal{P}} 2 \cdot 2^{|B|} \leq 2 \cdot 4^{|\mathcal{P}|}$$

many states. More precisely, \mathcal{A}_p is obtained from this disjoint union by adding ε -transitions from any accepting state of $\mathcal{A}_{B,i}^p$ to any initial state of $\mathcal{A}_{C,j}^p$ iff $B \supseteq C$ and $i \neq j$. The initial states of \mathcal{A}_p are the initial states of $\mathcal{B}_{B,1}^p$. A finite run is accepting if it ends in some final state of one of the automata $\mathcal{B}_{B,i}^p$, an infinite run is accepting if it takes infinitely many ε -transitions.

Note that a word $((p\theta_n q_n, a_n), b_n)_{0 \leq n < N} \in \Gamma_p^\infty$ is accepted by \mathcal{A}_p iff

- $a_0 = 1$
- if $\theta_n = !$, then $a_n = b_n$
- if $\theta_n = ?$ and $m \leq n$, then there exists $k \in \mathbb{N}$ with $p\theta_k q_k = p?q_n$, $a_k = b_k$, and, for all ℓ in between m and k , we have $a_m = a_\ell = a_k$.

Hence the CFM consisting of these components accepts the language Col. \square

The *index* $\text{ind}(v)$ of a node $v \in V_p$ is the maximal number of mutually non-equivalent nodes from V_p below v . Note that $c(v) = \text{ind}(v) \bmod 2$ for all nodes v if the pair (M, c) satisfies (1) in the definition of the language Col.

Lemma 3.8. *Let $(M, c) \in \text{Col}$. Then, for any $(v, v') \in \text{msg}$ with $\text{ind}(v) < \text{ind}(v')$, we have $c(v) \neq c(v')$.*

Proof. Suppose there is $(v, v') \in \text{msg}$ with $\text{ind}(v) < \text{ind}(v')$ but $c(v) = c(v')$. Since any element of M dominates a finite set, we can assume v' to be minimal with this problem. If $\text{ind}(v) + 1 = \text{ind}(v')$, we are done since $c(v) = \text{ind}(v) \bmod 2 \neq (\text{ind}(v) + 1) \bmod 2 = c(v')$. So let $\text{ind}(v) + 1 < \text{ind}(v')$. Since $(M, c) \in \text{Col}$ and $\text{ind}(v') - 1 > \text{ind}(v) \geq 1$, there exists $(u, u') \in \text{msg}$ with $\lambda(u') = \lambda(v')$, $c(u) = c(u')$, and $\text{ind}(u') = \text{ind}(v') - 1$. In particular, $u' < v'$ and therefore $u < v$. But then $\text{ind}(u) \leq \text{ind}(v)$. Now we have $\text{ind}(u) \leq \text{ind}(v) < \text{ind}(v') - 1 = \text{ind}(u')$, i.e., $u' < v'$ is another counterexample to the statement of the lemma. But this contradicts the choice of v' . \square

Corollary 3.9. *Let $(M, c) \in \text{Col}$ and let (v_1, v_2, \dots) be some infinite path in M . Then there exist infinitely many $i \in \mathbb{N}$ with $c(v_i) \neq c(v_{i+1})$.*

Proof. Since $\text{ind}^{-1}(n)$ is finite for any $n \in \mathbb{N}$, there are infinitely many $i \in \mathbb{N}$ with $\text{ind}(v_i) < \text{ind}(v_{i+1})$. If $(v_i, v_{i+1}) \in \text{proc}$, then $\text{ind}(v_{i+1}) = \text{ind}(v_i) + 1$ and therefore $c(v_i) \neq c(v_{i+1})$. If, in the other case, $(v_i, v_{i+1}) \in \text{msg}$, then by Lemma 3.8, we get $c(v_i) \neq c(v_{i+1})$. \square

4. TRANSLATION OF LOCAL FORMULAS

Let α be a local formula of PDL. We will construct a “small” CFM that accepts a pair (M, c) with M an MSC and $c : V \rightarrow \{0, 1\}$ iff c is the characteristic function of the set of positions satisfying α , i.e.,

$$c(v) = \begin{cases} 1 & \text{if } M, v \models \alpha \\ 0 & \text{otherwise.} \end{cases}$$

To obtain this CFM, we will first construct another CFM that accepts $(M, (c_\beta)_{\beta \in \text{sub}(\alpha)})$ iff, for all positions $v \in V$ and all subformulas β of α , we have $M, v \models \beta$ iff $c_\beta(v) = 1$. This CFM will consist of several CFMs running in conjunction, one for each subformula. For instance, if $\sigma \in \Sigma$ and $\delta = \beta \vee \gamma$ are subformulas of α , then we will have sub-CFMs that check whether, for any position v , we have $c_\sigma(v) = 1$ iff $\lambda(v) = \sigma$ and $c_\delta(v) = c_\beta(v) \vee c_\gamma(v)$, respectively. We first define these sub-CFMs for subformulas of the form σ , $\beta \vee \gamma$, and $\neg\beta$.

Example 4.1. For $\sigma \in \Sigma$, we define the CFM $\mathcal{A}_\sigma = (\{m\}, 1, (\mathcal{A}_p)_{p \in \mathcal{P}}, F)$ as follows: For $p \in \mathcal{P}$, let $S_p = \{\iota_p\}$ and $(\iota_p, \tau, b, m, \iota_p) \in \rightarrow_p$ iff

- $\tau = \sigma$ and $b = 1$ or
- $\tau \neq \sigma$ and $b = 0$.

Furthermore, $F = \{(\iota_p)_{p \in \mathcal{P}}\}$. Then it is easily checked that (M, c) is accepted by \mathcal{A}_σ iff

$$\forall v \in V : \lambda(v) = \sigma \iff c(v) = 1.$$

Example 4.2. Next we define a CFM $\mathcal{A}_\vee = (\{m\}, 3, (\mathcal{A}_p)_{p \in \mathcal{P}}, F)$: For $p \in \mathcal{P}$, let $S_p = \{\iota_p\}$ and $(\iota_p, \tau, (b_1, b_2, b_3), m, \iota_p) \in \rightarrow_p$ iff $b_3 = b_1 \vee b_2$. Furthermore, $F = \{(\iota_p)_{p \in \mathcal{P}}\}$. Then it is easily checked that (M, c) is accepted by \mathcal{A}_\vee iff

$$\forall v \in V : c_3(v) = c_1(v) \vee c_2(v).$$

The CFM \mathcal{A}_\neg is defined similarly.

Example 4.3. Next we define a CFM $\mathcal{A}_E = (\{m\}, 1, (\mathcal{A}_p)_{p \in \mathcal{P}}, F)$: For $p \in \mathcal{P}$, let $S_p = \{\iota_p, s_p\}$ and \rightarrow_p contain precisely $(\iota_p, \tau, 0, m, \iota_p)$, $(\iota_p, \tau, 1, m, s_p)$, and (s_p, τ, b, m, s_p) for all $\tau \in \Sigma_p$ and $b \in \{0, 1\}$. Furthermore, F is the set of tuples $(f_p)_{p \in \mathcal{P}}$ that contain at least one occurrence of s_p . Hence the index of this CFM is the number of processes $|\mathcal{P}|$.

Then it is easily checked that (M, c) is accepted by \mathcal{A}_E iff there exists a node v with $c(v) = 1$.

The CFM $\mathcal{A}_A = (\{m\}, 1, (\mathcal{A}_p)_{p \in \mathcal{P}}, F)$ has again just one local state per process (and is therefore of index 1): For $p \in \mathcal{P}$, let $S_p = \{\iota_p\}$, $\rightarrow_p = \{(\iota_p, \tau, 1, m, \iota_p) \mid \tau \in \Sigma_p\}$, and $F = \{(\iota_p)_{p \in \mathcal{P}}\}$.

Then it is easily checked that (M, c) admits a run and is therefore accepted by \mathcal{A}_A iff $c(v) = 1$ for all nodes v .

4.1. The backward-path automaton. Let π be a path expression, i.e., a regular expression over the alphabet $\{\text{proc}, \text{msg}, \{\alpha_1\}, \dots, \{\alpha_n\}\}$. Replacing $\{\alpha_i\}$ by i , we obtain a regular expression over the alphabet $\Gamma = \{\text{proc}, \text{msg}, 1, 2, \dots, n\}$. Let $L_\pi \subseteq \Gamma^*$ be the language of this regular expression.

A word over Γ together with a node from an MSC describes a path starting in that node that walks *backwards*. The letters *proc* and *msg* denote the direction of the path, the letters i denote requirements about the node currently visited (namely, that α_i shall hold). This idea motivates the following definition:

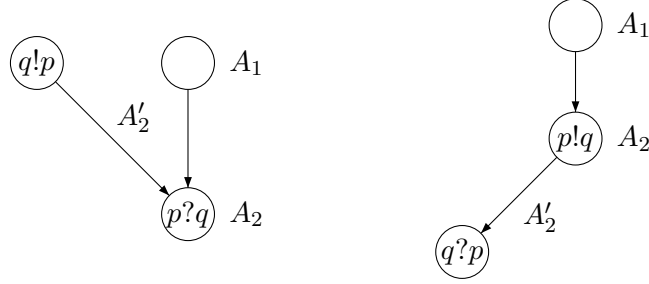


Figure 3: Transitions of the CFM.

Definition 4.4. For an MSC M , functions $c_1, \dots, c_n : V \rightarrow \{0, 1\}$, a node $v \in V$ and a word $W \in \Gamma^*$, we define inductively $(M, c_1, \dots, c_n), v \models^{-1} W$:

$$(M, c_1, \dots, c_n), v \models^{-1} \varepsilon$$

$$(M, c_1, \dots, c_n), v \models^{-1} \text{proc } W \iff \text{there is } v' = \text{proc}^{-1}(v) \text{ with } (M, c_1, \dots, c_n), v' \models^{-1} W$$

$$(M, c_1, \dots, c_n), v \models^{-1} \text{msg } W \iff \text{there is } v' = \text{msg}^{-1}(v) \text{ with } (M, c_1, \dots, c_n), v' \models^{-1} W$$

$$(M, c_1, \dots, c_n), v \models^{-1} iW \iff c_i(v) = 1 \text{ and } (M, c_1, \dots, c_n), v \models^{-1} W$$

We easily verify that $M, v \models \langle \pi \rangle^{-1} \#$ iff there exists $W \in L_\pi$ such that $M, v \models^{-1} W$.

Let $\mathcal{C} = (Q, \iota, T, G)$ be a finite automaton over Γ recognizing L_π . Note that we can assume $|Q| \in O(s(\pi))$. For $q \in Q$ and $W \in \Gamma^*$, we write $q.W \subseteq Q$ for the set of states that can be reached from q reading the word W , and we denote by $W.q \subseteq Q$ the set of states from which one can reach q when reading W . Furthermore, $P.L = \bigcup_{p \in P, W \in L} p.W$ and $L.P = \bigcup_{W \in L, p \in P} W.p$ for $P \subseteq Q$ and $L \subseteq \Gamma^*$ (if L (or P) is a singleton, then we may identify it with its unique element).

Lemma 4.5. *There exists a CFM \mathcal{A} with sets of local states 2^Q and set of messages 2^Q such that, for any run ρ of \mathcal{A} on (M, c_1, \dots, c_n) and any node v of M , we have $\rho(v) = \{q \in Q \mid \exists W \in \Gamma^* : q \in W.G \text{ and } M, v \models^{-1} W\}$.*

Proof. To define the set of transitions, let $A_1, A_2, A'_2 \subseteq 2^Q$, and let $a = (\sigma, b_1, \dots, b_n) \in \Sigma_p \times \{0, 1\}^n$ and $N = \{i \in [n] \mid b_i = 1\}$. Then we set

$$A_1 \xrightarrow{a, A'_2}_p A_2$$

iff the following conditions hold

- (1) if σ is a send action, then $A_2 = A'_2 = N^*.G \cup N^* \text{proc}.A_1$,
- (2) if σ is a receive action, then $A_2 = N^*.G \cup N^* \text{proc}.A_1 \cup N^* \text{msg}.A'_2$.

Here, A_1 is the local state assumed before the execution of the (labeled) action a , A_2 is the local state assumed afterwards, and A'_2 is the message involved in this transition. Depending on whether a is a receive or a send action, the message is consumed by a or emitted by a . These two situations are visualized in Figure 3.

The local initial state is \emptyset for any $p \in \mathcal{P}$ and any tuple of local states is accepting. Now let (ρ, μ) be a run of this CFM on (M, c_1, \dots, c_n) .

For $v \in V$ set $N_v = \{i \in [n] \mid c_i(v) = 1\}$ and $M(v) = \{W \in \Gamma^* \mid (M, c_1, \dots, c_n), v \models^{-1} W\}$. Then it is easily verified that

$$M(v) = N_v^* \cup \underbrace{N_v^* \text{proc } M(\text{proc}^{-1}(v))}_{\text{if } \text{proc}^{-1}(v) \text{ is defined}} \cup \underbrace{N_v^* \text{msg } M(\text{msg}^{-1}(v))}_{\text{if } \text{msg}^{-1}(v) \text{ is defined}} .$$

On the other hand,

$$\rho(v) = N_v^*.G \cup \underbrace{N_v^*.\text{proc}.\rho(\text{proc}^{-1}(v))}_{\text{if } \text{proc}^{-1}(v) \text{ is defined}} \cup \underbrace{N_v^*.\text{msg}.\rho(\text{msg}^{-1}(v))}_{\text{if } \text{msg}^{-1}(v) \text{ is defined}} .$$

Hence, by induction on the partial order (V, \leq) , we have $q \in \rho(v)$ iff $q \in M(v).G$. \square

Theorem 4.6. *Let $\langle \pi \rangle^{-1} \alpha$ be a local formula such that π is a regular expression over the alphabet $\{\text{proc}, \text{msg}, \{\alpha_1\}, \dots, \{\alpha_n\}\}$. Then there exists a CFM $\mathcal{A}_{\langle \pi \rangle^{-1} \alpha}$ of index 1 with the following property: Let M be an MSC and let $c_i : V \rightarrow \{0, 1\}$ be the characteristic function of the set of positions satisfying α_i (for all $i \in [n+1]$) where $\alpha_{n+1} = \alpha$. Then $(M, c_1, \dots, c_n, c_{n+1}, c)$ is accepted iff c is the characteristic function of the set of positions satisfying $\langle \pi \rangle^{-1} \alpha$.*

The CFM we construct has $2^{O(s(\pi))}$ local states per process, $2^{O(s(\pi))}$ many control messages, and any tuple of local states is accepting (in particular, the CFM has index 1).

Proof. Again, since $M, v \models \langle \pi \rangle^{-1} \alpha$ iff $M, v \models \langle \pi; \{\alpha\} \rangle^{-1} t$, we will assume $\alpha = t$. The CFM $\mathcal{A}_{\langle \pi \rangle^{-1} \alpha}$ simulates the run (ρ, μ) of the CFM \mathcal{A} from Lemma 4.5 and verifies that $c(v) = 1$ iff $\iota \in \rho(v)$ for all nodes $v \in V$. Then we have

$$\begin{aligned} c(v) = 1 &\iff \iota \in \rho(v) \\ &\iff \exists W \in \Gamma^* : \iota \in W.G \text{ and } M, v \models^{-1} W \\ &\iff \exists W \in L(\mathcal{C}) = L_\pi : M, v \models^{-1} W \\ &\iff \exists W \in L_\pi : M, v \models^{-1} W \\ &\iff M, v \models \langle \pi \rangle^{-1} t \end{aligned}$$

This concludes the proof of Theorem 4.6. \square

4.2. The forward-path automaton. We now turn to a similar CFM corresponding to subformulas of the form $\langle \pi \rangle t$. We will prove the following analog to Theorem 4.6. This proof will, however, be substantially more difficult.

Theorem 4.7. *Let $I \subseteq \text{Ch}$ and let $\langle \pi \rangle \alpha$ be a local formula such that π is a regular expression over the alphabet $\{\text{proc}, \text{msg}, \{\alpha_1\}, \dots, \{\alpha_n\}\}$. Then there exists a CFM $\mathcal{A}_{\langle \pi \rangle \alpha}$ of index 1 with the following property: Let M be an MSC with $\text{Inf}(M) = I$ and let $c_i : V \rightarrow \{0, 1\}$ be the characteristic function of the set of positions satisfying α_i (for all $i \in [n+1]$) where $\alpha_{n+1} = \alpha$. Then $(M, c_1, \dots, c_n, c_{n+1}, c)$ is accepted iff c is the characteristic function of the set of positions satisfying $\langle \pi \rangle \alpha$. The CFM we construct has $2^{O(s(\pi) + |\mathcal{P}|)}$ local states per process and $2^{O(s(\pi))}$ many control messages.*

The rest of this section is devoted to the proof of this theorem. Since $M, v \models \langle \pi \rangle \alpha$ iff $M, v \models \langle \pi; \{\alpha\} \rangle t$, we will assume $\alpha = t$, i.e., α holds true for any node of any MSC.

Let $\Gamma = \{\text{proc}, \text{msg}, 1, 2, \dots, n\}$. If, in the regular expression π , we replace any occurrence of $\{\alpha_i\}$ by i , we obtain a regular expression over the alphabet Γ . Let $L_\pi \subseteq \Gamma^*$ be the language denoted by this regular expression. Then there is a finite automaton $\mathcal{C} = (Q, \iota, T, G)$ over Γ with set of states Q , initial state ι , set of transitions T , and set of final states G recognizing L_π . Note that $|Q| \in O(s(\pi))$.

A word over Γ together with a node from an MSC describe a path starting in that node walking *forwards*. The following is therefore the forward-version of Def. 4.4.

Definition 4.8. For an MSC M , functions $c_1, \dots, c_n : V \rightarrow \{0, 1\}$, a node $v \in V$ and a word $W \in \Gamma^*$, we define inductively $(M, c_1, \dots, c_n), v \models W$:

$$\begin{aligned} (M, c_1, \dots, c_n), v &\models \varepsilon \\ (M, c_1, \dots, c_n), v &\models \text{proc } W \iff \text{there exists } v' = \text{proc}(v) \text{ with } (M, c_1, \dots, c_n), v' \models W \\ (M, c_1, \dots, c_n), v &\models \text{msg } W \iff \text{there exists } v' = \text{msg}(v) \text{ with } (M, c_1, \dots, c_n), v' \models W \\ (M, c_1, \dots, c_n), v &\models i W \iff c_i(v) = 1 \text{ and } (M, c_1, \dots, c_n), v \models W \end{aligned}$$

Now the following is immediate.

Lemma 4.9. *Let M be an MSC and, for $i \in [n]$, let $c_i : V \rightarrow \{0, 1\}$ be the characteristic function of the set of positions satisfying α_i . Then $M, v \models \langle \pi \rangle \#$ iff there exists $W \in L_\pi$ such that $M, v \models W$.*

Thus, in order to prove Theorem 4.7, it suffices to construct a CFM that accepts (M, c_1, \dots, c_n, c) iff

$$\begin{aligned} \forall v \in V : c(v) = 0 &\implies \forall W \in L_\pi : (M, c_1, \dots, c_n), v \not\models W \\ \forall v \in V : c(v) = 1 &\implies \exists W \in L_\pi : (M, c_1, \dots, c_n), v \models W. \end{aligned}$$

Since the class of languages accepted by CFMs is closed under intersection, we can handle the two implications separately in the following two subsections.

4.2.1. Any 0 is justified. We construct a CFM that accepts (M, c_1, \dots, c_n, c) iff, for any $v \in V$ with $c(v) = 0$, there does not exist $W \in L_\pi$ with $(M, c_1, \dots, c_n, c), v \models W$. The basic idea is rather simple: whenever the CFM encounters a node v with $c(v) = 0$, it will start the automaton \mathcal{C} (that accepts L_π) and check that it cannot reach an accepting state whatever path we choose starting in v . Since the CFM has to verify more than one 0, the set of local states S_p equals $2^{Q \setminus G}$ with initial state $\iota_p = \emptyset$ for any $p \in \mathcal{P}$. The set of control messages C equals $2^{Q \setminus G}$, too. Furthermore, any tuple of local states is accepting.

To define the set of transitions, let $A_1, A_2 \in S_p$ and $A'_2 \in C$. Moreover, let $a = (\sigma, b_1, \dots, b_n, b) \in \Sigma_p \times \{0, 1\}^{n+1}$ and $N = \{i \in [n] \mid b_i = 1\}$. Then we have a transition

$$A_1 \xrightarrow{a, A'_2}_p A_2$$

iff the following conditions hold:

- (1) if $b = 0$, then $\iota.N^* \subseteq A_2$,
- (2) $A_1.\text{proc}.N^* \subseteq A_2$,
- (3) if σ is a receive action, then $A'_2.\text{msg}.N^* \subseteq A_2$,
- (4) if σ is a send action, then $A'_2 = A_2$.

Lemma 4.10. *Let (ρ, μ) be a run of the above CFM on (M, c_1, \dots, c_n, c) and let $v_0 \in V$ with $c(v_0) = 0$. Then there does not exist $W \in L_\pi$ with $(M, c_1, \dots, c_n), v_0 \models W$.*

Proof. Suppose there is $W \in L_\pi$ with $(M, c_1, \dots, c_n), v_0 \models W$. Write $W = w_0 a_1 w_1 \dots a_n w_m$ with $a_k \in \{\text{proc}, \text{msg}\}$ and $w_k \in [n]^*$ for all appropriate k . Since $(M, c_1, \dots, c_n), v_0 \models W$, there exist nodes $v_k \in V$ with $v_{k+1} = a_{k+1}(v_k)$ and $w_k \subseteq N_{v_k}^*$ where $N_{v_k} = \{i \in [n] \mid c_i(v_k) = 1\}$. Since $W \in L_\pi$, there are states $q_i \in Q$ with $q_0 \in \iota.w_0$, $q_{i+1} \in q_i.a_{i+1}w_{i+1}$, and $q_m \in G$.

Since $c(v_0) = 0$, we have $\iota.N_{v_0}^* \subseteq \rho(v_0)$ by (1) and therefore $q_0 \in \iota.w_0 \subseteq \rho(v_0)$ by $w_0 \in N_{v_0}^*$. By induction, assume $k < m$ and $q_k \in \rho(v_k)$. If $a_{k+1} = \text{proc}$, then by (2) $q_{k+1} \in q_k.\text{proc}.N_{v_{k+1}}^* \subseteq \rho(v_{k+1})$. If $a_{k+1} = \text{msg}$, then v_k is a send event. Hence, by (4), $\mu(v_k) = \rho(v_k)$. Since $(v_k, v_{k+1}) \in \text{msg}$, this implies $\mu(v_{k+1}) = \rho(v_k)$. Hence, by (3), $q_{k+1} \in \rho(v_k).\text{msg}.N_{v_{k+1}}^* \subseteq \rho(v_{k+1})$. This finishes the inductive argument. Hence $q_m \in \rho(v_n) \cap G$, contradicting our definition $S_p = 2^{Q \setminus G}$. \square

Lemma 4.11. *Suppose (M, c_1, \dots, c_n, c) satisfies*

$$\forall v \in V : c(v) = 0 \implies \forall W \in L_\pi : (M, c_1, \dots, c_n), v \not\models W.$$

Then (M, c_1, \dots, c_n, c) admits a run of the above CFM.

Proof. For $v \in V$, let $N_v = \{i \in [n] \mid c_i(v) = 1\}$. Then define $\rho(v)$ to be the union of the following sets

- (a) $\iota.N_v^*$ if $c(v) = 0$,
- (b) $\rho(\text{proc}^{-1}(v)).\text{proc}.N_v^*$ if $\text{proc}^{-1}(v)$ is defined (i.e., if v is not minimal on its process),
- (c) $\rho(\text{msg}^{-1}(v)).\text{msg}.N_v^*$ if $\text{msg}^{-1}(v)$ is defined (i.e., if $\lambda(v)$ is a receive action).

Furthermore, let

$$\mu(v) = \begin{cases} \rho(v) & \text{if } \lambda(v) \text{ is a send action} \\ \rho(\text{msg}^{-1}(v)) & \text{otherwise.} \end{cases}$$

Then, for any $v \in V$, the transition conditions (1-4) are satisfied by the mappings ρ and μ (recall that the local initial states are \emptyset).

Now, by contradiction, assume (ρ, μ) is no run, i.e., there is some $v_0 \in V$ with $\rho(v_0) \not\subseteq 2^{Q \setminus G}$. Hence there exists $q_0 \in \rho(v_0) \cap G$. Setting $W_0 = \varepsilon$, we therefore have

$$(M, c_1, \dots, c_n, c), v_k \models W_k, q_k \in \rho(v_k), \text{ and } q_k.W_k \cap G \neq \emptyset \quad (*)$$

for $k = 0$. Now assume that $(*)$ holds for some $k \geq 0$.

First, assume $c(v_k) = 0$ and $q_k \in \iota.N_{v_k}^* \subseteq \rho(v_k)$ because of (a). Hence, there exists $w_k \in N_{v_k}^*$ with $q_k \in \iota.w_k$. But then $(M, c_1, \dots, c_n, c), v_k \models w_k.W_k$ and $w_k.W_k \in L_\pi$, a contradiction. Hence we have $q_k \in \rho(v_k)$ because of (b) or (c). If $q_k \in \rho(\text{proc}^{-1}(v_k)).\text{proc}.N_{v_k}^*$, then set $v_{k+1} = \text{proc}^{-1}(v_k)$ and choose $q_{k+1} \in \rho(v_{k+1})$ and $w_k \in N_{v_k}^*$ with $q_k \in q_{k+1}.\text{proc}.w_k$. Setting $W_{k+1} = \text{proc}.w_k.W_k$ yields $(*)$ for $k + 1$. If $q_k \in \rho(\text{msg}^{-1}(v_k)).\text{msg}.N_{v_k}^*$, we can argue similarly.

Hence we find an infinite sequence of nodes $v_0 > v_1 > v_2 \dots$ which is impossible since v_0 dominates only a finite set. Thus, (ρ, μ) is a run. \square

Proposition 4.12. *There exists a CFM \mathcal{A}_0 of index 1 that accepts (M, c_1, \dots, c_n, c) iff*

$$\forall v \in V : c(v) = 0 \implies \forall W \in L_\pi : (M, c_1, \dots, c_n), v \not\models W.$$

The number of local states per process as well as the number of messages are in $2^{O(s(\pi))}$. Furthermore, any run of the CFM is accepting.

Proof. The proof is immediate by the above two lemmas. \square

4.2.2. *Any 1 is justified.* We next construct a CFM that accepts (M, c_1, \dots, c_n, c) iff, for any $v \in V$ with $c(v) = 1$, there exists $W \in L_\pi$ with $(M, c_1, \dots, c_n, c), v \models W$. Again, the basic idea is simple: whenever the CFM encounters a node v with $c(v) = 1$, it will start the automaton \mathcal{C} (that accepts L_π) and check that it can reach an accepting state along one of the possible paths. Thus, before, we had to prevent \mathcal{C} from reaching an accepting state. This time, we have to ensure that any verification of a $c(v) = 1$ will eventually result in an accepting state being reached. For sequential Büchi-automata, solutions to this problem are known: collect some claims to be verified in one set and, only when all of them are verified, start verifying those claims that have been encountered during the previous verification phase. The resulting Büchi-automaton accepts iff the verification phase is changed infinitely often. We will adapt precisely this idea here. But then, the CFM would have to accept if, along *each and every* path, the verification phase changes infinitely often. This is the point where the CFM \mathcal{A}_{Col} comes into play since, by Corollary 3.9, it verifies that any path runs through infinitely many color changes. Thus, we will first construct a CFM that runs on tuples $(M, c_0, c_1, \dots, c_n, c)$ where we assume that $(M, c_0) \in \text{Col}$. The actual CFM that verifies all claims $c(v) = 1$ will run this newly constructed CFM in conjunction with \mathcal{A}_{Col} (that verifies $(M, c_0) \in \text{Col}$) and project away the labeling c_0 .

For any $p \in \mathcal{P}$, the set of local states S_p equals $2^Q \times 2^Q \times \{0, 1\}$ with initial state $\iota_p = (\emptyset, \emptyset, 1)$, the set of control messages C equals $2^Q \times 2^Q \times \{0, 1\}$.

To define the set of transitions, let $(A_1, B_1, d_1), (A_2, B_2, d_2) \in S_p$ and $(A'_2, B'_2, d'_2) \in C$. Furthermore, let $a = (\sigma, b_0, b_1, \dots, b_n, b) \in \Sigma_p \times \{0, 1\}^{n+2}$. Now we would like to define the conditions for the existence of a transition

$$(A_1, B_1, d_1) \xrightarrow{a, (A'_2, B'_2, d'_2)}_p (A_2, B_2, d_2).$$

We have to distinguish between σ being a send or a receive event, cf. Figure 4. For $\sigma = p!q$ the pair (A_1, B_1) contains the in-going information whereas (A_2, B_2) and (A'_2, B'_2) carry the out-going information propagated along the process and the channel, respectively. On the other hand, for $\sigma = p?q$ now both (A_1, B_1) and (A'_2, B'_2) contain the in-going information whereas the out-going information can be propagated along the process line only, hence, it is enclosed in (A_2, B_2) only. Therefore, we put

$$A_{\text{in}} = A_1, A_{\text{out}} = A_2 \cup A'_2, B_{\text{in}} = B_1, B_{\text{out}} = B_2 \cup B'_2$$

whenever σ is a send event and

$$A_{\text{in}} = A_1 \cup A'_2, A_{\text{out}} = A_2, B_{\text{in}} = B_1 \cup B'_2, B_{\text{out}} = B_2$$

whenever σ is a receive event.

Now the idea is the following: The CFM saves the actual color within its state and propagates it via the channel whenever σ is a send. Whenever we stay within the same color ($d_2 = d_1$ and, for σ a receive, also $d_2 = d'_2$) we propagate the states (from the finite automaton \mathcal{C}) contained in A_{in} to A_{out} and likewise from B_{in} to B_{out} . But whenever the color changes ($d_2 \neq d_1$ or, for σ a receive, $d_2 \neq d'_2$), we require the respective part of A_{in} to be empty and all the information from the respective B_{in} is swept to A_{out} . Moreover, whenever a new 1 has to be verified we start \mathcal{C} and collect the states obtained this way within B_{out} . Now we formalize these ideas.

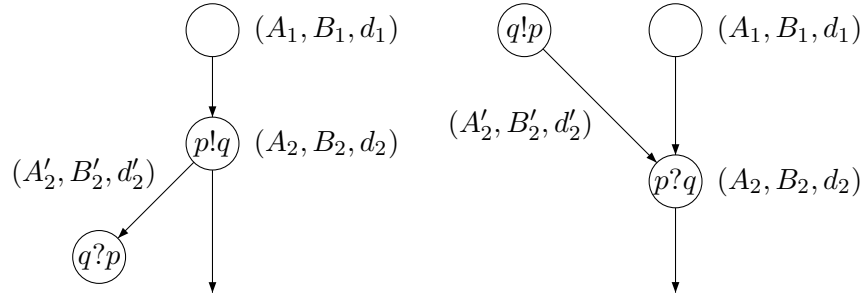


Figure 4: Transitions of the CFM.

Let $N = \{i \in [n] \mid b_i = 1\}$. Recall that $\mathcal{C} = (Q, \iota, T, G)$ is the finite automaton recognizing L_π . Then the transition above is defined iff the following conditions hold:

- (1) $d_2 = b_0$ and, if σ is a send, then also $d'_2 = b_0$,
- (2) if $b = 1$, then $\iota.N^* \cap (G \cup B_{\text{out}}) \neq \emptyset$,
- (3) $\forall q \in A_1 : q.\text{proc } N^* \cap (G \cup A_{\text{out}}) \neq \emptyset$, and,
if σ is a receive, then also $\forall q \in A'_2 : q.\text{msg } N^* \cap (G \cup A_{\text{out}}) \neq \emptyset$,
- (4) if $d_2 = d_1$, then $\forall q \in B_1 : q.\text{proc } N^* \cap (G \cup B_{\text{out}}) \neq \emptyset$,
- (5) if $d_2 \neq d_1$, then $A_1 = \emptyset$ and $\forall q \in B_1 : q.\text{proc } N^* \cap (G \cup A_{\text{out}}) \neq \emptyset$,
- (6) if $d_2 = d'_2$ and σ is a receive, then $\forall q \in B'_2 : q.\text{msg } N^* \cap (G \cup B_{\text{out}}) \neq \emptyset$,
- (7) if $d_2 \neq d'_2$ and σ is a receive, then $A'_2 = \emptyset$ and $\forall q \in B'_2 : q.\text{msg } N^* \cap (G \cup A_{\text{out}}) \neq \emptyset$.

Note that for a color change (cases (5) and (7)) the respective conditions in (3) become obsolete since $A_1 = \emptyset$ and/or $A'_2 = \emptyset$.

Recall that I is a set of channels and that we are only interested in MSCs that use precisely these channels infinitely often. Let $(f_p)_{p \in \mathcal{P}} \in \prod_{p \in \mathcal{P}} S_p$ be accepting in \mathcal{A} iff $f_p \in \{(\emptyset, \emptyset, 0), (\emptyset, \emptyset, 1)\}$ for all $p \in \mathcal{P}$ that are not involved in any of the channels from I , i.e., that satisfy $I \cap (\{p\} \times \mathcal{P} \cup \mathcal{P} \times \{p\}) = \emptyset$ (note that a process p is not involved in any of the channels from I iff it is not involved in any of the channels used infinitely often iff p executes only finitely many events). This finishes the construction of the CFM \mathcal{A} of index 1.

Lemma 4.13. *Let (ρ, μ) be an accepting run of the above CFM \mathcal{A} on $(M, c_0, c_1, \dots, c_n, c)$ and suppose $(M, c_0) \in \text{Col}$ and $I \subseteq \text{Inf}(M)$. Then, for any $v_0 \in V$ with $c(v_0) = 1$, there exists $W \in L_\pi$ with $(M, c_1, \dots, c_n), v_0 \models W$.*

Proof. For $v \in V$, let $\rho(v) = (A_v, B_v, d_v)$, $\mu(v) = (A'_v, B'_v, d'_v)$, $N_v = \{i \in [n] \mid c_i(v) = 1\}$. Similarly as above, whenever $\lambda(v)$ is a send event, we put

$$A_{\text{out}}(v) = A_v \cup A'_v, \quad B_{\text{out}}(v) = B_v \cup B'_v,$$

and whenever $\lambda(v)$ is a receive event, we put

$$A_{\text{out}}(v) = A_v, \quad B_{\text{out}}(v) = B_v.$$

Since $c(v_0) = 1$, (2) implies the existence of $w_0 \in N_{v_0}^*$ and $q_0 \in \iota.w_0 \cap (G \cup B_{\text{out}})$. Now we define a finite or infinite sequence $(v_i, w_i, q_i)_{0 \leq i < N}$ with $N \in \mathbb{N} \cup \{\omega\}$, $v_i \in V$, $w_i \in \{\text{proc}, \text{msg}\}N_{v_i}^*$ for $i \geq 1$, and $q_i \in Q$ such that the following hold for all $0 \leq i < N$:

- (a) $(v_i, v_{i+1}) \in \text{proc} \cup \text{msg}$,
- (b) $q_i \in G \cup A_{\text{out}}(v_i) \cup B_{\text{out}}(v_i)$,

- (c) $w_{i+1} \in a N_{v_{i+1}}^*$ and $q_{i+1} \in q_i \cdot w_{i+1}$ with $a = \begin{cases} \text{proc} & \text{if } (v_i, v_{i+1}) \in \text{proc}, \\ \text{msg} & \text{otherwise,} \end{cases}$
- (d) if $q_i \in A_{\text{out}}(v_i)$, then $q_{i+1} \in G \cup A_{\text{out}}(v_{i+1})$,
- (e) $q_i \in G \iff N = i + 1$.

Let us assume that the sequence has already been constructed up to index i . Then we proceed as follows:

- (i) If $q_i \in G$, then set $N = i + 1$ which finishes the construction of the sequence and implies (e) *a posteriori* for all $0 \leq i < N$.
- (ii) Suppose $q_i \in A_{v_i} \setminus G$. Since $A_{v_i} \neq \emptyset$ and since the run is accepting, the node v_i is not maximal on its process, so we can set $v_{i+1} = \text{proc}(v_i)$. Then, by (3), we can choose $w_{i+1} \in \text{proc} N_{v_{i+1}}^*$ and $q_{i+1} \in q_i \cdot w_{i+1} \cap (G \cup A_{\text{out}}(v_{i+1}))$.
- (iii) Suppose $q_i \in A_{\text{out}}(v_i) \setminus (G \cup A_{v_i})$. Then v_i is a send event. Hence, $v_{i+1} = \text{msg}(v_i)$ is a well-defined receive event. Hence, by (3), there exist $w_{i+1} \in \text{msg} N_{v_{i+1}}^*$ and $q_{i+1} \in q_i \cdot w_{i+1} \cap (G \cup A_{\text{out}}(v_{i+1}))$ such that (a) – (e) hold.
- (iv) Suppose $q_i \in B_{v_i} \setminus (G \cup A_{\text{out}}(v_i))$. Since $B_{v_i} \neq \emptyset$ and since the run is accepting, the node v_i cannot be maximal on its process, i.e., $v_{i+1} = \text{proc}(v_i)$ is well-defined. Then
- (a) if $d_{v_{i+1}} = d_{v_i}$, by (4), $q_i \cdot \text{proc} N_{v_{i+1}}^* \cap (G \cup B_{\text{out}}(v_{i+1})) \neq \emptyset$, and
- (b) if $d_{v_{i+1}} \neq d_{v_i}$, by (5), $q_i \cdot \text{proc} N_{v_{i+1}}^* \cap (G \cup A_{\text{out}}(v_{i+1})) \neq \emptyset$.
- Hence we can choose $w_{i+1} \in \text{proc} N_{v_{i+1}}^*$ and q_{i+1} such that (a) – (e) hold.
- (v) Finally, suppose $q_i \in B_{\text{out}}(v_i) \setminus (G \cup A_{\text{out}}(v_i) \cup B_{v_i})$ such that v_i is a send event, i.e., $v_{i+1} = \text{msg}(v_i)$ is a well-defined receive event. Hence
- (a) if $d_{v_{i+1}} = d'_{v_{i+1}}$, by (6), $q_i \cdot \text{msg} N_{v_{i+1}}^* \cap (G \cup B_{\text{out}}(v_{i+1})) \neq \emptyset$, and
- (b) if $d_{v_{i+1}} \neq d'_{v_{i+1}}$, by (7), $q_i \cdot \text{msg} N_{v_{i+1}}^* \cap (G \cup A_{\text{out}}(v_{i+1})) \neq \emptyset$.
- Again, we can choose w_{i+1} and q_{i+1} such that (a) – (e) hold.

If the construction can be carried out *ad infinitum*, then set $N = \omega$ which, again, ensures (e) *a posteriori* for all $i < N$.

Now suppose $N = \omega$. Then, by Cor. 3.9, there exist $0 \leq i < k$ with $c_0(v_i) \neq c_0(v_{i+1})$ and $c_0(v_k) \neq c_0(v_{k+1})$. By (1), this implies $d_{v_i} \neq d_{v_{i+1}}$ whenever $(v_i, v_{i+1}) \in \text{proc}$ or $d'_{v_{i+1}} \neq d_{v_{i+1}}$ whenever $(v_i, v_{i+1}) \in \text{msg}$. Similarly, $d_{v_k} \neq d_{v_{k+1}}$ for $(v_k, v_{k+1}) \in \text{proc}$ and $d'_{v_{k+1}} \neq d_{v_{k+1}}$ for $(v_k, v_{k+1}) \in \text{msg}$. Let us assume $(v_i, v_{i+1}) \in \text{proc}$ and $(v_k, v_{k+1}) \in \text{proc}$, i.e., $d_{v_i} \neq d_{v_{i+1}}$ and $d_{v_k} \neq d_{v_{k+1}}$. Hence, by (5), we get $A_{v_i} = A_{v_k} = \emptyset$. Since, by (e), $q_i \notin G$, $A_{v_i} = \emptyset$, and $(v_i, v_{i+1}) \in \text{proc}$, we get $q_i \in B_{v_i}$ and, therefore, by case (iv)(b) of the construction above $q_{i+1} \in A_{\text{out}}(v_{i+1})$. Applying (e) and (d) inductively, this results in $q_k \in A_{\text{out}}(v_k)$. Since $(v_k, v_{k+1}) \in \text{proc}$, we conclude by the construction (cases (ii) and (iii)) $q_k \in A_{v_k} \setminus G$. But this contradicts $A_{v_k} = \emptyset$. For the other cases a contradiction is obtained similarly. Hence, N is finite.

Certainly, $(M, c_1, \dots, c_n), v_{N-1} \models \varepsilon$. From (c), we obtain $(M, c_1, \dots, c_n), v_{N-2} \models w_{N-1}$ and, by induction, $(M, c_1, \dots, c_n), v_0 \models W$ with $W = w_0 w_1 \dots w_{N-1}$. Since $q_0 \in \iota.w_0$, (c) implies $q_{N-1} \in \iota.W$ and therefore $W \in L_\pi$ follows from (e). \square

Lemma 4.14. *Suppose (M, c_1, \dots, c_n, c) satisfies*

$$\forall v \in V : c(v) = 1 \implies \exists W \in L_\pi : (M, c_1, \dots, c_n), v \models W$$

and $\text{Inf}(M) \subseteq I$. Then there exists a mapping $c_0 : V \rightarrow \{0, 1\}$ such that $(M, c_0) \in \text{Col}$ and the above CFM \mathcal{A} accepts $(M, c_0, c_1, \dots, c_n, c)$.

Proof. For any $v \in V$ with $c(v) = 1$, there exist $0 \leq k^v \in \mathbb{N}$, $w_0^v \in [n]^*$, $w_i^v \in \{\text{proc}, \text{msg}\}[n]^*$ for $1 \leq i \leq k^v$, and $q_i^v \in Q$ for $0 \leq i \leq k^v$ such that

- (a) $q_0^v \in \iota.w_0^v$, $q_{i+1}^v \in q_i^v.w_{i+1}^v$ for $1 \leq i < k^v$, and $q_{k^v}^v \in G$
- (b) $v_0^v = v$ and, for $0 \leq i < k^v$, $v_{i+1}^v = \begin{cases} \text{proc}(v_i^v) & \text{if } w_{i+1}^v \in \text{proc}[n]^* \\ \text{msg}(v_i^v) & \text{if } w_{i+1}^v \in \text{msg}[n]^* \end{cases}$

We define inductively a sequence of subsets of V : Let $H_0 = \emptyset$. Inductively, let $H_{n+1} \subseteq V \setminus \bigcup_{0 \leq i \leq n} H_i$ be nonempty and finite such that

- (A) $\bigcup_{0 \leq i \leq n+1} H_i$ is downwards closed in M ,
- (B) for any $v \in V \setminus \bigcup_{0 \leq i \leq n+1} H_i$ with $\lambda(v) = p?q$,
 - (B1) there exist infinitely many $v' \in V \setminus \bigcup_{0 \leq i \leq n+1} H_i$ with $\lambda(v) = \lambda(v')$,
 - (B2) there exist $u, u' \in H_{n+1}$ with $(u, u') \in \text{msg}$ and $\lambda(u') = \lambda(v)$,
- (C) for any $v \in H_n$ with $c(v) = 1$, we have $v_{k^v}^v \in H_n \cup H_{n+1}$.

Then $V = \bigcup_{n \geq 0} H_n$.

Now set, for $v \in H_n$,

- $c_0(v) = n \bmod 2$ and $d_v = c_0(v)$
- if v is a send event, then

$$\begin{aligned} A_v &= \{q_i^{\bar{v}} \mid \bar{v} \in H_{n-1}, c(\bar{v}) = 1, 0 \leq i < k^{\bar{v}} \text{ such that } v = v_i^{\bar{v}} \text{ and } w_{i+1}^{\bar{v}} \in \text{proc}[n]^*\} \\ B_v &= \{q_i^{\bar{v}} \mid \bar{v} \in H_n, c(\bar{v}) = 1, 0 \leq i < k^{\bar{v}} \text{ such that } v = v_i^{\bar{v}} \text{ and } w_{i+1}^{\bar{v}} \in \text{proc}[n]^*\} \\ A'_v &= \{q_i^{\bar{v}} \mid \bar{v} \in H_{n-1}, c(\bar{v}) = 1, 0 \leq i < k^{\bar{v}} \text{ such that } v = v_i^{\bar{v}} \text{ and } w_{i+1}^{\bar{v}} \in \text{msg}[n]^*\} \\ B'_v &= \{q_i^{\bar{v}} \mid \bar{v} \in H_n, c(\bar{v}) = 1, 0 \leq i < k^{\bar{v}} \text{ such that } v = v_i^{\bar{v}} \text{ and } w_{i+1}^{\bar{v}} \in \text{msg}[n]^*\} \\ d'_v &= c_0(v) \end{aligned}$$

- if v is a receive event, then

$$\begin{aligned} A_v &= \{q_i^{\bar{v}} \mid \bar{v} \in H_{n-1}, c(\bar{v}) = 1, 0 \leq i < k^{\bar{v}} \text{ such that } v = v_i^{\bar{v}}\} \\ B_v &= \{q_i^{\bar{v}} \mid \bar{v} \in H_n, c(\bar{v}) = 1, 0 \leq i < k^{\bar{v}} \text{ such that } v = v_i^{\bar{v}}\} \\ A'_v &= A'_{\text{msg}^{-1}(v)} \\ B'_v &= B'_{\text{msg}^{-1}(v)} \\ d'_v &= d'_{\text{msg}^{-1}(v)} \end{aligned}$$

Then the pair of mappings (ρ, μ) with $\rho(v) = (A_v, B_v, d_v)$ and $\mu(v) = (A'_v, B'_v, d'_v)$ is an accepting run of the CFM on $(M, c_0, c_1, \dots, c_n, c)$ and $(M, c_0) \in \text{Col}$. \square

Proposition 4.15. *Let $I \subseteq \text{Ch}$. There is a CFM \mathcal{A}_1 that accepts (M, c_1, \dots, c_n, c) with $\text{Inf}(M) = I$ iff*

$$\forall v \in V : c(v) = 1 \implies \exists W \in L_\pi : (M, c_1, \dots, c_n), v \models W.$$

The number of local states per process is in $2^{O(|\mathcal{P}|+s(\pi))}$ and the number of messages is in $2^{O(s(\pi))}$.

Proof. By Lemma 3.2, there exists a CFM \mathcal{B} with the given number of states and messages that accepts $(M, c_0, c_1, \dots, c_n, c)$ iff it is accepted by \mathcal{A}_{Col} from Prop. 3.7 and by the above CFM \mathcal{A} , i.e., iff $(M, c_0) \in \text{Col}$, and $(M, c_0, c_1, \dots, c_n, c)$ is accepted by \mathcal{A} . Projecting away the function c_0 gives the CFM \mathcal{A}_1 by the above two lemmas. \square

Proof (of Theorem 4.7). The result follows immediately from Propositions 4.12, 4.15, and Lemma 3.2. \square

4.3. The overall construction.

Theorem 4.16. *Let $I \subseteq \text{Ch}$ and let α be a local formula of PDL. Then one can construct a CFM \mathcal{B} of index 1 such that $(M, (c_\beta)_{\beta \in \text{sub}(\alpha)})$ with $\text{Inf}(M) = I$ is accepted by \mathcal{B} iff $c_\beta : V \rightarrow \{0, 1\}$ is the characteristic function of the set of positions that satisfy β for all $\beta \in \text{sub}(\alpha)$.*

With m the number of subformulas of the form $\langle \pi \rangle \gamma$ and $\langle \pi \rangle^{-1} \gamma$ and $n \in \mathbb{N}$ such that $s(\pi; \gamma) \leq n$ for all such subformulas, the number of local states per process is in $2^{O(m(n+|\mathcal{P}|))}$ and the number of control messages is in $2^{O(mn)}$.

Proof. The CFM \mathcal{B} has to accept $(M, (c_\beta)_{\beta \in \text{sub}(\alpha)})$ iff

- (1) \mathcal{A}_σ accepts (M, c_σ) for all $\sigma \in \text{sub}(\alpha) \cap \Sigma$ (cf. Example 4.1),
- (2) \mathcal{A}_\vee accepts $(M, c_\gamma, c_\delta, c_{\gamma \vee \delta})$ for all $\gamma \vee \delta \in \text{sub}(\alpha)$ (cf. Example 4.2),
- (3) \mathcal{A}_\neg accepts $(M, c_\gamma, c_{\neg \gamma})$ for all $\neg \gamma \in \text{sub}(\alpha)$ (cf. Example 4.2),
- (4) $\mathcal{A}_{\langle \pi \rangle \gamma}$ accepts $(M, c_{\alpha_1}, \dots, c_{\alpha_n}, c_\gamma, c_{\langle \pi \rangle \gamma})$ for all $\langle \pi \rangle \gamma \in \text{sub}(\alpha)$ where $\alpha_1, \dots, \alpha_n$ are those local formulas for which $\{\alpha_i\}$ appears in the path expression π (cf. Theorem 4.7), and
- (5) $\mathcal{A}_{\langle \pi \rangle^{-1} \gamma}$ accepts $(M, c_{\alpha_1}, \dots, c_{\alpha_n}, c_\gamma, c_{\langle \pi \rangle^{-1} \gamma})$ for all $\langle \pi \rangle^{-1} \gamma \in \text{sub}(\alpha)$ where $\alpha_1, \dots, \alpha_n$ are those local formulas for which $\{\alpha_i\}$ appears in the path expression π (cf. Theorem 4.6).

Recall that the CFMs from (4) all have index 1, their number of local states per process is bounded by $2^{O(n+|\mathcal{P}|)}$, and their number of messages is bounded by $2^{O(n)}$. Hence, by Lemma 3.2, there exists a CFM of index 1 that checks all the requirements in (4). Its number of states is in

$$\begin{aligned} (m+1) \cdot \prod_{\langle \pi \rangle \gamma \in \text{sub}(\alpha)} 2^{O(n+|\mathcal{P}|)} &\subseteq (m+1) \cdot 2^{O(m(n+|\mathcal{P}|))} \\ &\subseteq 2^{O(m(n+|\mathcal{P}|))} \end{aligned}$$

and the number of control messages belongs to

$$\prod_{\langle \pi \rangle \gamma \in \text{sub}(\alpha)} 2^{O(s(\pi))} \subseteq 2^{O(mn)}.$$

Any tuple of local states in any of the CFMs from (5) is accepting. Furthermore, any of them has $2^{O(n)}$ local states per process and equally many messages. Hence there is a CFM with $2^{O(mn)}$ local states per process and equally many messages that checks all the requirements in (5). Furthermore, all tuples of states of this machine are accepting.

Recall that the CFMs \mathcal{A}_σ , \mathcal{A}_\vee , and \mathcal{A}_\neg have just one local state per process, i.e., they only restrict the labels $(\sigma, (b_\beta)_{\beta \in \text{sub}(\alpha)})$ allowed in M . Hence, without additional states or messages, one can change the above two CFMs into a CFM \mathcal{B} of index 1 that checks (1)–(5). Its number of local states per process is in $2^{O(m(n+|\mathcal{P}|))}$ and its number of messages is in $2^{O(mn)}$. \square

5. TRANSLATION OF GLOBAL FORMULAS

A *basic global formula* is a formula of the form $A\alpha$ or $E\alpha$ for α a local formula. Then global formulas are positive Boolean combinations of basic global formulas.

Proposition 5.1. *Let φ be a global formula and $I \subseteq \text{Ch}$. Then one can construct a CFM \mathcal{A} that accepts M with $\text{Inf}(M) = I$ iff $M \models \varphi$.*

With ℓ the number of basic global subformulas of φ , m the number of subformulas of the form $\langle \pi \rangle \beta$ and $\langle \pi \rangle^{-1} \beta$, and $n \in \mathbb{N}$ such that $s(\pi; \beta) \leq n$ for all such subformulas, the number of local states per process is in $2^{O(m(n+|\mathcal{P}|))+|\mathcal{P}|^\ell}$, the number of control messages is in $2^{O(\ell+mn)}$, and the index is at most $|\mathcal{P}^\ell|$.

Proof. Let H be the set of basic global subformulas of φ . Let $\beta = \bigwedge \{ \alpha \mid E\alpha \in H \text{ or } A\alpha \in H \}$. Using Proposition 3.3, one can construct a CFM that accepts $(M, (c_\gamma)_{\gamma \in \text{sub}(\beta)})$ with $\text{Inf}(M) = I$ iff

- c_γ is the characteristic function of the set of positions satisfying γ for all $\gamma \in \text{sub}(\beta)$ (Thm. 4.16)
- $M \models A\alpha$ for all $A\alpha \in H$ (Example 4.3)
- $M \models E\alpha$ for all $E\alpha \in H$ (Example 4.3).

Recall that the CFM checking c_γ as well as those checking $A\alpha$ all have index 1 while the CFM for $E\alpha$ have index $|\mathcal{P}|$. Hence the number of local states per process of the resulting CFM belongs to $1 + (|H| + 1) \cdot 2^{O(m(n+|\mathcal{P}|))} \cdot 2^{|H|} \cdot |\mathcal{P}|^{|H|} \subseteq 2^{O(m(n+|\mathcal{P}|))+|\mathcal{P}|^\ell}$, its number of messages is in $2^{O(mn)}$, and its index is at most $|\mathcal{P}^H|$. Let \mathcal{A}_H denote the projection of this CFM to the set of MSCs (i.e., we project away the labelings c_γ). Then \mathcal{A}_H accepts an MSC M with $\text{Inf}(M) = I$ iff $M \models \psi$ for all $\psi \in H$.

Now the CFM \mathcal{A} is the disjoint union of at most 2^ℓ many CFMs of the form \mathcal{A}_H . \square

Theorem 5.2. *Let φ be a global formula of PDL. Then one can construct a CFM \mathcal{A} that accepts M iff $M \models \varphi$.*

With ℓ the number of basic global subformulas of φ , m the number of subformulas of the form $\langle \pi \rangle \beta$, and $n \in \mathbb{N}$ such that $s(\pi; \beta) \leq n$ for all such subformulas, the number of local states per process is in $2^{O(m(n+|\mathcal{P}|)+|\mathcal{P}|^\ell+|\mathcal{P}|^2)}$ and the number of control messages is in $2^{O(\ell+mn+|\mathcal{P}|^2)}$.

Proof. Let, for $I \subseteq \text{Ch}$, \mathcal{A}_I denote the CFM from Prop. 5.1 and \mathcal{B}_I that from Prop. 3.6. Using Prop. 3.3, one can construct a CFM \mathcal{C}_I accepting $L(\mathcal{A}_I) \cap L(\mathcal{B}_I)$. The number of local states per process of this CFM is $3 \cdot 2^{O(|\mathcal{P}|)} \cdot 2^{O(m(n+|\mathcal{P}|)+|\mathcal{P}|^\ell)} \cdot |\mathcal{P}^H|$.

Then the disjoint union \mathcal{A} of all these CFMs \mathcal{C}_I for $I \subseteq \text{Ch}$ has all the desired properties. \square

6. MODEL CHECKING

6.1. CFMs vs. PDL specifications. We aim at an algorithm that decides whether, given a global formula φ and a CFM \mathcal{A} , every MSC $M \in L(\mathcal{A})$ satisfies φ . The undecidability of this problem can be shown following, e.g., the proof in [MR04] (that paper deals with Lamport diagrams and a fragment LD_0 of PDL, but the proof ideas can be easily transferred to our setting). To gain decidability, we follow the successful approach of, e.g., [MM01, GMSZ02, GKM06], and restrict attention to existentially B -bounded MSCs from $L(\mathcal{A})$.

For a finite or infinite word $w \in \Sigma^\infty$ and $a \in \Sigma$, let $|w|_a$ denote the number of occurrences of a in w . For $0 \leq i \leq j < |w|$, the infix $w[i, j]$ is the factor of w starting in position i and ending in position j , i.e., $w = u w[i, j] v$ with $|u| = i$ and $|w[i, j]| = j - i + 1$. If $|w| > i$, then we write $w(i)$ for $w[i, i]$, the letter no. $i + 1$ in w (note that $w(0)$ is the first letter of w).

Let $M = (V, \leq, \lambda)$ be an MSC. A *linearization* of M is a linear order $\preceq \supseteq \leq$ on V of order type at most ω (i.e., also with respect to \preceq , any node $v \in V$ dominates a finite set). Since equally-labeled nodes of M are comparable, we can safely identify a linearization of M with a word from Σ^∞ .

A word $w \in \Sigma^\infty$ is *B-bounded* (where $B \in \mathbb{N}$) if, for any $(p, q) \in \text{Ch}$ and any finite prefix u of w , $0 \leq |u|_{p!q} - |u|_{q?p} \leq B$. An MSC M is *existentially B-bounded* if it admits a B -bounded linearization. Intuitively, this means that the MSC M can be scheduled in such a way that none of the channels (p, q) ever contains more than B pending messages.

Lemma 6.1. *A B-bounded word $w \in \Sigma^\infty$ is a linearization of some MSC M iff, for any $(p, q) \in \text{Ch}$, any finite prefix of w can be extended to a finite prefix u of w such that*

- (1) $|u|_{p!q} = |u|_{q?p}$ or
- (2) the last letter of u is $p!q$.

Proof. First suppose that w is a linearization of some MSC. Then $|w|_{p!q} = |w|_{q?p}$. If this number is finite, we can extend any finite prefix to some finite prefix satisfying (1). Otherwise, any suffix contains at least one occurrence of $p!q$, so any prefix can be extended to some larger prefix ending with $p!q$.

Conversely suppose that any finite prefix can be extended to a finite prefix satisfying (1) or (2). We construct from w an MSC as follows:

- the set of nodes equals $V = \{v \in \mathbb{N} \mid v < |w|\}$,
- for $v \in V$ let $\lambda(v) = w(v)$,
- let $(i, j) \in \text{proc}'$ iff $0 \leq i < j < |w|$ and there exists a process $p \in \mathcal{P}$ with $\lambda(i), \lambda(j) \in \Sigma_p$ and, for all k with $i \leq k < j$ and $\lambda(k) \in \Sigma_p$, we have $i = k$,
- let $(i, j) \in \text{msg}'$ iff $i, j \in V$ and there exists a channel $(p, q) \in \text{Ch}$ such that $w(i) = p!q$, $w(j) = q?p$, and $|w[0, i]|_{p!q} = |w[0, j]|_{q?p}$,
- then set $\preceq = (\text{msg}' \cup \text{proc}')^* \subseteq V^2$.

Suppose $(i, j) \in \text{msg}'$ and $j < i$. Then $|w[0, j]|_{p!q} - |w[0, j]|_{q?p} < |w[0, i]|_{p!q} - |w[0, j]|_{q?p} = 0$, contradicting the B -boundedness of w . Hence msg' and proc' are contained in \leq proving that \preceq is a partial order on V . Since \preceq is contained in the natural order \leq on the set of natural numbers V , the word w is a linearization of $M = (V, \preceq, \lambda)$. It therefore remains to be shown that M is an MSC:

- It is easily verified that $\text{msg} = \text{msg}'$ and $\text{proc} = \text{proc}'$ implying $\preceq = (\text{msg} \cup \text{proc})^*$.
- By the definition of proc' , any two nodes i and j with $P(i) = P(j)$ are ordered by \preceq .
- Let $(p, q) \in \text{Ch}$ be some channel. Since w is B -bounded, we have $|w|_{p!q} \geq |w|_{q?p}$. Now suppose $|w|_{p!q} > |w|_{q?p}$. Then there are only finitely many occurrences of $q?p$; let u_1 with $|u_1|_{p!q} - |u_1|_{q?p} > 0$ be a finite prefix of w that contains all occurrences of $q?p$. Then by our assumption on w , we can extend u_1 to a finite prefix u_2 of w whose last letter is $p!q$. Hence $|u_2|_{p!q} - |u_2|_{q?p} > |u_1|_{p!q} - |u_1|_{q?p}$. Inductively, we find a finite prefix u with $|u|_{p!q} - |u|_{q?p} > B$, contradicting the B -boundedness of w . Hence $|\lambda^{-1}(p!q)| = |w|_{p!q} = |w|_{q?p} = |\lambda^{-1}(q?p)|$ which finishes the proof that (V, \preceq, λ) is an MSC. □

We next construct, from a CFM $\mathcal{A} = (C, (\mathcal{A}_p)_{p \in \mathcal{P}}, F)$ and a bound $B \in \mathbb{N}$, a finite transition system over Σ with multiple Büchi-acceptance conditions that accepts the set of B -bounded linearizations of MSCs from $L(\mathcal{A})$:

- A configuration is a tuple $((s_p)_{p \in \mathcal{P}}, \chi, (p, q))$ with the current local states $s_p \in S_p$ for all $p \in \mathcal{P}$, the channel contents $\chi : \text{Ch} \rightarrow C^*$ with $|\chi(p', q')| \leq B$ for all $(p', q') \in \text{Ch}$, and the last active channel $(p, q) \in \text{Ch}$.
- The initial configuration is the tuple $((\iota_p)_{p \in \mathcal{P}}, \chi, (p, q))$ with $\chi(p', q') = \varepsilon$ for all $(p', q') \in \text{Ch}$, and where $(p, q) \in \text{Ch}$ is an arbitrary but fixed channel, i.e., the local machines are in their initial state and all channels are empty.
- We have a transition

$$((s_p^1)_{p \in \mathcal{P}}, \chi^1, (p^1, q^1)) \xrightarrow{a} ((s_p^2)_{p \in \mathcal{P}}, \chi^2, (p^2, q^2))$$

for an action $a \in \Sigma_p$ iff there exists a control message $c \in C$ such that

- (T1) $s_p^1 \xrightarrow{a, c} s_p^2$ is a transition of the local machine \mathcal{A}_p and $s_q^1 = s_q^2$ for $q \neq p$.
- (T2) Send events: if $a = p!q$, then $\chi^2(p, q) = \chi^1(p, q) c$ (i.e., message c is inserted into the channel from p to q) and $\chi^1(p', q') = \chi^2(p', q')$ for $(p', q') \neq (p, q)$ (i.e., all other channels are unchanged)
- (T3) Receive events: if $a = p?q$, then $\chi^1(q, p) = c \chi^2(q, p)$ (i.e., message c is deleted from the channel from q to p) and $\chi^1(q', p') = \chi^2(q', p')$ for $(q', p') \neq (q, p)$ (i.e., all other channels are unchanged)
- (T4) (p^2, q^2) is the channel that a writes to or reads from.

A finite or infinite path $((s_p^i)_{p \in \mathcal{P}}, \chi^i, (p^i, q^i))_{0 \leq i < \varkappa}$ (for some $\varkappa \in \mathbb{N} \cup \{\omega\}$) in this transition system is *successful* if

- (S1) there exists a tuple $(f_p)_{p \in \mathcal{P}} \in F$ such that, for all $p \in \mathcal{P}$ and $0 \leq i < \varkappa$, there exists $i \leq j < \varkappa$ with $s_p^j = f_p$ and
- (S2) for all $(p, q) \in \text{Ch}$ and $0 \leq i < \varkappa$, there exists $i \leq j < \varkappa$ such that $\chi^j(p, q) = \varepsilon$ or $(p^j, q^j) = (p, q)$.

Lemma 6.2. *Let $w \in \Sigma^\infty$. Then the following are equivalent:*

- (i) w is the label of some successful path in the above transition system.
- (ii) w is a B -bounded linearization of some MSC from $L(\mathcal{A})$.

Proof. To prove the implication (ii) \Rightarrow (i), let $M = (V, \preceq, \lambda) \in L(\mathcal{A})$ be an MSC accepted by \mathcal{A} , let $w \in \Sigma^\infty$ be a B -bounded linearization of M , and let (μ, ρ) be a successful run of \mathcal{A} on M . Without loss of generality, we can assume $V = \{v \in \mathbb{N} \mid 0 \leq v < |w|\}$ and $\preceq \subseteq \leq$ such that w is the sequence of labels of (V, \preceq, λ) . For $i = 0$, let $((s_p^i), \chi^i, (p^i, q^i))$ be the initial configuration of the transition system. Now let $i > 0$. For $p \in \mathcal{P}$, let $s_p^i = \iota_p$ if there is no $0 \leq j < i$ with $w(j) \in \Sigma_p$; otherwise set $s_p^i = \rho(j)$ for j the maximal natural number with $j < i$ and $w(j) \in \Sigma_p$. For $(p, q) \in \text{Ch}$, set $\chi^i(p, q) = \mu(j_1) \mu(j_2) \dots \mu(j_k)$ where $0 \leq j_1 < j_2 < \dots < j_k < i$ is the sequence of those nodes from V with $\lambda(j_\ell) = p!q$ and $\text{msg}(j_\ell) \geq i$ (since w is B -bounded, we have $0 \leq k \leq B$). Finally, (p^i, q^i) is the channel that the action $w(i-1)$ writes to or reads from. Then it can be checked that the sequence of these configurations $((s_p^i), \chi^i, (p^i, q^i))_{0 \leq i < |w|}$ forms a w -labeled path in the transition system. We show that it is successful:

- (S1) Since (ρ, μ) is successful, there exists $(f_p)_{p \in \mathcal{P}} \in F$ such that for all $p \in \mathcal{P}$ and all $v \in V$ with $\lambda(v) \in \Sigma_p$, there exists $v' \in V$ with $\lambda(v') \in \Sigma_p$, $v \preceq v'$, and $\rho(v') = f_p$ (or $f_p = \iota_p$ if no such node v exists). Now let $0 \leq i < |w|$ and let $v < i$ denote the maximal

natural number with $w(v) \in \Sigma_p$ (the case that no such number exists is left to the reader). Then there exists $v' \in V$ with $\lambda(v') \in \Sigma_p$, $v \preceq v'$, and $\rho(v') = f_p$. Because of the maximality of v , we obtain $i < v'$. Furthermore, $s_p^{v'+1} = \rho(v') = f_p$.

(S2) Let $0 \leq i < |w|$. Since w is B -bounded, the previous lemma implies the existence of $i \leq j < |w|$ such that $|w[0, j]|_{p!q} = |w[0, j]|_{q?p}$ or the last letter of $w[0, j]$ is $p!q$. Hence $\chi^{j+1}(p, q) = \varepsilon$ or $(p^{j+1}, q^{j+1}) = (p, q)$.

Conversely assume (i). Since all the channels in the transition system contain at most B messages, the word w is B -bounded. Since the w -labeled path satisfies (S2), the word w is, by the previous lemma, a linearization of some MSC. Now, using (S1) it can be verified similarly to above that this MSC is accepted by \mathcal{A} . \square

Theorem 6.3. *The following problem is PSPACE-complete:*

Input: $B \in \mathbb{N}$ (given in unary), CFM \mathcal{B} , and a global formula $\varphi \in \text{PDL}$.

Question: Is there an existentially B -bounded MSC $M \in L(\mathcal{B})$ with $M \models \varphi$?

Proof. Theorem 5.2 allows to build a CFM \mathcal{A}_φ that accepts M iff $M \models \varphi$. From Proposition 3.3, we then obtain a CFM \mathcal{A} with $L(\mathcal{A}) = L(\mathcal{B}) \cap L(\mathcal{A}_\varphi)$, i.e., $M \in L(\mathcal{A})$ iff $M \in L(\mathcal{B})$ and $M \models \varphi$. To decide the existence of some existentially B -bounded MSC in $L(\mathcal{A})$, it suffices to decide whether the above transition system has some successful path. Recall that such a path has to simultaneously satisfy $b = |\mathcal{P}| + |\text{Ch}|$ many Büchi-conditions. Extending the configurations of the transition system by a counter that counts up to $b+1$ allows to have just one Büchi-condition [Cho74]. Note that any configuration of the resulting transition system can be stored in space

$$\log(b) + |\mathcal{P}| \log n + |\text{Ch}| B \log |C| + \log |\text{Ch}|$$

where C is the set of message contents of \mathcal{A} and n is the maximal number of local states a process of \mathcal{A} has. But due to Theorem 5.2 the size of the CFM \mathcal{A}_φ is exponential in the size of φ . By Proposition 3.3, \mathcal{A} stays exponential in the size of the input. Hence, the model checking problem can be decided in polynomial space.

The hardness result follows from PSPACE-hardness of LTL model checking. \square

6.2. HMSCs vs. PDL specifications. In [Pel00], Peled provides a PSPACE model checking algorithm for high-level message sequence charts (HMSCs) against formulas of the logic TLC^- . The logic TLC^- is a fragment of our logic PDL as can be shown easily. Now, we aim to model check an HMSC against a global formula of PDL, and, thereby, to generalize Peled's result.

Definition 6.4. An *HMSC* $\mathcal{H} = (S, \rightarrow, s_0, c, \mathcal{M})$ is a finite, directed graph (S, \rightarrow) with initial node $s_0 \in S$, \mathcal{M} a finite set of finite MSCs, and a labeling function $c : S \rightarrow \mathcal{M}$.

For defining the semantics of HMSCs we need a concatenation operation. Let $M_1 = (V_1, \leq_1, \lambda_1)$ and $M_2 = (V_2, \leq_2, \lambda_2)$ be two finite MSCs over the same process set \mathcal{P} with disjoint node sets. Then $M_1 M_2 = (V, \leq, \lambda)$ is given by $V = V_1 \cup V_2$, $\lambda = \lambda_1 \cup \lambda_2$, and \leq is the least partial order containing $\leq_1 \cup \leq_2$ and $\{(v_1, v_2) \mid v_1 \in V_1, v_2 \in V_2, P(v_1) = P(v_2)\}$. Informally, the events of M_2 succeed the events of M_1 for each process, respectively.

Let $\mathcal{H} = (S, \rightarrow, s_0, c, \mathcal{M})$ be an HMSC. Let η be a maximal path of (S, \rightarrow) starting in s_0 , i.e., either a path $\eta = s_0 \rightarrow s_1 \rightarrow \dots \rightarrow s_n$ that ends in an $s_n \in S$ such that there is no $s \in S$ with $s_n \rightarrow s$ or an infinite path $\eta = s_0 \rightarrow s_1 \rightarrow \dots$. The labeling function c can

now be extended to paths by $c(\eta) = c(s_0)c(s_1)\dots$. The MSC language of the HMSC \mathcal{H} is now $L(\mathcal{H}) = \{c(\eta) \mid \eta \text{ is a maximal path starting in } s_0\}$. Note that for any HMSC \mathcal{H} the language $L(\mathcal{H})$ is existentially B -bounded for some $B \in \mathbb{N}$. Indeed, since any finite MSC M is existentially B_M -bounded for some $B_M \in \mathbb{N}$, there is a B -bounded linearization for every $c(\eta)$ when $B = \max\{B_M \mid M \in \mathcal{M}\}$.

Theorem 6.5. *The following problem is PSPACE-complete:*

Input: An HMSC \mathcal{H} and a global formula $\varphi \in \text{PDL}$.

Question: Is there an MSC $M \in L(\mathcal{H})$ with $M \models \varphi$?

Proof. Let $\mathcal{H} = (S, \rightarrow, s_0, c, \mathcal{M})$ be an HMSC. For every $s \in S$ we can find a linearization of the finite MSC $c(s)$. Now, it is easy to construct a finite (Büchi) automaton $\mathcal{S}_{\mathcal{H}}$ that accepts a linearization for each and every MSC $M \in L(\mathcal{H})$, and, vice versa, each (finite or infinite) word accepted by $\mathcal{S}_{\mathcal{H}}$ is a linearization of an $M \in L(\mathcal{H})$. Note that the size of $\mathcal{S}_{\mathcal{H}}$ is linear in the size of \mathcal{H} .

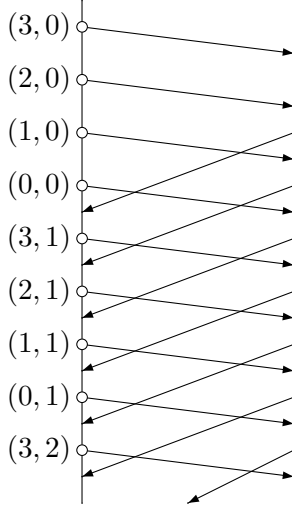
By Theorem 5.2, we can build a CFM \mathcal{A}_{φ} with $M \in L(\mathcal{A}_{\varphi})$ iff $M \models \varphi$. From \mathcal{A}_{φ} and $\mathcal{S}_{\mathcal{H}}$ (which is implicitly existentially B -bounded for some $B \in \mathbb{N}$) we construct stepwise a transition system \mathcal{S} by running \mathcal{A}_{φ} and $\mathcal{S}_{\mathcal{H}}$ simultaneously (cf. the construction before Lemma 6.2). The construction terminates because a run of $\mathcal{S}_{\mathcal{H}}$ allows for B -bounded linearizations only. A run in \mathcal{S} is successful if both projections of the run are successful. Now, \mathcal{S} admits a successful run iff there is an existentially B -bounded linearization w_M of some $M \in L(\mathcal{H}) \cap L(\mathcal{A}_{\varphi})$ (where B is implicitly given by \mathcal{H}). An analysis similar to the one in the proof of Theorem 6.3 shows that the existence of a successful path of \mathcal{S} can be decided in polynomial space.

Again, the hardness result is an easy consequence of PSPACE-hardness of LTL model checking. \square

7. PDL WITH INTERSECTION

Several extensions of PDL have been considered in the literature that still come with positive decidability results [HKT00, GLL07]. Though these results were obtained in the different context of evaluating a formula over a Kripke structure, it is natural to ask if such extensions can be handled in our setting as well. We will study here PDL with intersection (iPDL, for short), which is the canonical adaption of the logic IPDL, as defined in [HKT00], to our setting. In addition to the local formulas of PDL, we allow local formulas $\langle \pi_1 \cap \pi_2 \rangle \alpha$ where π_1 and π_2 are path expressions and α is a local formula. The intended meaning is that there exist two paths described by π_1 and π_2 respectively that both lead to the same node w where α holds.

It is the aim of this section to prove that CFMs are too weak to check all properties expressed in iPDL. To show this result more easily, we also allow atomic propositions of the form (a, b) with $a, b \in \{0, 1\}$; they are evaluated over an MSC $M = (V, \leq, \lambda)$ together with a mapping $c : V \rightarrow \{0, 1\}^2$. Then $(M, c), v \models (a, b)$ iff $c(v) = (a, b)$. Let $\mathcal{P} = \{0, 1\}$ be the set of processes. For $m \geq 1$, we first fix an MSC $M_m = (V_m, \leq, \lambda)$ for the remaining arguments: On process 0, it executes the sequence $(0!1)^m((0?1)(0!1))^\omega$. The sequence of events on process 1 is $(1?0)((1?0)(1!0))^\omega$. In other words, process 0 sends m messages to process 1 and then acknowledges any message received from 1 immediately. Differently, process 1 has a buffer for two messages. After receiving message number $k + 1$, it acknowledges message number k .

Figure 5: MSC M_4 and the mapping f .

Let $E_{0!1}$ denote the set of send-events of process 0. For the k^{th} send-event v on process 0, let $f(v) = ((m - k) \bmod m, (k - 1) \operatorname{div} m)$. Then f maps the set $E_{0!1}$ bijectively onto the grid $G_m = \{0, 1, \dots, m - 1\} \times \omega$; we denote the inverse of f by g . Figure 5 shows MSC M_4 together with the mapping f .

Lemma 7.1. *There exists a local formula α of iPDL such that, for any $m \geq 1$ and any $c : V_m \rightarrow \{0, 1\}^2$ satisfying $c(g(i, j)) = (0, 0)$ iff $i = 0$, we have $(M_m, c) \models \text{A}\alpha$ iff $c(g(i, j)) = c(g(i, j + i))$ for all $(i, j) \in G_m$.*

Proof. Let $(i, j) \in G_m$. Then observe the following:

- With π_1 denoting the path description $(\text{proc}; \{(0?1)\})^*; \text{proc}; \{(0!1)\}$, we have that $M_m, g(i, j) \models \langle \pi_1 \rangle \beta$ iff $i > 0$ and $M_m, g(i - 1, j) \models \beta$, or $i = 0$ and $M_m, g(m - 1, j + 1) \models \beta$.
- With π_2 denoting the path description $\text{msg}; \text{proc}; \text{msg}; \text{proc}$, we have $M_m, g(i, j) \models \langle \pi_2 \rangle \beta$ iff $M_m, g(i + 1, j + 1) \models \beta$ whenever $i < m - 1$.

As a consequence, we obtain

- if $i > 0$, then $M_m, g(i, j) \models \langle \pi_1; \pi_2 \rangle \beta$ iff $M_m, g(i, j + 1) \models \beta$.

Now let $c : V_m \rightarrow \{0, 1\}^2$ be a function with $c(g(i, j)) = (0, 0)$ iff $i = 0$. Then we have

- (1) $(M_m, c), g(i, j) \models \langle \{\neg(0, 0)\}; (\pi_1; \pi_2)^* \rangle \beta$ iff $i > 0$ and there exists $k \geq 0$ with $(M_m, c), g(i, j + k) \models \beta$,
- (2) $(M_m, c), g(i, j) \models \langle \{\neg(0, 0)\}; \pi_1^*; \{(0, 0)\} \rangle \beta$ iff $(M_m, c), g(0, j) \models \beta$,
- (3) $(M_m, c), g(0, j) \models \langle (\pi_2; \{\neg(0, 0)\})^* \rangle \beta$ iff there is $0 \leq k \leq m - 1$ with $(M_m, c), g(k, j + k) \models \beta$.

Now let $\pi_3 = (\{\neg(0, 0)\}; \pi_1^*; \{(0, 0)\}; (\pi_2; \{\neg(0, 0)\})^*$ and $\pi_4 = \{\neg(0, 0)\}; (\pi_1; \pi_2)^*$. Then, we have $(M_m, c), g(i, j) \models \langle \pi_3 \cap \pi_4 \rangle \beta$ iff $i > 0$ and $(M_m, c), g(i, j + i) \models \beta$. Now let

$$\alpha = ((0!1) \wedge \neg(0, 0)) \rightarrow \bigwedge_{x \in \{0, 1\}^2} x \leftrightarrow \langle \pi_3 \cap \pi_4 \rangle x .$$

Then, for all $(i, j) \in G_m$, we have $(M_m, c), g(i, j) \models \alpha$ iff $c(g(i, j)) = c(g(i, j + i))$. \square

Lemma 7.2. *Let $\mathcal{A} = (C, 2, (\mathcal{A}_p)_{p \in \mathcal{P}}, F)$ be a CFM that accepts all labeled MSCs (M_m, c) with $m \geq 1$ such that*

- (1) $c(g(i, j)) = (0, 0)$ iff $i = 0$,
- (2) $c(g(i, j)) = c(g(i, j + i))$ for all $(i, j) \in G_m$.

Then there exist $m \geq 1$ and a labeled MSC (M_m, c) accepted by \mathcal{A} , satisfying (1), and violating (2).

Proof. Let $\mathcal{A}_p = (S_p, \rightarrow_p, \iota_p)$ for $p = 0, 1$, let $m \geq 1$ be such that $|S_0| \cdot |S_1| \cdot |C|^{m-1} < 3^{\frac{(m-1)(m-2)}{2}}$, and let $M_m = (V_m, \leq, \lambda)$. Let furthermore H denote the set of mappings $c : V_m \rightarrow \{0, 1\}^2$ satisfying (1), (2), and $c(v) = (1, 1)$ for all $v \notin E_{011}$ (i.e., $\lambda(v) \neq (0!1)$). Then, for any $c \in H$, the pair (M_m, c) is accepted by \mathcal{A} – let (ρ_c, μ_c) be an accepting run of \mathcal{A} on (M_m, c) .

Let $v = g(m-1, m-2)$ and $W = \{w \in V \mid w \leq v\}$. Then, for any event $w \in W$ with $\lambda(w) = !0$, we have $\text{msg}(w) \in W$. On the other hand, there are precisely $m-1$ events $w_1, \dots, w_{m-1} \in W$ with $\lambda(w_i) = 0!1$ and $\text{msg}(w_i) \notin W$. Let furthermore $u \in W$ be the maximal event from process 1.

Consider (M_m, c_1) and (M_m, c_2) with $c_1, c_2 \in H$ and $c_1(g(i, j)) = c_2(g(i, j))$ for all $0 \leq j < i < m$. Then $c_1 = c_2$ by (2). Hence $|H|$ is the number of mappings from $\{(i, j) \mid 0 \leq j < i < m\}$ to $\{0, 1\}^2 \setminus \{(0, 0)\}$, i.e., $3^{\frac{(m-1)(m-2)}{2}}$.

Since this number exceeds $|S_0| \cdot |S_1| \cdot |C|^{m-1}$, there exist c_1 and c_2 with $c_1 \neq c_2$ in H with $\rho_{c_1}(v) = \rho_{c_2}(v)$, $\rho_{c_1}(u) = \rho_{c_2}(u)$, and $\mu_{c_1}(w_i) = \mu_{c_2}(w_i)$ for all $1 \leq i \leq m-1$.

Now define a mapping $c : V \rightarrow \{0, 1\}^2$ by $c(x) = c_1(x)$ for $x \in W$ and $c(x) = c_2(x)$ for $x \notin W$. Then, c satisfies (1) and violates (2). But (M_m, c) is accepted by \mathcal{A} : An accepting run (ρ, μ) is defined (similarly to c) by

$$\rho(x) = \begin{cases} \rho_{c_1}(x) & \text{for } x \in W \\ \rho_{c_2}(x) & \text{otherwise} \end{cases} \quad \text{and} \quad \mu(x) = \begin{cases} \mu_{c_1}(x) & \text{for } x \in W \\ \mu_{c_2}(x) & \text{otherwise.} \end{cases}$$

□

Theorem 7.3. *There exists a local formula α of iPDL such that the set of MSCs M satisfying $A\alpha$ cannot be accepted by a CFM.*

Proof. Let α be the local formula from Lemma 7.1. Towards a contradiction, assume \mathcal{A} is a CFM such that, for any pair (M, c) , we have $(M, c) \models A\alpha$ iff (M, c) is accepted by \mathcal{A} . In particular, \mathcal{A} accepts all pairs (M_m, c) satisfying (1) and (2) from Lemma 7.2. Hence there exists some pair (M_m, c) that is accepted by \mathcal{A} , satisfies (1), and violates (2). But now, by Lemma 7.1 again, $(M_m, c) \models \neg A\alpha$, contradicting our assumption on \mathcal{A} .

Using a new process 2, one can encode the mapping c by additional messages from processes 0 and 1 to process 2. □

8. OPEN QUESTIONS

The semantics of every PDL formula φ is the behavior of a CFM \mathcal{A} . Hence any PDL formula is equivalent to some formula from existential monadic second order, but a precise description of the expressive power of PDL is not known. Because of quantification over paths, it cannot be captured by first-order logic [DG04, Prop. 14]. On the other hand, PDL

is closed under negation, hence PDL is a proper fragment of existential monadic second order logic. But it is not even clear that semantical membership of this fragment is decidable.

The decidability of the model checking problem for CFMs against MSO-formulas was shown in [GKM06] for existentially B -bounded MSCs. For compositional MSCs (a mechanism for the description of sets of MSCs that is similar but more general than HMSCs) and MSO, the decidability of the model checking problem was established in [MM01]. Since the logic iPDL, i.e., PDL with intersection, can be translated effectively into an MSO-formula, the model checking problem is decidable for iPDL. However, the complexity of MSO model checking is non-elementary. Therefore, we would like to know if we can do any better for iPDL.

In PDL, we can express properties of the past and of the future of an event by taking either a backward- or a forward-path in the graph of the MSC. We are not allowed to speak about a zig-zag-path where e.g. a mixed use of proc and proc^{-1} would be possible. It is an open question whether formulas of such a “mixed PDL” could be transformed to CFMs.

REFERENCES

- [APP95] R. Alur, D. Peled, and W. Penczek. Model-checking of causality properties. In *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science (LICS 1995)*, pages 90–100, Washington, DC, USA, 1995. IEEE Computer Society.
- [BK08] B. Bollig and D. Kuske. Muller message-passing automata and logics. *Information and Computation*, 206(9-10):1084–1094, 2008.
- [BKM07] B. Bollig, D. Kuske, and I. Meinecke. Propositional dynamic logic for message-passing systems. In *Proceedings of FSTTCS’07*, volume 4855 of *Lecture Notes in Computer Science*, pages 303–315. Springer, 2007.
- [BL06] B. Bollig and M. Leucker. Message-passing automata are expressively equivalent to EMSO logic. *Theoretical Computer Science*, 358(2-3):150–172, 2006.
- [BZ83] D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2), 1983.
- [CE81] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proceedings of the Workshop on Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.
- [CGP00] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.
- [Cho74] Y. Choueka. Theories of automata on ω -tapes: a simplified approach. *Journal of Computer and System Sciences*, 8:117–141, 1974.
- [DG04] V. Diekert and P. Gastin. Local temporal logic is expressively complete for cograph dependence alphabets. *Information and Computation*, 195:30–52, 2004.
- [FL79] M.J. Fischer and R.E. Ladner. Propositional Dynamic Logic of regular programs. *J. Comput. System Sci.*, 18(2):194–211, 1979.
- [GK03] P. Gastin and D. Kuske. Satisfiability and model checking for MSO-definable temporal logics are in PSPACE. In *Proceedings of CONCUR’03*, volume 2761 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 2003.
- [GK07] P. Gastin and D. Kuske. Uniform satisfiability in PSPACE for local temporal logics over Mazurkiewicz traces. *Fundamenta Informaticae*, 80:169–197, 2007.
- [GK10] P. Gastin and D. Kuske. Uniform satisfiability problem for local temporal logics over Mazurkiewicz traces. *Information and Computation*, 208:797–816, 2010.
- [GKM06] B. Genest, D. Kuske, and A. Muscholl. A Kleene theorem and model checking algorithms for existentially bounded communicating automata. *Information and Computation*, 204:920–956, 2006.
- [GLL07] S. Göller, M. Lohrey, and C. Lutz. PDL with intersection and converse is 2EXP-complete. In *Proceedings of FoSSaCS’07*, volume 4423 of *Lecture Notes in Computer Science*, pages 198–212, 2007.

- [GMSZ02] B. Genest, A. Muscholl, H. Seidl, and M. Zeitoun. Infinite-state high-level MSCs: model-checking and realizability. In *Proceedings of ICALP'02*, volume 2380 of *Lecture Notes in Computer Science*, pages 657–668. Springer, 2002.
- [HKT00] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
- [HT97] J. G. Henriksen and P. S. Thiagarajan. A product version of dynamic linear time temporal logic. In *Proceedings of CONCUR'97*, volume 1243 of *Lecture Notes in Computer Science*, pages 45–58. Springer, 1997.
- [HT99] J. G. Henriksen and P. S. Thiagarajan. Dynamic linear time temporal logic. *Ann. Pure Appl. Logic*, 96(1-3):187–207, 1999.
- [ITU96] ITU-TS Recommendation Z.120: Message Sequence Chart 1996 (MSC96), 1996.
- [MM01] P. Madhusudan and B. Meenakshi. Beyond message sequence graphs. In *Proceedings of FSTTCS'01*, volume 2245 of *Lecture Notes in Computer Science*, pages 256–267. Springer, 2001.
- [MR00] B. Meenakshi and R. Ramanujam. Reasoning about message passing in finite state environments. In *Proceedings of ICALP'00*, volume 1853 of *Lecture Notes in Computer Science*, pages 487–498. Springer, 2000.
- [MR04] B. Meenakshi and R. Ramanujam. Reasoning about layered message passing systems. *Computer Languages, Systems, and Structures*, 30(3-4):529–554, 2004.
- [Pel00] D. Peled. Specification and verification of message sequence charts. In *Formal Techniques for Distributed System Development, FORTE/PSTV 2000*, volume 183 of *IFIP Conference Proceedings*, pages 139–154. Kluwer, 2000.
- [Pnu77] A. Pnueli. The temporal logic of programs. In *Proceedings of FOCS'77*, pages 46–57. IEEE Computer Society Press, 1977.
- [Tho90] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 133–191. Elsevier Science Publ. B.V., 1990.