Propositional Dynamic Logic for Message-Passing Systems

Benedikt Bollig[1], Dietrich Kuske[2], and Ingmar Meinecke[2]

[1] LSV, ENS Cachan, CNRS
61, Av. du Président Wilson, F-94235 Cachan Cedex, France
bollig@lsv.ens-cachan.fr
[2] Institut für Informatik, Universität Leipzig
PF 100920, D-04009 Leipzig, Germany
{kuske,meinecke}@informatik.uni-leipzig.de

Abstract. We examine a bidirectional Propositional Dynamic Logic (PDL) for message sequence charts (MSCs) extending LTL and TLC$^-$. Every formula is translated into an equivalent communicating finite-state machine (CFM) of exponential size. This synthesis problem is solved in full generality, i.e., also for MSCs with unbounded channels. The model checking problems for CFMs and for HMSCs against PDL formulas are shown to be in PSPACE for existentially bounded MSCs. It is shown that CFMs are to weak to capture the semantics of PDL with intersection.

1 Introduction

Messages sequence charts (MSCs) are an important formalism describing the executions of message-passing systems. They are a common notation in telecommunication and defined by an ITU standard [14]. A significant task is to verify system requirements. The model checking problem asks for an algorithm that decides whether, given a formula φ of a suitable logic and a finite machine $\mathcal{A}$, every behavior of $\mathcal{A}$ satisfies φ. There exist a few such suitable temporal logics. Meenakshi and Ramanujam proposed temporal logics over Lamport diagrams (which are similar to MSCs) [17, 18]. Peled [19] considered TLC$^-$ introduced in [1] for Mazurkiewicz traces. Like these logics, our logic PDL is interpreted directly over MSCs, not over linearizations; it combines elements from [18] (global next operator, past operators) and [19] (global next operator, existential interpretation of the until-operator). The ability to express properties of paths as regular expressions is also present in Henriksen and Thiagarajan's *dynamic* LTL [12, 13], an extension of LTL for traces. Differently from their approach, our path expressions are not bound to speak about the events of a single process, but they can move from one process to another. Moreover, we provide past operators to judge about events that have already been executed. We call our logic PDL as it is essentially the original propositional dynamic logic as first defined by Fischer and Ladner [8] but here in the framework of MSCs.

Already for very restrictive temporal logics, the model checking problem becomes undecidable [18]. In [19, 15, 11, 10], however, it was tackled successfully for several logics by restricting to existentially B-bounded MSCs, which can be scheduled such that the channel capacity respects a given size B. As a first step, [19, 15, 10] translate formulas into machine models such that the semantics of the formula and the machine coincide *for existentially B-bounded MSCs* (or their linearizations). In the early stages of system design it seems more natural not to fix a channel size B but to implement the entire semantics of φ. We therefore construct, from a PDL formula φ, a communicating finite-state machine (CFM, [5]) $\mathcal{A}_\varphi$ such that $L(\varphi) = L(\mathcal{A}_\varphi)$ wrt. the class of *all* (finite and infinite) MSCs.

In the literature, one finds several techniques to construct an automaton from a temporal formula: One can use a tableau construction (cf. [7]), an incremental tableau (cf. [6]), or alternating automata [20]. Here, we use an inductive method [9]: The events of an MSC are colored by additional bits, one for each subformula of φ. Then we construct, for each such subformula γ, a CFM $\mathcal{A}_\gamma$ whose task it is to check that the bit corresponding to γ is set at precisely those nodes where γ holds. For this, the CFM $\mathcal{A}_\gamma$ reads the bits corresponding to the top-level subformulas of γ. The overall CFM is obtained by running synchronously all the CFMs arising from the subformulas.

A typical subformula in PDL is $\gamma = \langle\pi\rangle\, tt$ expressing that there is a finite path starting in the current vertex that obeys the regular expression π. The construction of a CFM for such a subformula turns out to be the most difficult part. The basic idea is to start, in the current node v, a finite automaton $\mathcal{C}$ that accepts the language of π and to ensure that $\mathcal{C}$ will eventually reach an accepting state in some event v'. To ensure that this obligation is not propagated forever, we adopt and extend the solution for sequential systems [13]: The MSC is colored nondeterministically by two colors. Then a CFM checks that, along each and every path, the color changes infinitely often (this is possible although acceptance in a CFM refers to those paths that stay in one single process, only). Then the path from v to v' is allowed to change color at most once.

Altogether, we construct, for every PDL formula φ, an equivalent CFM $\mathcal{A}_\varphi$ that is exponential in the size of φ and the number of processes. Given another CFM $\mathcal{B}$, we then build a CFM $\mathcal{A}$ from $\mathcal{A}_\varphi$ and $\mathcal{B}$ with $L(\mathcal{A}) = L(\varphi) \cap L(\mathcal{B})$. Note that up to now, no restriction on the channel capacity is imposed. Finally, we decide whether $\mathcal{A}$ accepts some existentially B-bounded MSC. Only in this decision step, the bound B is used. We also show how to model check high-level MSCs (HMSCs) against PDL formulas. Both these model checking algorithms run in space polynomial in the size of the formula and of the CFM, and in the bound B. Since the logic TLC$^-$ of Peled is a fragment of PDL, we generalize the model checking result from [19] and provide a different algorithm.

By [4, 2], existential MSO logic is expressively equivalent to CFMs, and the set of CFM-languages is not closed under complementation. Since, on the other hand, PDL does not impose any restriction on the use of negation, we obtain that PDL is a proper fragment of existential MSO although this is not obvious.

The final technical section considers an enriched logic iPDL (PDL with intersection) where a node might be described by the intersection of two different paths. This extension strengthens the expressive power of the formulas. But adapting a proof technique from colored grids [16], we show that there is an iPDL-formula φ such that no CFM accepts precisely the semantics of φ.

A full version of this paper is available [3].

2 Definitions

The communication framework used in our paper is based on sequential processes that exchange asynchronously messages 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 $\mathrm{Ch} = \{(p,q) \in \mathcal{P}^2 \mid p \neq q\}$ denote the set of *channels*. Processes act by either sending a message from p to q (denoted $p!q$), or by receiving a message at p from q (denoted by $p?q$). For any process $p \in \mathcal{P}$, we define a local alphabet $\Sigma_p = \{p!q, p?q \mid q \in \mathcal{P} \setminus \{p\}\}$, and we set $\Sigma = \bigcup_{p\in\mathcal{P}} \Sigma_p$.

2.1 Message sequence charts

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 $\lambda : V \to \Sigma$ 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 define two binary relations proc and msg on V setting

- $(v, v') \in \mathrm{proc}$ 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 $v = u$,
- $(v, v') \in \mathrm{msg}$ iff there is a channel (p, q) with $\lambda(v) = p!q$, $\lambda(v') = q?p$, and $|\{u \mid \lambda(u) = p!q,\ u \leq v\}| = |\{u \mid \lambda(u) = q?p,\ u \leq v'\}|$.

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

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

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

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

2.2 Propositional dynamic logic (PDL)

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

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

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, for $\sigma \in \Sigma$, $M, v \models \sigma$ iff $\lambda(v) = \sigma$; $M, v \models \alpha_1 \vee \alpha_2$ and $M, v \models \neg\alpha$ are defined in the obvious manner. The semantics of *forward*-path expressions $\langle \pi \rangle\, \alpha$ is given by

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

The base cases for the semantics of *backward*-path expressions $\langle \pi \rangle^{-1} \alpha$ are defined similarly by

$$\begin{aligned}
M, v \models \langle \text{proc} \rangle^{-1} \alpha &\iff \text{there exists } v' \in V \text{ with } (v', v) \in \text{proc and } M, v' \models \alpha\\
M, v \models \langle \text{msg} \rangle^{-1} \alpha &\iff \text{there exists } v' \in V \text{ with } (v', v) \in \text{msg and } M, v' \models \alpha.
\end{aligned}$$

Replacing $\langle . \rangle$ with $\langle . \rangle^{-1}$ in the remaining clauses completes the definition of the semantics of local formulas.

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$^-$. In TLC$^-$, however, one cannot express properties such as "there is an even number of messages from p to q", which is certainly expressible in PDL.

Global formulas of PDL are positive Boolean combinations of formulas Eα and Aα where α is a local formula. Here, Eα expresses the existence of some node satisfying α while Aα says that α holds at all nodes. Because of this existential and universal quantification, the expressible global properties are closed under negation.

A local formula β is a *subformula* of a local formula α if it is a subformula of α (seen as Boolean combination of forward- and backward-path formulas), or if β is a subformula of some formula γ such that $\langle \pi \rangle\, \gamma$ or $\langle \pi \rangle^{-1} \gamma$ is a subformula of α or such that $\{\gamma\}$ appears in some path expression in α. We denote the set of subformulas of α by sub(α).

2.3 Communicating finite-state machines

This section defines CFMs [5], i.e., our model of a distributed system, together with its behavior.

Definition 2.2. *A* communicating finite-state machine *(*CFM*) is a tuple* $\mathcal{A} = (C, n, (\mathcal{A}_p)_{p \in \mathcal{P}}, F)$ *with* $n \geq 0$ *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* $p \in \mathcal{P}$ *with initial state* $\iota_p \in S_p$*,*
– $F \subseteq \prod_{p \in \mathcal{P}} S_p$ *is a set of global final states.*

A *run of* $\mathcal{A}$ *on* (M, c) (with $M = (V, \leq, \lambda)$ an MSC and $c : V \rightarrow \{0,1\}^n$, which can be seen as an n-tuple of mappings $V \rightarrow \{0,1\}$) is a pair 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. $\mu(v) = \mu(\mathrm{msg}(v))$ if $\mathrm{msg}(v)$ is defined,
2. $(\rho(\mathrm{proc}^{-1}(v)), \lambda(v), c(v), \mu(v), \rho(v)) \in \rightarrow_{P(v)}$ if $\mathrm{proc}^{-1}(v)$ is defined, and $(\iota_p, \lambda(v), c(v), \mu(v), \rho(v)) \in \rightarrow_{P(v)}$ otherwise.

Since, even in an infinite MSC, some of the processes may execute only finitely many events, acceptance of a run will depend on the set of states appearing cofinally [2]: let $\mathrm{cofin}_\rho(p) = \{\iota_p\}$ if $V_p = \emptyset$, and $\mathrm{cofin}_\rho(p) = \{s \in S_p \mid \forall v \in V_p\ \exists v' \in V_p : v \leq v' \wedge \rho(v') = s\}$ if $V_p \neq \emptyset$, where $V_p = P^{-1}(p)$. Then the run (ρ, μ) is *accepting* if there is some $(s_p)_{p \in \mathcal{P}} \in F$ such that $s_p \in \mathrm{cofin}_\rho(p)$ 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.

3 Translation of formulas

Let α be a local formula of PDL. We will construct a "small" CFM that accepts $(M, (c_\beta)_{\beta \in \mathrm{sub}(\alpha)})$ 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 for every position v whether $c_\sigma(v) = 1$ iff $\lambda(v) = \sigma$ and $c_\delta(v) = c_\beta(v) \vee c_\gamma(v)$, resp. Similarly, for each subformula $\neg\beta$, a sub-CFM checks $c_{\neg\beta}(v) \neq c_\beta(v)$ for each position v. While the construction of these sub-CFMs is rather straightforward, more work has to be invested for subformulas of the form $\langle\pi\rangle\,\alpha$ and $\langle\pi\rangle^{-1}\,\alpha$. Since these formulas are equivalent to $\langle\pi; \{\alpha\}\rangle\, tt$ and $\langle\pi; \{\alpha\}\rangle^{-1}\, tt$, respectively, we will only deal with the latter ones.

3.1 The backward-path automaton

Let π be a path expression, i.e., a regular expression over some alphabet $\Gamma = \{\mathrm{proc}, \mathrm{msg}, \{\alpha_1\}, \ldots, \{\alpha_n\}\}$. A word $W \in \Gamma^*$ together with a node v from an MSC M describe a path starting in that node that walks *backwards.* The letters proc and msg denote the direction of the path, the letters $\{\alpha_i\}$ denote requirements about the node currently visited, i.e., that α_i shall hold or, equivalently, that $c_i(v) = 1$ (where we write c_i instead of c_{α_i}). The existence of such a backward-path is denoted $(M, c_1, \ldots, c_n), v \models^{-1} W$.

Now let $\mathcal{C} = (Q, \iota, \delta, G)$ be a finite automaton over Γ accepting the language of the regular expression π. Then we can naturally build a first CFM $\mathcal{A}_1$ with sets of local states 2^Q such that the following are equivalent for all MSCs $M = (V, \leq, \lambda)$, all mappings $c_i : V \rightarrow \{0,1\}$, and all mappings $\rho : V \rightarrow 2^Q$:

– ρ is the state mapping of some run of $\mathcal{A}_1$ on $(M, c_1, \ldots, c_n)$
– for all $v \in V$ and $q \in Q$, we have $q \in \rho(v)$ iff there exists $W \in \Gamma^*$ with $q \xrightarrow{W}_{\mathcal{C}} G$ and $(M, c_1, \ldots, c_n), v \models^{-1} W$.

From $\mathcal{A}_1$, we obtain a CFM $\mathcal{A}_{\langle\pi\rangle^{-1}tt}$ accepting $(M, c_1, \ldots, c_n, c)$ iff $\mathcal{A}_1$ has a run on $(M, c_1, \ldots, c_n)$ such that, for all $v \in V$, we have $c(v) = 1$ iff $\iota \in \rho(v)$ (i.e., iff there exists $W \in L(\mathcal{C})$ with $(M, c_1, \ldots, c_n), v \models^{-1} W$). This construction proves

Theorem 3.1. *Let $\langle\pi\rangle^{-1} tt$ be a local formula such that π is a regular expression over the alphabet $\{\text{proc}, \text{msg}, \{\alpha_1\}, \ldots, \{\alpha_n\}\}$. Then there exists a CFM $\mathcal{A}_{\langle\pi\rangle^{-1}tt}$ with the following property: Let M be an MSC and let $c_i : V \to \{0, 1\}$ be the characteristic function of the set of positions satisfying α_i (for all $1 \leq i \leq n$). Then $(M, c_1, \ldots, c_n, c)$ is accepted by $\mathcal{A}_{\langle\pi\rangle^{-1}tt}$ iff c is the characteristic function of the set of positions satisfying $\langle\pi\rangle^{-1} tt$.*

3.2 The forward-path automaton

We now turn to a similar CFM corresponding to subformulas of the form $\langle\pi\rangle\, tt$. We will prove the following analog to Theorem 3.1. This proof will, however, be substantially more difficult.

Theorem 3.2. *Let $\langle\pi\rangle\, tt$ be a local formula such that π is a regular expression over the alphabet $\Gamma = \{\text{proc}, \text{msg}, \{\alpha_1\}, \ldots, \{\alpha_n\}\}$. Then there exists a CFM $\mathcal{A}_{\langle\pi\rangle tt}$ with the following property: Let M be an MSC and let $c_i : V \to \{0, 1\}$ be the characteristic function of the set of positions satisfying α_i (for all $1 \leq i \leq n$). Then $(M, c_1, \ldots, c_n, c)$ is accepted by $\mathcal{A}_{\langle\pi\rangle tt}$ iff c is the characteristic function of the set of positions satisfying $\langle\pi\rangle\, tt$.*

The rest of this section is devoted to the proof of this theorem. Let $\mathcal{C} = (Q, \iota, T, G)$ be a finite automaton over Γ that accepts the language of the regular expression π.

Let $W \in \Gamma^*$, $M = (V, \leq, \lambda)$ an MSC, and $v \in V$. These data describe a *forward*-path starting in v where the letters proc and msg denote the direction and the letters $\{\alpha_i\}$ requirements on the current node (i.e., that α_i shall hold). We denote the existence of such a forward path with $(M, c_1, \ldots, c_n), v \models W$.

In order to prove Theorem 3.2, it therefore suffices to construct a CFM that accepts $(M, c_1, \ldots, c_n, c)$ iff

$$\begin{aligned} &\forall v \in V : c(v) = 0 \implies \forall W \in L(\mathcal{C}) : (M, c_1, \ldots, c_n), v \not\models W && (1)\\ \wedge\ &\forall v \in V : c(v) = 1 \implies \exists W \in L(\mathcal{C}) : (M, c_1, \ldots, c_n), v \models W. && (2) \end{aligned}$$

Since the class of languages accepted by CFMs is closed under intersection, we can handle the two implications separately (cf. Prop. 3.3 and 3.6 below).

Proposition 3.3. *There exists a CFM $\mathcal{A}_0$ that accepts $(M, c_1, \ldots, c_n, c)$ iff (1) holds.*

Proof. 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 the language of the regular expression 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 and since $\mathcal{C}$ is nondeterminsitic, the set of local states S_p equals $2^{Q\setminus G}$ with initial state $\iota_p = \emptyset$ for any $p \in \mathcal{P}$. □

It remains to construct a CFM that checks (2). Again, the basic idea is simple: whenever the CFM encounters a node v with $c(v) = 1$ (i.e., a node that is supposed to satisfy $\langle \pi \rangle\, tt$), it will start the automaton $\mathcal{C}$ (that accepts the language of the regular expression π) and check that it can reach an accepting state along one of the possible paths. Before, we had to prevent $\mathcal{C}$ from reaching an accepting state. This time, we have to ensure that any verification of a claim $c(v) = 1$ will eventually result in an accepting state being reached.

To explain our construction, suppose $M = (V, \leq, \lambda)$ to be an MSC and $c_1, \ldots, c_n, c : V \to \{0, 1\}$ to be mappings. In order to verify (2), any node $v \in V$ with $c(v) = 1$ forms an obligation, namely the obligation to find a word $W \in L(\mathcal{C})$ such that $(M, c_1, \ldots, c_n), v \models W$. This obligation is either satisfied immediately or propagated to the successors of v, i.e., to the nodes $\mathrm{proc}(v)$ or $\mathrm{msg}(v)$ (provided, they exist). Thus, every node from V obtains a set O of obligations in the form of states of the finite automaton $\mathcal{C}$. The crucial point now is to ensure that none of these obligations is propagated forever. To this aim, the set of obligations is divided into two sets O_1 and O_2. In general, the obligations from O_1 at node v are satisfied or propagated to the obligations from O_1 at the node $\mathrm{msg}(v)$ or $\mathrm{proc}(v)$. Similarly, obligations from O_2 are propagated to O_2; in addition, newly arising obligations (in the form of nodes v with $c(v) = 1$) are moved into O_2. The only exception from this rule is the case $O_1 = \emptyset$, i.e., all "active" obligations are satisfied. In this case, all of O_2 can be moved to O_1. Then, the run of the CFM is accepting iff, along *each* path in the MSC, the exceptional rule is applied cofinally.

The problem arising here is that the success of a run of a CFM refers to paths along a *single* process, only. Hence, infinite paths that change process infinitely often cannot be captured directly. A solution is to guess an additional 0-1-coloring c_0 such that no path can stay in one color forever, and to allow a color change only if the exceptional rule is applied.

Thus, we are left with the task to construct a CFM accepting (M, c_0) if no infinite path in M stays monochromatic eventually (it is actually sufficient to accept only such pairs, but not necessarily all, but sufficiently many). To achieve this goal, we proceed as follows.

Let M be an MSC and $c_0 : V \to \{0, 1\}$. On V, we define an equivalence relation $\sim$ whose equivalence classes are the maximal monochromatic intervals on a process line.

Let Col be the set of all pairs (M, c_0) with $c_0 : V \to \{0, 1\}$ such that the following hold

(1) if v is minimal on its process, then $c_0(v) = 1$

(2) if $(v, v') \in \mathrm{msg}$ and $w' \leq v'$ with $P(w') = P(v')$, then there exists $(u, u') \in \mathrm{msg}$ with $\lambda(u') = \lambda(v')$, $c_0(u) = c_0(u')$, and $u' \sim w'$
(3) any equivalence class of $\sim$ is finite.

In general, there can be messages $(u, u'') \in \mathrm{msg}$ such that the colors of u and u'' are different, i.e., $c_0(u) \neq c_0(u'')$. Condition (2) ensures that there are also many messages (u, u') with $c_0(u) = c_0(u')$. More precisely, looking at the event w' on process q, process q will receive in the future a message from process p (at the event v'). Then the requirement is that process q receives some message from process p (a) in the $\sim$-equivalence class of w' such that (b) sending and receiving of this message have the same color.

Given the above conditions (1–3), it is almost immediate to check that Col can be accepted by some CFM:

Lemma 3.4. *There exists a CFM $\mathcal{A}_{\mathrm{Col}}$ that accepts the set* Col.

The main consequence of (1–3) is the following whose proof is elementary but not trivial:

Lemma 3.5. *Let $(M, c_0) \in \mathrm{Col}$ and let $(v_1, v_2, \dots)$ be some infinite path in M. Then there exist infinitely many $i \in \mathbb{N}$ with $c_0(v_i) \neq c_0(v_{i+1})$.*

Proof. The crucial point is the following: Let $(v, v') \in \mathrm{msg}$ be some message such that the numbers of mutually non-equivalent nodes on the process lines before v and v', resp., are different. Then one obtains $c_0(v) \neq c_0(v')$. □

These two lemmas and the ideas explained above prove

Proposition 3.6. *There exists a CFM $\mathcal{A}_1$ that accepts $(M, c_1, \dots, c_n, c)$ iff (2) holds.*

3.3 The overall construction

Theorem 3.7. *Let α be a local formula of* PDL. *Then one can construct a CFM $\mathcal{B}$ such that (M, c) is accepted by $\mathcal{B}$ iff $c : V \to \{0, 1\}$ is the characteristic function of the set of positions that satisfy α.*

Proof. One first constructs a CFM $\mathcal{A}$ that accepts $(M, (c_\beta)_{\beta \in \mathrm{sub}(\alpha)})$ iff

(1) $c_\sigma(v) = 1$ iff $\lambda(v) = \sigma$ for all $v \in V$ and $\sigma \in \mathrm{sub}(\alpha) \cap \Sigma$
(2) $c_{\gamma \vee \delta}(v) = \max(c_\gamma(v), c_\delta(v))$ for all $v \in V$ and $\gamma \vee \delta \in \mathrm{sub}(\alpha)$
(3) $c_{\neg\gamma}(v) \neq c_\gamma(v)$ for all $v \in V$ and $\neg\gamma \in \mathrm{sub}(\alpha)$
(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 formulas $\langle\pi\rangle\gamma \in \mathrm{sub}(\alpha)$ where $\alpha_1, \dots, \alpha_n$ are those local formulas for which $\{\alpha_i\}$ appears in the path expression π (cf. Theorem 3.2)
(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 $\langle\pi\rangle^{-1}\gamma \in \mathrm{sub}(\alpha)$ where $\alpha_1, \dots, \alpha_n$ are those local formulas for which $\{\alpha_i\}$ appears in the path expression π (cf. Theorem 3.1).

This can be achieved since the intersection of CFM-languages can be accepted by a CFM. The CFM $\mathcal{B}$ guesses the missing labelings c_β for $\beta \in \mathrm{sub}(\alpha) \setminus \{\alpha\}$ and simulates $\mathcal{A}$. $\square$

Recall that a global formula is a positive Boolean combination of formulas of the form $\mathrm{E}\alpha$ and $\mathrm{A}\alpha$ where α is a local formula. Note that the sets of pairs (M, c) with $c(v) = 1$ for at least one event (for all events, resp.) $v \in V$ can be accepted by CFMs. This, together with a careful analysis of the size of the CFMs constructed so far, leads to the following corollary:

Corollary 3.8. *Let φ be a global formula of PDL. Then one can construct a CFM $\mathcal{A}$ that accepts M iff $M \models \varphi$. The numbers of local states and of control messages of $\mathcal{A}$ belong to $2^{O((|\varphi|+|\mathcal{P}|)^2)}$.*

4 Model checking

4.1 CFMs vs. PDL specifications

We aim at an algorithm that decides whether, given a global formula φ and a CFM $\mathcal{A}$, every MSC from $L(\mathcal{A})$ satisfies φ. The undecidability of this problem can be shown following, e.g., the proof in [18] (the ideas from that paper can easily be transferred to our setting from Lamport diagrams and the fragment LD_0 of PDL). To gain decidability, we follow the successful approach of, e.g., [15, 11, 10], and restrict attention to existentially B-bounded MSCs from $L(\mathcal{A})$.

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, which we identify with a finite or infinite word from Σ^∞.

A word $w \in \Sigma^\infty$ is *B-bounded* (where $B \in \mathbb{N}$) if, for any $(p, q) \in \mathrm{Ch}$ and any prefix u of w, $0 \leq |u|_{p!q} - |u|_{q?p} \leq B$ where $|u|_\sigma$ denotes the number of occurrences of σ in u. An MSC M is *existentially B-bounded* if it admits a B-bounded linearization.

The CFM $\mathcal{A}$ can be translated into a finite transition system that accepts precisely the B-bounded linearizations of MSCs accepted by $\mathcal{A}$. Any configuration of this transition system consists of

- the buffer contents (i.e., $|\mathrm{Ch}|$ many words over C of length at most B),
- a local state per process,
- one channel (i.e., a pair from Ch),
- a global state that is accepting in $\mathcal{A}$, and
- a counter whose value is bounded by $|\mathrm{Ch}| + |\mathcal{P}|$ in order to handle multiple Büchi-conditions.

Hence a single configuration can be stored in space $O(\log(|\mathcal{P}|+|\mathrm{Ch}|)+|\mathcal{P}|\log n+|\mathrm{Ch}|B\log|C| + \log|\mathrm{Ch}|)$ where n is the number of local states per process. This therefore also describes the space requirement for deciding whether the CFM $\mathcal{A}$ accepts at least one existentially B-bounded MSC.

Since the number of local states per process as well as that of messages of the CFM in Cor. 3.8 is exponential, we obtain the following result on the model checking of a CFM vs. a PDL specification:

Theorem 4.1. *The following is PSPACE-complete:*
Input: $B \in \mathbb{N}$ (given in unary), CFM $\mathcal{B}$, and a global formula $\varphi \in$ PDL.
Question: Is there an existentially B-bounded MSC $M \in L(\mathcal{B})$ with $M \models \varphi$?

Hardness follows from PSPACE-hardness of LTL-model checking.

4.2 HMSCs vs. PDL specifications

In [19], Peled provides a PSPACE model checking algorithm for high-level message sequence charts (HMSCs) against formulas of the logic TLC^{-}, a fragment of PDL. Now, we aim to model check an HMSC against a global formula of PDL, and, thereby, to generalize Peled's result.

Definition 4.2. *An* HMSC $\mathcal{H} = (S, \rightarrow, s_0, \ell, \mathcal{M})$ *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 $\ell : S \rightarrow \mathcal{M}$.*

To define the semantics of an HMSC $\mathcal{H}$, one replaces the MSCs $\ell(s)$ by an arbitrary linearization and then concatenates the words along a maximal initial path in $\mathcal{H}$. Then an MSC M is accepted by $\mathcal{H}$ (i.e., belongs to $L(\mathcal{H})$) if one of its linearizations belongs to this word language $L \subseteq \Sigma^{\infty}$. Note that there is necessarily some $B \in \mathbb{N}$ such that all words in L are B-bounded. Furthermore, this number B can be computed from $\mathcal{H}$. Now construct, as above, from the CFM $\mathcal{A}$ from Cor. 3.8 the finite transition system that accepts all B-bounded linearizations of MSCs satisfying the global formula φ. Considering the intersection of the language of this transition system with L allows us to prove

Theorem 4.3. *The following problem is PSPACE-complete:*
Input: An HMSC $\mathcal{H}$ and a global formula $\varphi \in$ PDL.
Question: Is there an MSC $M \in L(\mathcal{H})$ with $M \models \varphi$?

5 PDL with intersection

PDL with intersection (or iPDL) allows, besides the local formulas of PDL, also 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. We show that this extends the expressive power of PDL beyond that of CFMs.

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 $(1?0)\,((1?0)\,(1!0))^{\omega}$ (cf. Fig. 1). The send-events on process 0 are named in $\{0, 1, \ldots, m-1\} \times \omega$ as indicated in Fig. 1. Let $\mathcal{M}$ denote the set of pairs (V_m, c) with $c : V_m \rightarrow \{0, 1\}^2$ such that $c(i, j) = 0$ iff $i = 0$.

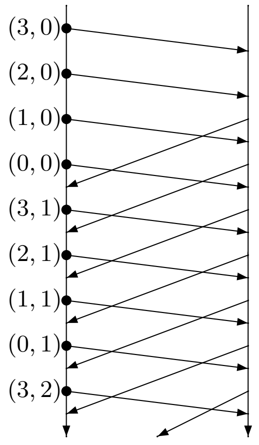

Fig. 1. MSC M_4 and the mapping f.

Then one can construct a local formula α such that, for any $(M, c) \in \mathcal{M}$, we have $(M, c) \models \mathrm{A}\alpha$ iff $c(i, j) = c(i, j + i)$ for all suitable pairs (i, j). Now suppose $\mathcal{A} = (C, 2, (\mathcal{A}_p)_{p \in \mathcal{P}}, F)$ to be a CFM that accepts all labeled MSCs $(M_m, c) \in \mathcal{M}$ satisfying $c(i, j) = c(i, j + i)$ for all suitable (i, j). Then $\mathcal{A}$ also accepts some labeled MSC $(M, c) \in \mathcal{M}$ that violates this condition. It follows

Theorem 5.1. *There exists a local formula α of* iPDL *such that the set of MSCs satisfying* $\mathrm{A}\alpha$ *cannot be accepted by a CFM.*

6 Open questions

Since the semantics of every PDL formula φ is the behavior of a CFM, it is equivalent with some formula from existential monadic second-order logic [4, 2]. Since PDL is closed under negation, it is a proper fragment of existential monadic second order logic. Because of quantification over paths, it cannot be captured by first-order logic. We do not know if first-order logic is captured by PDL nor do we have any precise description of its expressive power.

Since the logic iPDL, i.e., PDL with intersection, can be translated effectively into MSO, the model checking problem for CFMs and existentially B-bounded MSCs is decidable for iPDL [10]. 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 and what the complexity of the model checking would be.

References

1. 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.
2. B. Bollig and D. Kuske. Distributed Muller automata and logics. Research Report LSV-06-11, Laboratoire Spécification et Vérification, ENS Cachan, France, 2006.
3. B. Bollig, D. Kuske, and I. Meinecke. Propositional dynamic logic for message-passing systems. Research Report LSV-07-22, Laboratoire Spécification et Vérification, ENS Cachan, France, 2007. `http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/PDF/rr-lsv-2007-22.pdf`.
4. B. Bollig and M. Leucker. Message-passing automata are expressively equivalent to EMSO logic. *Theoretical Computer Science*, 358(2-3):150–172, 2006.
5. D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2), 1983.
6. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.
7. E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, chapter 16*, pages 995–1072. Elsevier Publ. Co., Amsterdam, 1990.
8. M.J. Fischer and R.E. Ladner. Propositional Dynamic Logic of regular programs. *J. Comput. System Sci.*, 18(2):194–211, 1979.
9. P. Gastin and D. Kuske. Satisfiability and model checking for MSO-definable temporal logics are in PSPACE. In *CONCUR'03*, Lecture Notes in Comp. Science vol. 2761, pages 222–236. Springer, 2003.
10. 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.
11. B. Genest, A. Muscholl, H. Seidl, and M. Zeitoun. Infinite-state high-level MSCs: model-checking and realizability. In *ICALP'02*, Lecture Notes in Comp. Science vol. 2380, pages 657–668. Springer, 2002.
12. J. G. Henriksen and P. S. Thiagarajan. A product version of dynamic linear time temporal logic. In Antoni Mazurkiewicz and Józef Winkowski, editors, *8th International International Conference on Concurrency Theory (CONCUR 1997)*, volume 1243, pages 45–58, Warsaw, Poland, 1–4 1997. Springer-Verlag.
13. J. G. Henriksen and P. S. Thiagarajan. Dynamic linear time temporal logic. *Ann. Pure Appl. Logic*, 96(1-3):187–207, 1999.
14. ITU-TS Recommendation Z.120: Message Sequence Chart 1996 (MSC96), 1996.
15. P. Madhusudan and B. Meenakshi. Beyond message sequence graphs. In *FSTTCS 2001*, Lecture Notes in Comp. Science vol. 2245, pages 256–267. Springer, 2001.
16. O. Matz and W. Thomas. The monadic quantifier alternation hierarchy over graphs is infinite. In *LICS'97*, pages 236–244. IEEE Computer Society Press, 1997.
17. B. Meenakshi and R. Ramanujam. Reasoning about message passing in finite state environments. In *Proc. of ICALP 2000*, volume 1853 of *Lecture Notes in Computer Science*, pages 487–498. Springer, 2000.
18. B. Meenakshi and R. Ramanujam. Reasoning about layered message passing systems. *Computer Languages, Systems, and Structures*, 30(3-4):529–554, 2004.
19. 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.
20. M.Y. Vardi. Nontraditional applications of automata theory. In *TACS'94*, Lecture Notes in Computer Science vol. 789, pages 575–597. Springer, 1984.