ilmedia

TECHNISCHE UNIVERSITÄT ILMENAU

Roy Mennicke:

Propositional Dynamic Logic with Converse and Repeat for Message-Passing Systems

Original published in:
Logical Methods in Computer Science, 9 (2013), 2, paper 12, 35 p.
ISSN Online: 1860-5974
DOI: 10.2168/LMCS-9(2:12)2013
URL: http://www.lmcs-online.org/ojs/viewarticle.php?id=1298
(Visited: 2013-08-05)

Logical Methods in Computer Science
Vol. 9(2:12)2013, pp. 1–35
www.lmcs-online.org

Submitted Dec. 21, 2012
Published Jun. 28, 2013

PROPOSITIONAL DYNAMIC LOGIC WITH CONVERSE AND REPEAT FOR MESSAGE-PASSING SYSTEMS

ROY MENNICKE

Ilmenau University of Technology, Germany
e-mail address: roy.mennicke@tu-ilmenau.de

Abstract. The model checking problem for propositional dynamic logic (PDL) over message sequence charts (MSCs) and communicating finite state machines (CFMs) asks, given a channel bound B, a PDL formula φ and a CFM $\mathcal{C}$, whether every existentially B-bounded MSC M accepted by $\mathcal{C}$ satisfies φ. Recently, it was shown that this problem is PSPACE-complete. In the present work, we consider CRPDL over MSCs which is PDL equipped with the operators converse and repeat. The former enables one to walk back and forth within an MSC using a single path expression whereas the latter allows to express that a path expression can be repeated infinitely often. To solve the model checking problem for this logic, we define message sequence chart automata (MSCAs) which are multi-way alternating parity automata walking on MSCs. By exploiting a new concept called concatenation states, we are able to inductively construct, for every CRPDL formula φ, an MSCA precisely accepting the set of models of φ. As a result, we obtain that the model checking problem for CRPDL and CFMs is still in PSPACE.

1. Introduction

Automatic verification is the process of translating a computer system to a mathematical model, formulating a requirements specification in a formal language, and automatically checking the obtained model against this specification. In the past, finite automata, Kripke structures, and Büchi automata turned out to be suitable formalisms to model the behavior of complex non-parallel systems. Two of the most common specification languages are the temporal logics LTL [21] and CTL [3]. After deciding on a modeling and a specification formalism, automatic verification melts down to the *model checking problem*: Given a model $\mathcal{A}$ with behavior $L(\mathcal{A})$ and a specification φ representing the expected behavior $L(\varphi)$, does $L(\mathcal{A}) \subseteq L(\varphi)$ hold?

Distributed systems exchanging messages can be modeled by communicating finite-state machines (CFMs) which were introduced in [2]. A CFM consists of a finite number of finite automata communicating using FIFO channels. Each run of such a machine can be understood as a message sequence chart (MSC). The latter is an established ITU standard and comes with a formal definition as well as a convenient graphical notation. In a simplified

2012 ACM CCS: [**Theory of computation**]: Logic—Verification by model checking.
Key words and phrases: message sequence charts, alternating automata, communicating finite-state machines, propositional dynamic logic.

DOI:10.2168/LMCS-9(2:12)2013

model, an MSC can be considered as a structure consisting of send and receive events which are assigned to unique processes where the events of each process are linearly ordered. For every send event, there exists a matching receive event and vice versa. Unfortunately, the model checking problem for CFMs is undecidable even for very simple temporal logics – this is a direct consequence of the undecidability of the emptiness problem for CFMs. One solution to this problem is to establish a bound B on the number of messages pending on a channel. The bounded model checking problem of CFMs then reads as follows: given a channel bound B, a specification φ and a CFM $\mathcal{C}$, does every existentially B-bounded MSC M accepted by $\mathcal{C}$ satisfy φ? An existentially B-bounded MSC is an MSC which admits an execution with B-bounded channels. Using this approach several results for different temporal logics were obtained in [16, 10, 9, 1].

In [1], a bidirectional propositional dynamic logic (PDL) was proposed for the automatic verification of distributed systems modeled by CFMs. This logic was originally introduced by Fischer and Ladner [5] for Kripke structures and allows to express fundamental properties in an easy and intuitive manner. PDL for MSCs is closed under negation, it is a proper fragment of the existential monadic second-order logic (EMSO) in terms of expressiveness (but it is no syntactic fragment) [1], and the logic TLC$^-$ considered by Peled [20] is a fragment of it. PDL distinguishes between local and global formulas. The former ones are evaluated at a specific event of an MSC whereas the latter are Boolean combinations of local formulas quantifying existentially over all events of an MSC. Consider for example the local formula $\alpha = p!q \wedge \neg \langle \mathsf{proc}^* \rangle\, p?q$. An event satisfies α if it is a send event of a message from process p to q which is not followed by a reply message from q to p. The global formula $\mathsf{E}\alpha$ expresses that there exists such an event v.

By a rather involved translation of PDL formulas into CFMs, Bollig, Kuske, and Meinecke demonstrated in [1] that the bounded model checking problem for CFMs and PDL can be decided in polynomial space. However, by means of this approach, Bollig et al. were not able to support the popular converse operator. The latter, introduced in [22], is an extension of PDL which allows to walk back and forth within an MSC using a single path expression of PDL. For example one can specify a path expression $(\mathsf{proc}^{-1}; \mathsf{msg})^*$ describing "zigzag-like" paths going back on a process and traversing a send event in an alternating manner. It is an open question whether PDL formulas enriched with the converse operator can be translated into CFMs. Bollig et al. only managed to provide an operator which enables path expressions to either walk backward or forward.

In the present work, we consider CRPDL over MSCs which is PDL equipped with the operators converse ($\_^{-1}$) and repeat ($\_^{\omega}$) [23]. The latter allows to express that a path expression can be repeated infinitely often. For example, an event v on process p satisfies $\langle \mathsf{proc} \rangle^{\omega}$ if there are infinitely many events on p succeeding v. We are able to demonstrate that the bounded model checking problem of CFMs and CRPDL is in PSPACE and therefore generalize the model checking result from [1]. In order to obtain this result, we define multi-way alternating parity automata over MSCs which we call local message sequence chart automata (or local MSCAs for short). Local MSCAs are started at specific events of an MSC and accept sets of pointed MSCs which are pairs of an MSC M and an event v of M. Using a game theoretic approach, it can be shown that local MSCAs are closed under complementation. We demonstrate that every local formula α of CRPDL can be translated in polynomial space into an equivalent local MSCA whose size is linear in the size of α — this can be done independently from any channel bound. We also define global MSCAs consisting of a local MSCA $\mathcal{M}$ and a set of global initial states. A global initial

state is a tuple of states $(\iota_1, \iota_2, \ldots, \iota_n)$ where n is the number of processes. If, for every process p, there exists an accepting run of $\mathcal{M}$ starting in the minimal event of p and the initial state ι_p, then the global MSCA accepts the whole MSC. For every global formula φ, we can construct in polynomial space a global MSCA $\mathcal{G}$ such that $\mathcal{G}$ precisely accepts the set of models of φ. After fixing a channel bound B, the automaton $\mathcal{G}$ is then transformed into a two-way alternating word automaton and, after that, into a Büchi automaton recognizing the set of all B-bounded linearizations of the models of φ.

In the literature, one can basically find two types of approaches to turn a temporal formula into a Büchi automaton. On the one hand, Vardi and others [25, 8] transformed LTL formulas into alternating automata in one single step and, afterwards, these alternating automata were translated into Büchi automata. On the other hand, there were performed inductive constructions which lead to a Büchi automaton without the need for an intermediate step [13, 6, 7, 1]. In the present work, we combine these two approaches to obtain a very modular and easy to understand proof. For a given CRPDL formula, we inductively construct an alternating automaton which is later translated into a Büchi automaton. In this process, we utilize a new concept called concatenation states. These special states allow the concatenation of local MSCAs. For example, if $\mathcal{M}$ is the local MSCA obtained for the formula $\langle \mathsf{proc} \rangle\, \mathsf{tt}$, then we can concatenate two copies of $\mathcal{M}$ to obtain an automaton for the formula $\langle \mathsf{proc}; \mathsf{proc} \rangle\, \mathsf{tt}$.

Outline. We proceed as follows. In Sect. 2, we define MSCs, CRPDL, MSCAs, and give introductory examples. In Sect. 3, we show that local MSCAs are effectively closed under complementation. In Sect. 4, we construct, for every local CRPDL formula α, a local MSCA which precisely accepts the models of α. In Sect. 5, we effectively show that, for every global CRPDL formula φ, the set of models of φ is the language of a global MSCA. In the sections 6 and 7, we prove that the bounded satisfiability problem for CRPDL and the bounded model checking problem for CRPDL and CFMs both are PSPACE-complete.

A conference version of this paper was published as [18].

Acknowledgements. The author likes to express his sincere thanks to his doctoral adviser Dietrich Kuske for his guidance and valuable advice. Furthermore, he is grateful to Benedikt Bollig for comments leading to a considerable technical simplification. This paper also greatly benefits from the detailed reviews and helpful remarks of the anonymous referees.

2. Preliminaries

We let $\mathsf{poly}(n)$ denote the set of polynomial functions in one argument. For every natural number $n \geq 1$, we set $[n] = \{1, 2, \ldots, n\}$.

We fix a finite set $\mathbb{P} = \{1, 2, \ldots, |\mathbb{P}|\}$ of processes. Let $\mathsf{Ch} = \{(p, q) \in \mathbb{P}^2 \mid p \neq q\}$ denote the set of *communication channels*. For all $p \in \mathbb{P}$, we define a local alphabet $\Sigma_p = \{p!q, p?q \mid q \in \mathbb{P} \setminus \{p\}\}$ which we use in the following way. An event labelled by $p!q$ marks the send event of a message from process p to process q whereas $p?q$ is the label of a receive event of a message sent from q to p. We set $\Sigma = \bigcup_{p \in \mathbb{P}} \Sigma_p$. Since $\mathbb{P}$ is finite, the local alphabets Σ_p and Σ are also finite.

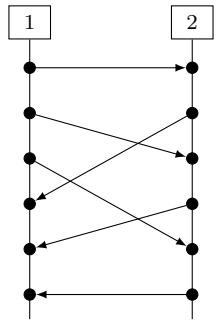

Figure 1: An example of a finite MSC.

2.1. **Message Sequence Charts.** Message sequence charts model the behavior of a finite set of parallel processes communicating using FIFO channels. The following example shows that they come with a convenient graphical representation.

Example 2.1. Figure 1 shows a finite MSC M over the set of processes $\mathbb{P} = \{1, 2\}$.

In the graphical representation of an MSC M over the set of processes $\mathbb{P}$, there is a vertical axis for every process from $\mathbb{P}$. On the edge for process $p \in \mathbb{P}$, the events occurring on p are drawn as small black circles. Thus, a linear ordering $\preceq_p^M$ on the set of events from process p is implicitly defined. In the formal definition of an MSC, which we give in the following, the direct successor relation induced by the linear ordering $\preceq_p^M$ is given by proc_p^M. For technical convenience, we force processes to contain at least one event. Messages sent between two processes are depicted by arrows pointing from the send event to the matching receive event. Formally, messages are represented by the binary relation msg^M and, for every send event, there exists a matching receive event and vice versa.

Definition 2.2. A *message sequence chart* (*MSC*) is a structure

$$M = \left(V^M, (\mathsf{proc}_p^M)_{p \in \mathbb{P}}, \mathsf{msg}^M, \lambda^M\right)$$

where

- V^M is a set of *events*,
- $\mathsf{proc}_p^M, \mathsf{msg}^M \subseteq (V^M \times V^M)$ for all $p \in \mathbb{P}$,
- $\lambda^M \colon V^M \to \Sigma$ is a labeling function,
- for all $p \in \mathbb{P}$, the relation proc_p^M is the direct successor relation of a linear order $\preceq_p^M$ on $V_p^M := \{v \in V^M \mid \lambda(v) \in \Sigma_p\}$,
- $(V_p^M, \preceq_p^M)$ is non-empty and finite or isomorphic to $(\mathbb{N}, \leq)$,
- for all $v, w \in V^M$, we have $(v, w) \in \mathsf{msg}^M$ if and only if there exists $(p, q) \in \mathsf{Ch}$ such that $\lambda^M(v) = p!q$, $\lambda^M(w) = q?p$, and
$$|\{u \in V^M \mid \lambda^M(u) = p!q, u \preceq_p^M v\}| = |\{u \in V^M \mid \lambda^M(u) = q?p, u \preceq_q^M w\}|,$$
- for every $v \in V^M$ there exists $w \in V^M$ such that $(v, w) \in \mathsf{msg} \cup \mathsf{msg}^{-1}$.

If $v \in V^M$, then we denote by $P_M(v)$ the process at which v is located, i.e., $P_M(v) = p$ if and only if $\lambda^M(v) \in \Sigma_p$. Finally, if $v \in V^M$, then the pair (M, v) is called a *pointed MSC*.

Definition 2.3. We fix the set $\mathbb{M} = \{\mathsf{proc}, \mathsf{proc}^{-1}, \mathsf{msg}, \mathsf{msg}^{-1}, \mathsf{id}\}$ of directions. An MSC M induces a partial function $\eta_M : (V^M \times V^M) \to \mathbb{M}$. For all $v, v' \in V^M$, we define

$$\eta_M(v, v') = \begin{cases} \mathsf{proc} & \text{if } (v, v') \in \mathsf{proc}^M \\ \mathsf{proc}^{-1} & \text{if } (v', v) \in \mathsf{proc}^M \\ \mathsf{msg} & \text{if } (v, v') \in \mathsf{msg}^M \\ \mathsf{msg}^{-1} & \text{if } (v', v) \in \mathsf{msg}^M \\ \mathsf{id} & \text{if } v = v' \\ \text{undefined} & \text{otherwise} \end{cases}$$

where $\mathsf{proc}^M = \bigcup_{p \in \mathbb{P}} \mathsf{proc}_p^M$.

2.2. **Propositional Dynamic Logic with Converse and Repeat.** In this section, we introduce a new logic called propositional dynamic logic with converse and repeat (or CRPDL for short). In CRPDL, we distinguish between *local* and *global* formulas. The former ones are evaluated at specific events of an MSC. The latter are positive Boolean combinations of properties of the form "there exists an event satisfying a local formula" or "all events satisfy a local formula".

Definition 2.4. *Local formulas* α and *path expressions* π of CRPDL are defined by the following grammar, where $D \in \mathbb{M}$ and σ ranges over the alphabet Σ:

$$\alpha ::= \sigma \mid \neg\alpha \mid \langle\pi\rangle\, \alpha \mid \langle\pi\rangle^\omega$$
$$\pi ::= D \mid \{\alpha\} \mid \pi;\pi \mid \pi + \pi \mid \pi^*$$

Formulas of the form $\langle\pi\rangle\, \alpha$ are called *path formulas*. The *size* of a local formula α is the length of the string α.

Note that proc^{-1} and msg^{-1} form the converse operator [22] which allows to walk back and forth within an MSC using a single path expression. The formula $\langle\pi\rangle^\omega$ provides the functionality of the repeat operator [23]. It allows to express that a path expression can be repeated infinitely often.

Intuitively, a path formula $\langle\pi\rangle\, \alpha$ expresses that one can move along a path described by π and then α holds. In the following formal definition of the semantics of local formulas, we write $\mathsf{reach}_M(v, \pi)$ to denote the set of events which can be reached from v using a path described by π. A formal definition of $\mathsf{reach}_M(v, \pi)$ is given at the end of Definition 2.5.

Definition 2.5. Let (M, v) be a pointed MSC, $\sigma \in \Sigma$, $D \in \mathbb{M}$, α be a local formula, π, π_1, π_2 be path expressions. We define:

$$\begin{aligned}
M, v \models \sigma &\iff \lambda^M(v) = \sigma \\
M, v \models \neg\alpha &\iff M, v \not\models \alpha \\
M, v \models \langle D \rangle\, \alpha &\iff \text{there exists } v' \text{ with } \eta_M(v, v') = D \text{ and } M, v' \models \alpha \\
M, v \models \langle \{\alpha\} \rangle\, \beta &\iff M, v \models \alpha \text{ and } M, v \models \beta \\
M, v \models \langle \pi_1 + \pi_2 \rangle\, \alpha &\iff M, v \models \langle \pi_1 \rangle\, \alpha \text{ or } M, v \models \langle \pi_2 \rangle\, \alpha \\
M, v \models \langle \pi_1 ; \pi_2 \rangle\, \alpha &\iff M, v \models \langle \pi_1 \rangle\, \langle \pi_2 \rangle\, \alpha \\
M, v \models \langle \pi^* \rangle\, \alpha &\iff \text{there exists an } n \geq 0 \text{ with } M, v \models (\langle \pi \rangle)^n \alpha \\
M, v \models \langle \pi \rangle^\omega &\iff \text{there exist infinitely many events } v_0, v_1, \ldots \text{ such that} \\
&\qquad\qquad v_0 = v \text{ and } v_{i+1} \in \mathsf{reach}_M(v_i, \pi) \text{ for all } i \geq 0
\end{aligned}$$

where $\mathsf{reach}_M(v, \pi)$ is inductively defined as follows:

$$\mathsf{reach}_M(v, D) = \begin{cases} \{v'\} & \text{if } \eta_M(v, v') = D \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathsf{reach}_M(v, \{\alpha\}) = \begin{cases} \{v\} & \text{if } M, v \models \alpha \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathsf{reach}_M(v, \pi_1 ; \pi_2) = \bigcup_{v' \in \mathsf{reach}_M(v, \pi_1)} \mathsf{reach}_M(v', \pi_2)$$

$$\mathsf{reach}_M(v, \pi_1 + \pi_2) = \mathsf{reach}_M(v, \pi_1) \cup \mathsf{reach}_M(v, \pi_2)$$

$$\mathsf{reach}_M(v, \pi^*) = \{v\} \cup \bigcup_{n \geq 1} \mathsf{reach}_M(v, \pi^n)$$

By $L(\alpha)$ we denote the set of pointed MSCs which satisfy α.

We set $\mathsf{tt} = \sigma \vee \neg\sigma$ for some $\sigma \in \Sigma$. If $\alpha = \langle \pi \rangle\, \mathsf{tt}$, then we define

$$\mathsf{reach}_M(v, \alpha) = \mathsf{reach}_M(v, \pi)\,.$$

Furthermore, we use $\alpha_1 \wedge \alpha_2$ as an abbreviation for $\langle \{\alpha_1\} \rangle\, \alpha_2$ and write $\alpha_1 \vee \alpha_2$ for the formula $\neg(\neg\alpha_1 \wedge \neg\alpha_2)$. Finally, for all $q \in \mathbb{P}$, we define $P_q = \bigvee_{p \in \mathbb{P}, q \neq p} (q!p \vee q?p)$. For every pointed MSC (M, v), we have $M, v \models P_q$ if and only if $P_M(v) = q$.

Remark 2.6. It can be easily seen that $M, v \models \langle \pi \rangle\, \alpha$ if and only if $M, v \models \langle \pi ; \{\alpha\} \rangle\, \mathsf{tt}$. Because of this fact, every time we are dealing with path formulas in the future, we will assume that $\alpha = \mathsf{tt}$.

Example 2.7. The existential until construct $\alpha \mathsf{EU} \beta$ [12] can be expressed by the local formula $\langle (\{\alpha\}; (\mathsf{proc} + \mathsf{msg}))^* \rangle\, \beta$.

We now define global formulas which are positive Boolean combinations of properties of the form "there exists an event satisfying a local formula α" or "all events satisfy a local formula α".

Definition 2.8. The syntax of *global formulas* is given by the grammar

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

where α ranges over the set of local formulas. Their semantics is as follows: If M is an MSC, α is a local formula, and φ_1, φ_2 are global formulas, then

$$\begin{aligned} M \models \mathsf{E}\alpha &\iff \text{there exists } v \in V^M \text{ with } M, v \models \alpha\,, \\ M \models \mathsf{A}\alpha &\iff M, v \models \alpha \text{ for all } v \in V^M\,, \\ M \models \varphi_1 \vee \varphi_2 &\iff M \models \varphi_1 \text{ or } M \models \varphi_2\,, \text{ and} \\ M \models \varphi_1 \wedge \varphi_2 &\iff M \models \varphi_1 \text{ and } M \models \varphi_2\,. \end{aligned}$$

We define the *size* of a global formula φ to be the length of the string φ. By $L(\varphi)$, we denote the set of MSCs M with $M \models \varphi$.

Note that even though there are no negation operators allowed in global formulas, the expressible properties are still closed under negation. This is because conjunction and disjunction operators as well as existential and universal quantification are available.

Example 2.9 ([1])**.** Let $\beta_p = \langle \mathsf{proc}^*; \mathsf{msg}; \mathsf{proc}^*; \mathsf{msg} \rangle\, P_p$. If (M, v) is a pointed MSC such that $M, v \models \beta_p$, then process p can be reached from v with exactly two messages. If M is the MSC from Fig. 1, then $M, v \models \beta_1$ if and only if v is one of the first three events on process 1. The global formula $\varphi_p = \mathsf{A}\beta_p$ states that β_p holds for every event of an MSC M (which in particular implies that M is infinite).

Example 2.10. An MSC M satisfies $\mathsf{E} \bigwedge_{p \in \mathbb{P}} (\langle (\mathsf{proc} + \mathsf{msg} + \mathsf{proc}^{-1} + \mathsf{msg}^{-1})^* \rangle\, P_p)$ if and only if the graph $(V^M, \mathsf{proc}^M \cup \mathsf{msg}^M \cup (\mathsf{proc}^M)^{-1} \cup (\mathsf{msg}^M)^{-1})$ is connected.

Example 2.11. Now, let $\pi_p = ((\mathsf{proc} + \mathsf{msg})^*; \{P_p\})$ for every $p \in \mathbb{P}$. Imagine that M is an MSC which models the circulation of a single token granting access to a shared resource. Then $M \models \mathsf{E} \langle \pi_1; \pi_2; \ldots; \pi_{|\mathbb{P}|} \rangle^\omega$ if and only if no process ever gets excluded from using the shared resource.

2.3. **Message Sequence Chart Automata (MSCA).** In this section, we give the definition of MSCAs which basically are multi-way alternating parity automata walking forth and back on the process and message edges of MSCs. We first define local MSCAs which are started at individual events of an MSC. They also come with a so called concatenation state. This type of state is used to concatenate local MSCAs in order to obtain more complex local MSCAs. Using this technique, we will show in a subsequent section that every local formula of CRPDL can be transformed into a local MSCA.

Definition 2.12. If X is a non-empty set, then $\mathcal{B}^+(X)$ denotes the *set of all positive Boolean expressions over* X together with the expression $\bot$. The latter expression is always evaluated to false. We say that $Y \subseteq X$ is a *model* of $E \in \mathcal{B}^+(X)$ and write $Y \models E$ if E is evaluated to true when assigning true to every element contained in Y and assigning false to all other elements from $X \setminus Y$. The set $Y \subseteq X$ is a *minimal* model of E if $Y \models E$ and $Z \not\models E$ for all $Z \subsetneq Y$. We denote the set of all models of E by $\mathsf{mod}(E)$ whereas we write $[\![E]\!]$ for the set of all minimal models of E.

For instance, $\{a, b, c\}$, $\{a, b\}$, $\{a, c\}$, $\{b, c\}$, and $\{a\}$ are all models of the positive Boolean expression $a \vee (b \wedge c) \in \mathcal{B}^+(\{a, b, c\})$. However, only $\{a\}$, and $\{b, c\}$ are minimal models.

Definition 2.13. A *local message sequence chart automaton (local MSCA)* is a quintuple $\mathcal{M} = (S, \delta, \iota, c, \kappa)$ where

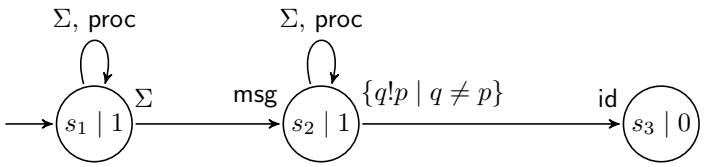

Figure 2: The local MSCA $\mathcal{M}$ from Example 2.14.

- S is a finite set of states,
- $\delta\colon (S \times \Sigma) \to \mathcal{B}^+(\mathbb{M} \times S)$ is a transition function,
- $\iota \in S$ is an initial state,
- $c \in S$ is a concatenation state, and
- $\kappa : S \to \mathbb{N}$ is a ranking function.

The *size* of $\mathcal{M}$ is $|S| + |\delta|$. If we do not pay attention to the concatenation state c, then we sometimes write $(S, \delta, \iota, \kappa)$ instead of $(S, \delta, \iota, c, \kappa)$. If $s \in S$, $\sigma \in \Sigma$, and $\tau \in [\![\delta(s, \sigma)]\!]$, then τ is called a *transition*.

For example, the transition $\tau = \{(\mathsf{proc}, s_1), (\mathsf{msg}, s_2)\}$ which is a minimal model of the expression $(\mathsf{proc}, s_1) \wedge ((\mathsf{msg}, s_2) \vee (\mathsf{msg}^{-1}, s_2))$ can be interpreted in the following way: Let us assume that $\mathcal{M}$ is in state $s \in S$ at an event v. If it performs the transition τ, then it changes, in parallel, from the state s into the states s_1 and s_2, i.e., the run splits. In the case of state s_1, it moves to the event succeeding the event v on the current process. For s_2, the automaton walks along a message edge to the receive event of the message sent in v. Hence, the conjunctive connectives implement *universal branching* whereas the disjunctive connectives realize *existential branching* and nondeterminism, respectively. As a consequence, local MSCAs are alternating automata and their runs may split. Therefore, in order to be able to define runs of local MSCAs, we first introduce labelled trees.

Later, in the construction of local MSCAs from local formulas, the concatenation state c of $\mathcal{M}$ will be used to concatenate local MSCAs for simple local formulas in order to obtain automata which are equivalent to more complex formulas.

Example 2.14. Let $p \in \mathbb{P}$ be fixed. Consider the local MSCA $\mathcal{M} = (S, \delta, s_1, \kappa)$ which is depicted in Fig. 2. Its set of states S consists of the three states s_1, s_2, and s_3 where s_1 is the initial state. Each state is depicted by a circle. The label of the circles also tells us the rank of each state. For example, $s_1 \mid 1$ expresses that $\kappa(s_1) = 1$. Furthermore, we have $\kappa(s_2) = 1$ and $\kappa(s_3) = 0$. Transitions are depicted by arrows. For instance, the arrow from s_1 to s_2 labelled by Σ and msg says that the automaton can make a transition from s_1 to s_2 by following a message edge and going to the matching receive event, respectively. We write Σ because this transition can be executed no matter what the label of the current event is. Alternatively, the automaton can stay in state s_1 by going to the successor of the current event — this is expressed by the loop at s_1. More formally, for all $\sigma \in \Sigma$, we have $\delta(s_1, \sigma) = (\mathsf{proc}, s_1) \vee (\mathsf{msg}, s_2)$, $\delta(s_3, \sigma) = \bot$, and

$$\delta(s_2, \sigma) = \begin{cases} (\mathsf{id}, s_3) & \text{if } \sigma = q!p \text{ where } q \in \mathbb{P} \setminus \{p\} \\ (\mathsf{proc}, s_2) & \text{otherwise.} \end{cases}$$

Note that the above example makes use of existential branching only whereas the MSCA of the next example also implements universal branching.

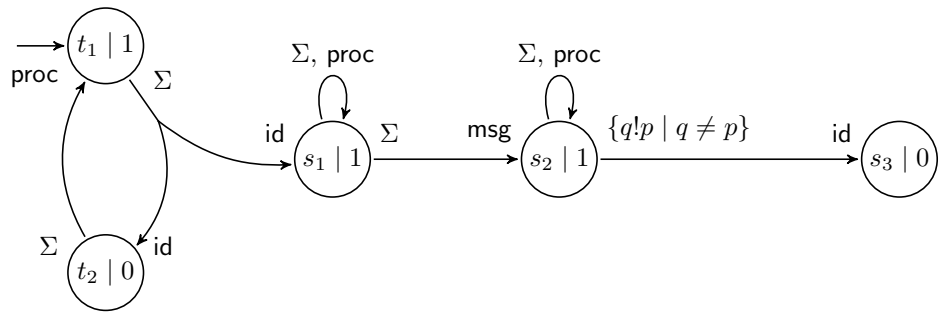

Figure 3: The local MSCA $\mathcal{M}'$ from Example 2.15.

Example 2.15. Consider the local MSCA from Fig. 3. Note that universal branching is depicted by forked arrows. We have $\mathcal{M}' = (S \cup \{t_1, t_2\}, \delta \cup \delta', t_1, \kappa \cup \kappa')$ where $\mathcal{M} = (S, \delta, s_1, \kappa)$ is the local MSCA from Example 2.14, $\kappa'(t_1) = 1$, $\kappa'(t_2) = 0$, and $\delta(t_1, \sigma) = (\mathsf{id}, t_2) \wedge (\mathsf{id}, s_1)$ and $\delta(t_2, \sigma) = (\mathsf{proc}, t_1)$ for all $\sigma \in \Sigma$.

Definition 2.16. A *tree* is a directed, connected, cycle-free graph (C, E) with the set of nodes C and the set of edges E such that there exists exactly one node with no incoming edges (which is called *root*) and all other nodes have exactly one incoming edge.

We now define so-called S-labelled trees over pointed MSCs where S is an arbitrary set. Later, the set S will be the set of states of a local MSCA.

Definition 2.17. Let S be an arbitrary set, M be an MSC, and $v \in M$. An *S-labelled tree over* (M, v) is a quintuple $\rho = (C, E, r, \mu, \nu)$ where

(1) (C, E) is a tree with root r,
(2) $\mu : C \to S$ is a labeling function,
(3) $\nu : C \to V^M$ is a positioning function with $\nu(r) = v$,
(4) $\mu(y_1) \neq \mu(y_2)$ or $\nu(y_1) \neq \nu(y_2)$ for all $(x, y_1), (x, y_2) \in E$ with $y_1 \neq y_2$, and
(5) $\eta_M(\nu(x), \nu(y))$ is defined for all $(x, y) \in E$.

The elements of C are called *configurations*. If $x \in C$, then $E_\rho(x) = \{y \in C \mid (x, y) \in E\}$ denotes the set of the direct successor configurations of x in ρ. For convenience, we identify μ with its natural extension, i.e., $\mu(x_1 x_2 x_3 \ldots) = \mu(x_1)\mu(x_2)\mu(x_3) \ldots \in S^* \cup S^\omega$.

We use S-labelled trees to define runs of local MSCAs. The condition (4) has no influence on the expressiveness of local MSCAs but simplifies the proofs in Section 4. Intuitively, it prevents a local MSCA from doing unnecessary work. By item (5), we ensure that an MSCA cannot jump within an MSC but must move along process or message edges.

Definition 2.18. Let S be a set, (M, v) be a pointed MSC, and $\rho = (C, E, r, \mu, \nu)$ be an S-labelled tree over (M, v). A *path* in ρ of length $n \in \mathbb{N} \cup \{\omega\}$ is a sequence $x_1 x_2 x_3 \ldots \in C^n$ such that $x_{i+1} \in E_\rho(x_i)$ for all $1 \leq i < n$. It is a *branch of* ρ if $x_1 = r$ and $E_\rho(x_n) = \emptyset$ (provided that $n \in \mathbb{N}$).

That means every branch of ρ begins in the root of ρ and either leads to some leaf of ρ or is infinite.

Definition 2.19. If $C' \subseteq C$ such that $(C', E \cap (C' \times C'))$ is a tree with root r', we denote by $\rho \restriction C'$ the *restriction of* ρ *to* C', i.e., the S-labelled tree (C', E', r', μ', ν') where $E' = E \cap (C' \times C')$, $\mu' = \mu \restriction C'$, and $\nu' = \nu \restriction C'$.

We want the runs of local MSCAs to be maximal. That means that, during a run, a local MSCA is forced to execute a transition if it is able to do so. If the MSCA is unable to proceed, we say that it is stuck.

Definition 2.20. Let M be an MSC and $\mathcal{M} = (S, \delta, \iota, \kappa)$ be a local MSCA. The automaton $\mathcal{M}$ is *stuck* at $v \in V^M$ in the state $s \in S$ if for every transition $\tau \in [\![\delta(s, \lambda^M(v))]\!]$ there exists a movement $(D, s') \in \tau$ such that there exists no event $v' \in V^M$ with $\eta_M(v, v') = D$.

We are now prepared to define runs of local MSCAs.

Definition 2.21. Let $\mathcal{M} = (S, \delta, \iota, \kappa)$ be a local MSCA and $\rho = (C, E, r, \mu, \nu)$ be an S-labelled tree over a pointed MSC (M, v). We define $\mathsf{tr}_\rho : C \to 2^{\mathbb{M} \times S}$ to be the function which maps every $x \in C$ to the set

$$\left\{ \left(\eta_M(\nu(x), \nu(x')), \mu(x')\right) \mid x' \in E_\rho(x) \right\} .$$

The tree ρ is a *run* of $\mathcal{M}$ on (M, v) if $\mu(r) = \iota$ and, for all $x \in C$, the *run condition* is fulfilled, i.e.,

- if $E_\rho(x) \neq \emptyset$, then $\mathsf{tr}_\rho(x) \in [\![\delta(\mu(x), \lambda^M(\nu(x)))]\!]$, and
- if $E_\rho(x) = \emptyset$, then $\mathcal{M}$ is stuck at the event $\nu(x)$ in state $\mu(x)$.

Definition 2.22. Let $(s_i)_{i \geq 1} \in S^* \cup S^\omega$ be a sequence of states. By $\mathsf{inf}((s_i)_{i \geq 1})$, we denote the set of states occurring infinitely often in $(s_i)_{i \geq 1}$. If $(s_i)_{i \geq 1}$ is finite, then it is *accepting* if it ends in a state s whose rank $\kappa(s)$ is even. If it is infinite, it is accepting if the minimum of the ranks of all states occurring infinitely often is even, i.e., $\min\{\kappa(s) \mid s \in \mathsf{inf}((s_i)_{i \geq 1})\}$ is even.

If ρ is a run of $\mathcal{M}$, and b is a branch of ρ, then b is *accepting* if its label $\mu(b)$ is accepting. A run ρ of $\mathcal{M}$ is *accepting* if every branch of ρ is accepting. By $L(\mathcal{M})$, we denote the set of all pointed MSCs (M, v) for which there exists an accepting run of $\mathcal{M}$. Furthermore, for all $p \in \mathbb{P}$, $L_p(\mathcal{M})$ is the set of MSCs M with $(M, v) \in L(\mathcal{M})$ where v is the minimal element from V_p^M with respect to $\preceq_p^M$.

Example 2.23. Let $\mathcal{M}$ and $\mathcal{M}'$ be the MSCAs from the Examples 2.14 and 2.15, respectively. It can be easily checked that, for every pointed MSC (M, v), we have $(M, v) \in L(\mathcal{M})$ if and only if $M, v \models \beta_p$ where $\beta_p = \langle \mathsf{proc}^*; \mathsf{msg}; \mathsf{proc}^*; \mathsf{msg} \rangle\, P_p$ is the formula from Example 2.9. In contrast, a pointed MSC (M, v) is accepted by $\mathcal{M}'$ if and only if $M, v' \models \beta_p$ for all $v' \in V^M$ with $v \preceq_p^M v'$.

We also introduce the notion of global MSCAs which come as a local MSCA together with a set of global initial states.

Definition 2.24. A *global message sequence chart automaton* (global MSCA) is a tuple $\mathcal{G} = (\mathcal{M}, I)$ where $\mathcal{M} = (S, \delta, \iota, \kappa)$ is a local MSCA and $I \subseteq S^{|\mathbb{P}|}$ is a set of global initial states. The language of $\mathcal{G}$ is defined by

$$L(\mathcal{G}) = \bigcup_{(s_1, \ldots, s_{|\mathbb{P}|}) \in I} \bigcap_{p \in \mathbb{P}} L_p(S, \delta, s_p, \kappa) .$$

The size of $\mathcal{G}$ is the size of $\mathcal{M}$.

Intuitively, an MSC M is accepted by $\mathcal{G}$ if and only if there exists a global initial state $(s_1, s_2, \ldots, s_{|\mathbb{P}|}) \in I$ such that, for every $p \in \mathbb{P}$, the local MSCA $\mathcal{M}$ accepts (M, v_p) when started in the state s_p where v_p is the minimal event on process p.

Example 2.25. Let $\mathcal{G} = (\mathcal{M}', \{(t_1, \ldots, t_1)\})$ be the global MSCA where $\mathcal{M}'$ is the local MSCA from Example 2.15. We have $M \in L(\mathcal{G})$ if and only if $M \models \varphi_p$ where φ_p is the global formula from Example 2.9.

3. Closure under Complementation

If $\mathcal{M}$ is a local MSCA, then a local MSCA $\mathcal{M}^\#$ recognizing the complement of $L(\mathcal{M})$ can be easily obtained. Basically one just needs to exchange $\wedge$ and $\vee$ in the image of the transition function of $\mathcal{M}$ and update the ranking function.

To make this more precise, let us first define the dual expression $\widetilde{E}$ of a positive Boolean expression E.

Definition 3.1. Let X be a set and $E \in \mathcal{B}^+(X)$. Then the *dual expression* $\widetilde{E}$ *of* E denotes the positive Boolean expression obtained by exchanging $\wedge$ and $\vee$ in E.

Let us state the following two easy lemmas on positive Boolean expressions and their dual counterparts.

Lemma 3.2. *Let X be a set and $E \in \mathcal{B}^+(X)$. Then, for all $Y \in \mathsf{mod}(E)$ and $Z \in \mathsf{mod}(\widetilde{E})$, we have $Y \cap Z \neq \emptyset$.*

Proof. If $E = a$ for some $a \in X$, then the lemma easily follows. For the induction step, let us assume that $E = E_1 \wedge E_2$ such that, for all $i \in [2]$, $Y \in \mathsf{mod}(E_i)$, and $Z \in \mathsf{mod}(\widetilde{E}_i)$, we have $Y \cap Z \neq \emptyset$. If $Y \in \mathsf{mod}(E)$ and $Z \in \mathsf{mod}(\widetilde{E})$, then, without loss of generality, $Y \models E_1$ and $Z \models \widetilde{E}_1$. It follows from the induction hypothesis that $Y \cap Z \neq \emptyset$. The case $E = E_1 \vee E_2$ is shown analogously. □

Lemma 3.3. *Let X be a set and $E \in \mathcal{B}^+(X)$. If $Z \subseteq X$ such that $Z \cap Y \neq \emptyset$ for all $Y \in \mathsf{mod}(E)$, then $Z \in \mathsf{mod}(\widetilde{E})$.*

Proof. If $E = a$ for some $a \in X$, then the lemma easily follows. For the induction step, let $E_1, E_2 \in \mathcal{B}^+(X)$ such that, for all $i \in [2]$, the following holds: if $Z \subseteq X$ and $Z \cap Y \neq \emptyset$ for all $Y \in \mathsf{mod}(E_i)$, then $Z \in \mathsf{mod}(\widetilde{E}_i)$.

For the case $E = E_1 \vee E_2$, let $Z \subseteq X$ such that $Z \cap Y \neq \emptyset$ for all $Y \in \mathsf{mod}(E)$. If $i \in [2]$ and $Y \in \mathsf{mod}(E_i)$, then $Y \models E$. Hence, $Z \cap Y \neq \emptyset$ for all $i \in [2]$ and $Y \in \mathsf{mod}(E_i)$. From our induction hypothesis it follows that $Z \models \widetilde{E}_1$ and $Z \models \widetilde{E}_2$ and, therefore, $Z \models \widetilde{E}$. Now, let us consider the case $E = E_1 \wedge E_2$. Towards a contradiction, suppose that there exists a $Z \subseteq X$ such that $Z \cap Y \neq \emptyset$ for all $Y \in \mathsf{mod}(E)$ and $Z \not\models \widetilde{E}$. Since $\widetilde{E} = \widetilde{E}_1 \vee \widetilde{E}_2$, we have $Z \not\models \widetilde{E}_1$ and $Z \not\models \widetilde{E}_2$. From our induction hypothesis it follows that there exist $Y_1 \in \mathsf{mod}(E_1)$ and $Y_2 \in \mathsf{mod}(E_2)$ with $Z \cap Y_1 = Z \cap Y_2 = \emptyset$. Since we also have $Y_1 \cup Y_2 \models E$, this is a contradiction to our definition of Z. □

We are now prepared to dualize local MSCAs.

Definition 3.4. Let $\mathcal{M} = (S, \delta, \iota, c, \kappa)$ be a local MSCA. The *dual MSCA* $\mathcal{M}^\#$ is the local MSCA $(S, \delta^\#, \iota, c, \kappa^\#)$ where

- $\kappa^\#(s) = \kappa(s) + 1$ for all $s \in S$ and
- $\delta^\#(s, \sigma) = \widetilde{\delta(s, \sigma)}$ for all $s \in S$ and $\sigma \in \Sigma$.

Remark 3.5. Let $\mathcal{M} = (S, \Delta, \iota, \kappa)$ be a local MSCA and $(s_i)_{i\geq 1} \in S^\infty$ be a sequence of states. Because of our definition of $\kappa^\#$, a state $s \in S$ has an even rank in $\mathcal{M}$ if and only if it has an odd rank in $\mathcal{M}^\#$. It follows that $(s_i)_{i\geq 1}$ is accepting in $\mathcal{M}$ if and only if it is not accepting in $\mathcal{M}^\#$.

If (M, v) is a pointed MSC, ρ is a run of $\mathcal{M}$ on (M, v), and $\rho^\#$ is a run of $\mathcal{M}^\#$ on (M, v), then one can observe that ρ contains a branch $x_1x_2x_3\ldots$ and $\rho^\#$ contains a branch $x'_1x'_2x'_3\ldots$ such that $\mu(x_1x_2x_3\ldots) = \mu(x'_1x'_2x'_3\ldots)$, i.e. they are labelled by the same sequence of states. Because of the fact stated in Remark 3.5, $x_1x_2x_3\ldots$ is accepting in $\mathcal{M}$ if and only if $x'_1x'_2x'_3\ldots$ is not accepting in $\mathcal{M}^\#$. By means of this observation, a result on parity games, and the ideas presented in [19], we prove the following theorem:

Theorem 3.6. *If $\mathcal{M}$ is a local MSCA and (M, v) is a pointed MSC, then*

$$(M, v) \in L(\mathcal{M}) \iff (M, v) \notin L(\mathcal{M}^\#)\,.$$

The rest of this section prepares the proof of the above theorem. The actual proof can be found on page 15.

Definition 3.7. Let $\mathcal{M} = (S, \delta, \iota, c, \kappa)$ be a local MSCA and (M, v) be a pointed MSC. With $\mathcal{M}$ and the pointed MSC (M, v), we associate a *game* $G(\mathcal{M}, M, v)$ played by the two players *Automaton* and *Pathfinder* in the arena (C_A, C_P, E_A, E_P) where $C_A = V^M \times S$, $C_P = V^M \times 2^{\mathbb{M}\times S}$, $E_A \subseteq C_A \times C_P$, $E_P \subseteq C_P \times C_A$,

$$\big((v, s), (v, \tau)\big) \in E_A \iff \tau \in [\![\delta(s, \lambda^M(v))]\!] \text{ and, for all } (D, s') \in \tau, \text{ there exists an event } v' \in V^M \text{ such that } \eta_M(v, v') = D\,,$$

and

$$\big((v, \tau), (v', s)\big) \in E_P \iff \text{there exists } D \in \mathbb{M} \text{ such that } (D, s) \in \tau \text{ and } \eta_M(v, v') = D\,.$$

C_A is the set of *game positions* of the player Automaton. Analogously, at a position from C_P it is Pathfinder's turn. The game position (v, ι) is called the *initial position*.

A *play* of $G(\mathcal{M}, M, v)$ starts at the initial position (v, ι) from the set C_A, i.e., the player Automaton has to move first. He chooses a transition τ from $[\![\delta(\iota, \lambda^M(v))]\!]$ resulting in a game position $(v, \tau) \in C_P$. Now, it is Pathfinder's turn who has to pick a movement (D, s) from τ. This leads to the game position $(v', s) \in C_A$ with $\eta_M(v, v') = D$. After that, Automaton has to move next and so on. More formally, we define:

Definition 3.8. Let $\mathcal{M} = (S, \delta, \iota, \kappa)$ be a local MSCA and (M, v) be a pointed MSC. A *partial play* ξ of $G(\mathcal{M}, M, v)$ is a sequence of one of the following two forms:

(1) $\xi = \big((v_i, s_i)(v_i, \tau_i)\big)_{1\leq i\leq n} \in (C_AC_P)^n$ where
- $n \geq 1$
- $(v_1, s_1) = (v, \iota)$
- $((v_i, s_i), (v_i, \tau_i)) \in E_A$ for all $1 \leq i \leq n$
- $((v_i, \tau_i), (v_{i+1}, s_{i+1})) \in E_P$ for all $1 \leq i < n$

(2) $\xi = \big((v_i, s_i)(v_i, \tau_i)\big)_{1\leq i<n}(v_n, s_n) \in (C_PC_A)^{n-1}C_A$ where
- $n \geq 1$
- $(v_1, s_1) = (v, \iota)$
- $((v_i, s_i), (v_i, \tau_i)) \in E_A$ for all $1 \leq i < n$
- $((v_i, \tau_i), (v_{i+1}, s_{i+1})) \in E_P$ for all $1 \leq i < n$

The sequence $(s_i)_{1\leq i\leq n} \in S^n$ is called the *label* of ξ. By $\xi \restriction C_A$ we denote the sequence $(v_1, s_1)(v_2, s_2)\dots$ which is obtained by restricting ξ to the positions from C_A.

The sequence $\xi = (v_1, s_1)\big((v_i, \tau_i), (v_{i+1}, s_{i+1})\big)_{1\leq i<n}$ with $n \in (\mathbb{N} \setminus \{0\}) \cup \{\infty\}$ is a *play* of $G(\mathcal{M}, M, v)$ if the following conditions are fulfilled:

- $\big((v_i, s_i)(v_i, \tau_i)\big)_{1\leq i\leq j}(v_{j+1}, s_{j+1}) \in (C_A C_P)^j C_A$ is a partial play for all $0 \leq j < n$
- if $n \in \mathbb{N}$, then there does not exit a $(v, \tau) \in C_P$ such that $\big((v_n, s_n), (v, \tau)\big) \in E_A$, i.e., the player Automaton cannot move any more.

If ξ is a play, then the label of ξ is the sequence $s_1 s_2 s_3 \dots \in S^\infty$. Automaton is declared the *winner* of the play ξ if the label of ξ is accepting in $\mathcal{M}$. Otherwise, the play is won by Pathfinder.

We define (memoryless) (winning) strategies in the usual way:

Definition 3.9. A strategy of player Automaton in the game $G(\mathcal{M}, M, v)$ is a total function $f : ((C_A C_P)^* C_A) \to C_P$. A (partial) play $\xi = (v_1, s_1)\big((v_i, \tau_i)(v_{i+1}, s_{i+1})\big)_{1\leq i<n}$ is called a *(partial) f-play* if $(v_i, \tau_i) = f\big((v_1, s_1)(v_1, \tau_1)\dots(v_i, s_i)\big)$ for every $1 \leq i < n$. The strategy f is called *memoryless* if $f(\xi_1) = f(\xi_2)$ for all $(v, s) \in C_A$ and $\xi_1, \xi_2 \in (C_A C_P)^*\{(v, s)\}$. Furthermore, f is a *winning strategy* if every f-play of $G(\mathcal{M}, M, v)$ is won by the player Automaton — no matter what the moves of Pathfinder are.

A (memoryless) (winning) strategy for player Pathfinder is defined analogously. Note that we can consider a memoryless strategy f as a function $f : C_A \to C_P$. Even though we require a strategy to be a total function, we often define a concrete strategy only partially and assume that all other (uninteresting) values are mapped to a fixed game position from C_P.

In the following, let $\mathcal{M} = (S, \delta, \iota, \kappa)$ be an MSCA and (M, v) be a pointed MSC. Furthermore, let (C_A, C_P, E_A, E_P) be the arena of $G(\mathcal{M}, M, v)$ and $(C_A, C_P^\#, E_A^\#, E_P^\#)$ be the arena of $G(\mathcal{M}^\#, M, v)$. Firstly, let us state the fact that parity games enjoy memoryless determinacy.

Proposition 3.10 ([15])**.** *From any game position in $G(\mathcal{M}, M, v)$, either Automaton or Pathfinder has a memoryless winning strategy.*

We now establish a connection between accepting runs of $\mathcal{M}$ and winning strategies of the player Automaton.

Lemma 3.11. *If (M, v) is accepted by $\mathcal{M}$, then Automaton has a winning strategy in the game $G(\mathcal{M}, M, v)$.*

Proof. Let $\rho = (C, E, r, \mu, \nu)$ be an accepting run of $\mathcal{M}$ on (M, v). We construct a strategy f for Automaton which ensures that, for every f-play ξ of $G(\mathcal{M}, M, v)$, the label of ξ is also a label of a branch of ρ. Let $x \in C$ be a configuration with $E_\rho(x) \neq \emptyset$ and $b = x_1 x_2 \dots x_n$ be the unique path from the root r to x in ρ. Consider the finite sequence

$$\xi = (v_1, s_1)(v_1, \tau_1)(v_2, s_2)\dots(v_n, s_n) \in (C_A C_P)^{n-1} C_A$$

where $v_i = \nu(x_i)$, $s_i = \mu(x_i)$, $\tau_j = \mathsf{tr}_\rho(x_j)$ for all $i \in [n]$ and $j \in [n-1]$. The sequence ξ is a partial play of $G(\mathcal{M}, M, v)$. We define $f(\xi) = (v_n, \mathsf{tr}_\rho(x_n))$. The partial function f becomes a total function and, therefore, a strategy for the player Automaton by mapping every value, for which we did not define f, to a fixed game position from C_P.

We show that every f-play develops along a branch of ρ. Every play of $G(\mathcal{M}, M, v)$ starts in the initial position $(\iota, v) = (\nu(r), \mu(r))$. For the induction step, let

$$\xi = (v_1, s_1)(v_1, \tau_1)(v_2, s_2) \dots (v_n, s_n) \in (C_A C_P)^{n-1} C_A$$

be a partial f-play and $b = x_1 \dots x_n \in C^n$ be the prefix of the branch of ρ such that $\mu(x_i) = s_i$, $\nu(x_i) = v_i$ for all $i \in [n]$ and $\mathsf{tr}_\rho(x_j) = \tau_j$ for all $j \in [n-1]$. If $\mathcal{M}$ is stuck in x_n, then we have $E_\rho(x_n) = \emptyset$ and the player Automaton cannot proceed in ξ. Otherwise, we have $E_\rho(x_n) \neq \emptyset$ and $f(\xi)$ is defined. After Automaton's f-conform move we are at game position $f(\xi) = (v_n, \mathsf{tr}_\rho(x_n)) \in C_P$. For every move $(D, s) \in \mathsf{tr}_\rho(x_n)$ of Pathfinder, there exists a configuration $x \in E_\rho(x_n)$ with $\mu(x) = s$ and $\eta_M(\nu(x_n), \nu(x)) = D$. Hence, every f-play ξ develops along a branch b of ρ. Since b is accepting and b and ξ are labelled by the same sequence over S, ξ is a play won by Automaton. Therefore, f is a winning strategy of Automaton in the game $G(\mathcal{M}, M, v)$. □

Lemma 3.12. *If the player Automaton has a winning strategy in the game $G(\mathcal{M}, M, v)$, then the pointed MSC (M, v) is accepted by $\mathcal{M}$.*

Proof. Let us assume that there exists a winning strategy f for Automaton in $G(\mathcal{M}, M, v)$. By Prop. 3.10, we can assume that f is memoryless. We inductively construct an accepting run ρ of $\mathcal{M}$ on (M, v). Firstly, we set $\rho_1 = (C_1, E_1, r, \mu_1, \nu_1)$ where $C_1 = \{r\}$, $E_1 = \emptyset$, $\mu_1(r) = \iota$, and $\nu_1(r) = v$. Now, let us assume that the S-labelled tree $\rho_i = (C_i, E_i, r, \mu_i, \nu_i)$ is already defined. Let $\{x_1, x_2, \dots, x_n\}$ be the set of all leaves of ρ_i in which $\mathcal{M}$ is not stuck, i.e., for all $j \in [n]$, the local MSCA $\mathcal{M}$ is not stuck in state $\mu_i(x_j)$ at position $\nu_i(x_j)$. For every $j \in [n]$, let τ_j be the transition such that $f(\nu_i(x_j), \mu_i(x_j)) = (\nu_i(x_j), \tau_j)$. We set $\rho_{i+1} = (C_{i+1}, E_{i+1}, r, \mu_{i+1}, \nu_{i+1})$ to the smallest (with respect to the size of the set of configurations C_{i+1}) S-labelled tree such that $\rho_{i+1} \restriction C_i = \rho_i$ and, for all $j \in [n]$, $\mathsf{tr}_{\rho_{i+1}}(x_j) = \tau_j$.

Let $\rho = (C, E, r, \mu, \nu) = \bigcup_{i \geq 1} \rho_i$. It can be easily checked that ρ is a run of $\mathcal{M}$ on (M, v). Now, let $b = x_1 x_2 x_3 \dots \in C^\infty$ be a branch of ρ. Consider the play

$$\xi = \big(\nu(x_1), \mu(x_1)\big)\big(\nu(x_1), \mathsf{tr}_\rho(x_1)\big)\big(\nu(x_2), \mu(x_2)\big)\big(\nu(x_2), \mathsf{tr}_\rho(x_2)\big) \dots$$

of $G(\mathcal{M}, M, v)$. It follows from the construction of ρ that ξ is an f-play. Since f is a winning strategy for player Automaton, ξ is won by Automaton. Since ξ and b share the same label, the branch b is accepting in $\mathcal{M}$. Hence, ρ is an accepting run of the local MSCA $\mathcal{M}$. □

The next two lemmas state that a player has a winning strategy in the current game if and only if there exists a winning strategy for its opponent in the dual game.

Lemma 3.13. *If Automaton has a winning strategy in $G(\mathcal{M}, M, v)$, then Pathfinder has a winning strategy in the game $G(\mathcal{M}^\#, M, v)$.*

Proof. Let f_A be a winning strategy of Automaton in the game $G(\mathcal{M}, M, v)$. We show that there exists a strategy f_P for Pathfinder such that, for every f_P-play $\xi^\#$ in $G(\mathcal{M}^\#, M, v)$, there exists an f_A-play ξ in $G(\mathcal{M}, M, v)$ such that $\xi \restriction C_A = \xi^\# \restriction C_A$. Note that the initial positions of the games $G(\mathcal{M}, M, v)$ and $G(\mathcal{M}^\#, M, v)$ are the same. For the induction step, let $n \geq 0$, $\xi^\# \in (C_A C_P^\#)^n \{(w, s)(w, \tau^\#)\}$ be a partial play of $G(\mathcal{M}^\#, M, v)$, and $\xi \in (C_A C_P)^n \{(w, s)(w, \tau)\}$ be a partial f_A-play of $G(\mathcal{M}, M, v)$ such that $\xi^\# \restriction C_A = \xi \restriction C_A$. From Lemma 3.2 it follows that there exists a movement $(D, s') \in \tau \cap \tau^\#$. Pathfinder chooses (D, s') as his next move resulting in a game position (w', s') where $\eta^M(w, w') = D$, i.e.,

$f_P(\xi^\#) = (w', s')$. Clearly, the sequences $\xi(w', s')$ and $\xi^\#(w', s')$ are equal when restricting them to positions from C_A.

Thus, for every f_P-play $\xi^\#$ in $G(\mathcal{M}^\#, M, v)$, there exists an f_A-play ξ in $G(\mathcal{M}, M, v)$ such that $\mu(\xi^\#) = \mu(\xi)$. Since f_A is a winning strategy, ξ is a play won by Automaton in $G(\mathcal{M}, M, v)$. From Remark 3.5 it follows that the play $\xi^\#$ in $G(\mathcal{M}^\#, M, v)$ is won by Pathfinder. Hence, we showed that Pathfinder has a winning strategy in the game $G(\mathcal{M}^\#, M, v)$. □

Lemma 3.14. *If Pathfinder has a winning strategy in $G(\mathcal{M}^\#, M, v)$, then Automaton has a winning strategy in the game $G(\mathcal{M}, M, v)$.*

Proof. Let f_P be a winning strategy of Pathfinder in the game $G(\mathcal{M}^\#, M, v)$. We show that there exists a winning strategy f_A of Automaton ensuring that, for every f_A-play ξ in the game $G(\mathcal{M}, M, v)$, there exists an f_P-play $\xi^\#$ in $G(\mathcal{M}^\#, M, v)$ such that $\xi^\# \restriction C_A = \xi \restriction C_A$. The initial positions of the games $G(\mathcal{M}, M, v)$ and $G(\mathcal{M}^\#, M, v)$ are the same. For the induction step, let $\xi \in (C_A C_P)^n\{(w, s)\}$ be a partial play of $G(\mathcal{M}, M, v)$, and $\xi^\# \in (C_A C_P^\#)^n\{(w, s)\}$ be a partial f_P-play in $G(\mathcal{M}^\#, M, v)$ such that $\xi \restriction C_A = \xi^\# \restriction C_A$. We define

$$X = \{(D, s') \in S \times \mathbb{M} \mid \text{there exists } \tau^\# \in [\![\delta^\#(s, \lambda^M(w))]\!] \text{ and } w' \in V^M \\ \text{such that } f_P(w, \tau^\#) = (w', s') \text{ and } \eta^M(w, w') = D\}$$

to be the set of the possible f_P-conform moves of Pathfinder after Automaton's next move in the play $\xi^\#$. We claim that there exists a transition $\tau \in [\![\delta(s, \lambda^M(w))]\!]$ with $\tau \subseteq X$.

Towards a contradiction, suppose there is no such τ. Then, for all $\tau' \in [\![\delta(s, \lambda^M(w))]\!]$, there exists a movement $(D_{\tau'}, s_{\tau'}) \in \tau'$ with $(D_{\tau'}, s_{\tau'}) \notin X$. If $Z = \{(D_{\tau'}, s_{\tau'}) \mid \tau' \in [\![\delta(s, \lambda^M(w))]\!]\}$, then $Z \cap Y \neq \emptyset$ for all $Y \in \mathsf{mod}(\delta(s, \lambda^M(w)))$. From Lemma 3.3 it follows that $Z \in \mathsf{mod}(\delta^\#(s, \lambda^M(w)))$. Hence, there exists a transition $\tau'' \in [\![\delta^\#(s, \lambda^M(w))]\!]$ with $\tau'' \subseteq Z$. However, we have $\tau'' \cap X = \emptyset$ which is a contradiction to our definition of X.

Automaton chooses the above transition τ with $\tau \subseteq X$ as his next move resulting in a game position (w, τ) in the game $G(\mathcal{M}, M, v)$, i.e., $f_A(\xi) = (w, \tau)$. For every move (D, s') of Pathfinder in $G(\mathcal{M}, M, v)$, there exists a $\tau^\# \in [\![\delta^\#(w, s)]\!]$ with $f_P(\xi^\#(w, \tau^\#)) = (w', s')$ and $\eta_M(w, w') = D$. This follows from the fact that $(D, s') \in X$. Clearly, the sequences $\xi(w, \tau)(w', s')$ and $\xi^\#(w, \tau^\#)(w', s')$ are equal when restricting them to positions from C_A.

Let ξ be an f_A-play in $G(\mathcal{M}, M, v)$. There exists an f_P-play $\xi^\#$ in $G(\mathcal{M}^\#, M, v)$ with $\xi \restriction C_A = \xi^\# \restriction C_A$. Since $\mu(\xi) = \mu(\xi^\#)$ and since $\xi^\#$ is a play won by Pathfinder in $G(\mathcal{M}^\#, M, v)$, the play ξ in $G(\mathcal{M}, M, v)$ must be won by Automaton (by Remark 3.5). Thus, f_A is a winning strategy for Automaton in $G(\mathcal{M}, M, v)$. □

We are now able to prove our main theorem from this section.

Proof of Theorem 3.6. By Lemma 3.11 and Lemma 3.12, the pointed MSC (M, v) is accepted by $\mathcal{M}$ if and only if Automaton has a winning strategy in $G(\mathcal{M}, M, v)$. By the lemmas 3.13 and 3.14, the latter is the case if and only if Pathfinder has a winning strategy in the game $G(\mathcal{M}^\#, M, v)$. From Prop. 3.10 it follows that this is the case if and only if Automaton has no winning strategy in $G(\mathcal{M}^\#, M, v)$ respectively $\mathcal{M}^\#$ does not accept (M, v) (again by the Lemmas 3.11 and 3.12). □

Figure 4: Illustrations of the local MSCAs $\mathcal{M}_\sigma$ (left side) and $\mathcal{M}_{\langle \mathsf{proc} \rangle \mathsf{tt}}$ (right side).

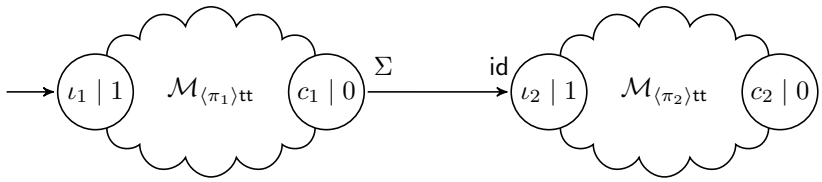

Figure 5: Illustration of $\mathcal{M}_{\langle \pi_1 ; \pi_2 \rangle \mathsf{tt}}$.

4. Translation of Local CRPDL Formulas

In this section, we show that, for every local CRPDL formula α, one can compute a local MSCA $\mathcal{M}_\alpha$ in polynomial time which exactly accepts the set of models of α. More formally:

Theorem 4.1. *From a local formula α, one can construct in time $\mathsf{poly}(|\alpha|)$ a local MSCA $\mathcal{M}_\alpha$ such that, for all pointed MSCs (M, v), we have $M, v \models \alpha$ if and only if $(M, v) \in L(\mathcal{M}_\alpha)$. The size of $\mathcal{M}_\alpha$ is linear in the size of α.*

The rest of this section prepares the proof of the above theorem. The actual proof can be found on page 23.

4.1. Construction.

If α is a local formula, then we distinguish the following cases:

Case $\alpha = \sigma$. We define $\mathcal{M}_\sigma = (\{\iota, c\}, \delta, \iota, c, \kappa)$ where $\kappa(\iota) = 1$, $\kappa(c) = 0$, and

$$\delta(s, \sigma') = \begin{cases} (\mathsf{id}, c) & \text{if } \sigma = \sigma' \text{ and } s = \iota \\ \bot & \text{otherwise} \end{cases}$$

for all $s \in \{\iota, c\}$ and $\sigma' \in \Sigma$. The local MSCA $\mathcal{M}_\sigma$ is depicted on the left side of Fig. 4.

Case $\alpha = \neg\beta$. We define $\mathcal{M}_{\neg\beta}$ to be the dual automaton of $\mathcal{M}_\beta$ (cf. Definition 3.4).

Case $\alpha = \langle D \rangle \, \mathsf{tt}$ with $D \in \mathbb{M}$. We define $\mathcal{M}_{\langle D \rangle \mathsf{tt}} = (\{\iota, c\}, \delta, \iota, c, \kappa)$ where $\kappa(\iota) = 1$, $\kappa(c) = 0$, and

$$\delta(s, \sigma) = \begin{cases} (D, c) & \text{if } s = \iota \\ \bot & \text{otherwise} \end{cases}$$

for all $s \in \{\iota, c\}$ and $\sigma \in \Sigma$. On the right side of Fig. 4, there is an illustration of $\mathcal{M}_{\langle \mathsf{proc} \rangle \mathsf{tt}}$.

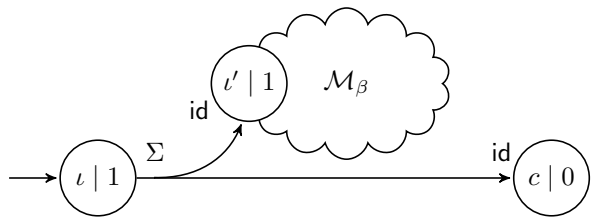

Figure 6: Illustration of the local MSCA $\mathcal{M}_{\langle\{\beta\}\rangle\mathsf{tt}}$.

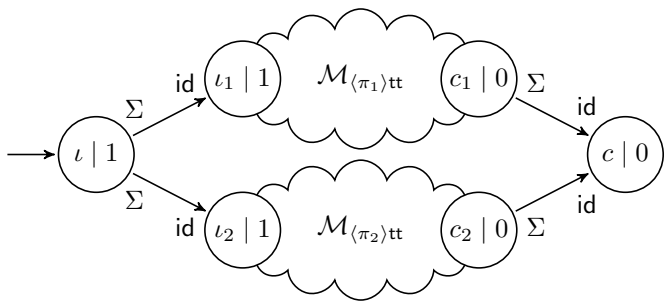

Figure 7: Illustration of $\mathcal{M}_{\langle\pi_1+\pi_2\rangle\mathsf{tt}}$.

Case $\alpha = \langle\pi_1;\pi_2\rangle\,\mathsf{tt}$. If $\mathcal{M}_{\langle\pi_i\rangle\mathsf{tt}} = (S_i, \delta_i, \iota_i, c_i, \kappa_i)$ for $i \in [2]$ and $\delta_1(c_1, \sigma) = \bot$ for all $\sigma \in \Sigma$, then we define $\mathcal{M}_{\langle\pi_1;\pi_2\rangle\mathsf{tt}}$ to be the local MSCA $(S, \delta, \iota_1, c_2, \kappa)$ where $S = S_1 \uplus S_2$, $\kappa = \kappa_1 \cup \kappa_2$, and

$$\delta(s,\sigma) = \begin{cases} (\mathsf{id}, \iota_2) & \text{if } s = c_1 \\ \delta_1(s,\sigma) & \text{if } s \in S_1 \setminus \{c_1\} \\ \delta_2(s,\sigma) & \text{if } s \in S_2 \end{cases}$$

for all $s \in S$ and $\sigma \in \Sigma$. Figure 5 shows an illustration of $\mathcal{M}_{\langle\pi_1;\pi_2\rangle\mathsf{tt}}$.

The automaton $\mathcal{M}_{\langle\pi_1;\pi_2\rangle\mathsf{tt}}$ is the concatenation of the local MSCAs $\mathcal{M}_{\langle\pi_1\rangle\mathsf{tt}}$ and $\mathcal{M}_{\langle\pi_2\rangle\mathsf{tt}}$. Intuitively, $\mathcal{M}_{\langle\pi_1;\pi_2\rangle\mathsf{tt}}$ starts a copy of $\mathcal{M}_{\langle\pi_1\rangle\mathsf{tt}}$ and, when this copy changes into its concatenation state c_1, the automaton $\mathcal{M}_{\langle\pi_1;\pi_2\rangle\mathsf{tt}}$ proceeds with starting a copy of the local MSCA $\mathcal{M}_{\langle\pi_2\rangle\mathsf{tt}}$. Note that $\mathcal{M}_{\langle\pi_1;\pi_2\rangle\mathsf{tt}}$ is forced to start the copy of $\mathcal{M}_{\langle\pi_2\rangle\mathsf{tt}}$ since runs of local MSCAs are maximal by definition (see Definition 2.21) and we have $\{(\mathsf{id}, \iota_2)\} \in [\![\delta(c_1, \sigma)]\!]$ for every $\sigma \in \Sigma$.

Case $\alpha = \langle\{\beta\}\rangle\,\mathsf{tt}$. If $\mathcal{M}_\beta = (S', \delta', \iota', c', \kappa')$, then we define $\mathcal{M}_{\langle\{\beta\}\rangle\mathsf{tt}} = (S, \delta, \iota, c, \kappa)$ where $S = S' \uplus \{\iota, c\}$, $\kappa = \kappa' \cup \{(\iota, 1), (c, 0)\}$, and

$$\delta(s,\sigma) = \begin{cases} (\mathsf{id}, \iota') \wedge (\mathsf{id}, c) & \text{if } s = \iota \\ \bot & \text{if } s = c \\ \delta'(s,\sigma) & \text{if } s \in S' \end{cases}$$

for all $s \in S$ and $\sigma \in \Sigma$. The automaton $\mathcal{M}_{\langle\{\beta\}\rangle\mathsf{tt}}$ is depicted in Figure 6.

Intuitively, the local MSCA $\mathcal{M}_{\langle\{\beta\}\rangle\mathsf{tt}}$ starts $\mathcal{M}_\beta$ to test whether $M, v \models \beta$ holds and, at the same time, changes into its concatenation state.

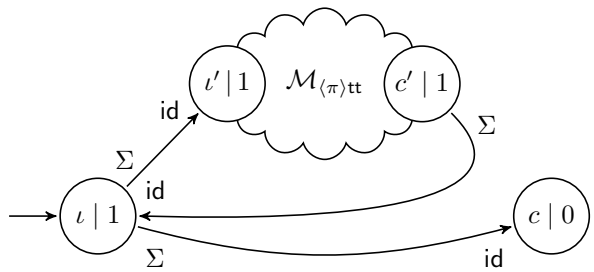

Figure 8: Illustration of the local MSCA $\mathcal{M}_{\langle\pi^*\rangle\mathsf{tt}}$.

Case $\alpha = \langle\pi_1 + \pi_2\rangle\,\mathsf{tt}$. If $\mathcal{M}_{\langle\pi_i\rangle\mathsf{tt}} = (S_i, \delta_i, \iota_i, c_i, \kappa_i)$ and $\delta_i(c_i, \sigma) = \bot$ for all $i \in [2]$ and $\sigma \in \Sigma$, then we define $\mathcal{M}_{\langle\pi_1+\pi_2\rangle\mathsf{tt}}$ to be the local MSCA $(S, \delta, \iota, c, \kappa)$ where $S = S_1 \uplus S_2 \uplus \{\iota, c\}$, $\kappa = \kappa_1 \cup \kappa_2 \cup \{(\iota, 1), (c, 0)\}$, and

$$\delta(s,\sigma) = \begin{cases} (\mathsf{id}, \iota_1) \vee (\mathsf{id}, \iota_2) & \text{if } s = \iota \\ (\mathsf{id}, c) & \text{if } s = c_i \text{ and } i \in [2] \\ \delta_i(s,\sigma) & \text{if } s \in S_i \setminus \{c_i\} \text{ and } i \in [2] \\ \bot & \text{if } s = c \end{cases}$$

for all $s \in S$ and $\sigma \in \Sigma$. The local MSCA $\mathcal{M}_{\langle\pi_1+\pi_2\rangle\mathsf{tt}}$ is visualized in Fig. 7.

Case $\alpha = \langle\pi^*\rangle\,\mathsf{tt}$. If $\mathcal{M}_{\langle\pi\rangle\mathsf{tt}} = (S', \delta', \iota', c', \kappa')$ and $\delta'(c', \sigma) = \bot$ for all $\sigma \in \Sigma$, then we set $\mathcal{M}_{\langle\pi^*\rangle\mathsf{tt}} = (S, \delta, \iota, c, \kappa)$ where $S = S' \uplus \{\iota, c\}$, κ and κ' coincide on $S' \setminus \{c'\}$, $\kappa'(s) = 1$ if $s \in \{\iota, c'\}$, $\kappa'(c) = 0$, and

$$\delta(s,\sigma) = \begin{cases} (\mathsf{id}, \iota) & \text{if } s = c' \\ (\mathsf{id}, \iota') \vee (\mathsf{id}, c) & \text{if } s = \iota \\ \bot & \text{if } s = c \\ \delta'(s,\sigma) & \text{if } s \in S' \setminus \{c'\} \end{cases}$$

for all $s \in S$ and $\sigma \in \Sigma$. See Fig. 8 for a visualization of $\mathcal{M}_{\langle\pi^*\rangle\mathsf{tt}}$.

Intuitively, the local MSCA $\mathcal{M}_{\langle\pi^*\rangle\mathsf{tt}}$ executes a copy of the automaton $\mathcal{M}_{\langle\pi\rangle\mathsf{tt}}$ and, every time this copy changes into its concatenation state c', the local MSCA $\mathcal{M}_{\langle\pi^*\rangle\mathsf{tt}}$ nondeterministically decides whether it restarts this copy again or changes into the concatenation state c.

Case $\alpha = \langle\pi\rangle^\omega$. If $\mathcal{M}_{\langle\pi\rangle\mathsf{tt}} = (S, \delta', \iota, c, \kappa)$ and $\delta'(c, \sigma) = \bot$ for all $\sigma \in \Sigma$, then we set $\mathcal{M}_{\langle\pi\rangle^\omega} = (S, \delta, \iota, c, \kappa)$ where

$$\delta(s,\sigma) = \begin{cases} (\mathsf{id}, \iota) & \text{if } s = c \\ \delta'(s,\sigma) & \text{if } s \in S \setminus \{c\} \end{cases}$$

for all $s \in S$ and $\sigma \in \Sigma$.

4.2. **Concatenation States.** In this section, we prove a technical proposition stating that, for all path formulas α, every accepting run of the local MSCA $\mathcal{M}_\alpha$ exhibits exactly one configuration labelled by the concatenation state. It will be of use in Sect. 4.3 to show the correctness of our construction.

Firstly, we introduce the notion of main states. A state s is called a main state if the concatenation state can be reached from s. The intuition of this type of states is the following: If π is a path expression and ρ is an accepting run of $\mathcal{M}_{\langle\pi\rangle\mathsf{tt}}$, then ρ exhibits one main branch b by which $\mathcal{M}$ "processes" the path expression π. The label of b solely consists of the not yet formally defined main states. In all the other branches of ρ, i.e., in the branches which fork from b, $\mathcal{M}$ basically executes tests of the form $\{\alpha\}$. All these branches are labelled by non-main states.

Definition 4.2. Let $\mathcal{M} = (S, \delta, \iota, c, \kappa)$ be a local MSCA and $s \in S$. We inductively define the set of *main states* $\mathsf{ms}(\mathcal{M})$ of $\mathcal{M}$: $\mathsf{ms}(\mathcal{M})$ is the least set such that, for all $s \in S$, we have $s \in \mathsf{ms}(\mathcal{M})$ if and only if

(1) $s = c$ or
(2) there exist $s' \in \mathsf{ms}(\mathcal{M})$, $\sigma \in \Sigma$, $D \in \mathbb{M}$, and $\tau \in [\![\delta(s, \sigma)]\!]$ such that $(D, s') \in \tau$.

By examining our construction, one can make the following two simple observations.

Remark 4.3. If α is a path formula and $\mathcal{M}_\alpha = (S, \delta, \iota, c, \kappa)$, the following conditions hold:

(1) we have $\delta(c, \sigma) = \bot$ for every $\sigma \in \Sigma$
(2) for all $s \in \mathsf{ms}(\mathcal{M}_\alpha)$, we have

$$\kappa(s) = \begin{cases} 0 & \text{if } s = c \\ 1 & \text{otherwise} \end{cases}$$

If α is a path formula of the form $\langle\pi_1; \pi_2\rangle\,\mathsf{tt}$, then in our construction of $\mathcal{M}_\alpha$, we required $\delta_1(c_1, \sigma) = \bot$ for all $\sigma \in \Sigma$ where δ_1 is the transition relation and c_1 is the concatenation state of $\mathcal{M}_{\langle\pi_1\rangle\mathsf{tt}}$. It follows from the above observation (1) that our construction can be applied to all formulas of the form $\langle\pi_1; \pi_2\rangle\,\mathsf{tt}$. Similarly, this holds for our construction of $\mathcal{M}_\alpha$ in the cases $\alpha = \langle\pi_1 + \pi_2\rangle\,\mathsf{tt}$, $\alpha = \langle\pi^*\rangle\,\mathsf{tt}$, and $\alpha = \langle\pi\rangle^\omega$.

Lemma 4.4. *If α is a path formula and $\mathcal{M}_\alpha = (S, \delta, \iota, c, \kappa)$, then the following two conditions hold:*

(a) $\iota \in \mathsf{ms}(\mathcal{M}_\alpha)$
(b) *for all $s \in \mathsf{ms}(\mathcal{M}_\alpha)$, $\sigma \in \Sigma$, and $\tau \in [\![\delta(s, \sigma)]\!]$, we have*

$$|\tau \cap (\mathsf{ms}(\mathcal{M}_\alpha) \times \mathbb{M})| = 1 \tag{4.1}$$

Intuitively, the above lemma states that every run of M_α exhibits exactly one branch labelled solely by main states and that all other configurations of this run which are not part of this path are labelled by non-main states.

Proof. By simple inspection, our claim follows for the cases $\alpha = \langle D\rangle\,\mathsf{tt}$ with $D \in \mathbb{M}$ and $\alpha = \langle\{\beta\rangle\}\mathsf{tt}$. As our induction hypothesis, let us assume that the above lemma holds for $\mathcal{M}_{\langle\pi_i\rangle\mathsf{tt}} = (S_i, \delta_i, \iota_i, c_i, \kappa_i)$ where $i \in [2]$. If $\alpha = \langle\pi_1; \pi_2\rangle\,\mathsf{tt}$, then it can be easily checked that $\mathsf{ms}(\mathcal{M}_\alpha) = \mathsf{ms}(\mathcal{M}_{\langle\pi_1\rangle\mathsf{tt}}) \cup \mathsf{ms}(\mathcal{M}_{\langle\pi_2\rangle\mathsf{tt}})$. Hence, $\iota_1 \in \mathsf{ms}(\mathcal{M}_\alpha)$ and, therefore, property (a) is fulfilled. Now, let $\tau \in [\![\delta(s, \sigma)]\!]$ for some $s \in S$ and $\sigma \in \Sigma$. Then $\tau = \{(\mathsf{id}, \iota_2)\}$ (if $s = c_1$), $\tau \in [\![\delta_1(s, \sigma)]\!]$, or $\tau \in [\![\delta_2(s, \sigma)]\!]$. Together with our induction hypothesis it follows

that (4.1) holds for τ. Now, let us consider the case $\alpha = \langle \pi_1 + \pi_2 \rangle\, \mathsf{tt}$. By easy inspection it follows that

$$\mathsf{ms}(\mathcal{M}_\alpha) = \{\iota, c\} \cup \mathsf{ms}(\mathcal{M}_{\langle \pi_1 \rangle \mathsf{tt}}) \cup \mathsf{ms}(\mathcal{M}_{\langle \pi_2 \rangle \mathsf{tt}})\,.$$

Hence, property (a) follows. If $\tau \in [\![\delta(s,\sigma)]\!]$ for some $s \in S$ and $\sigma \in \Sigma$, then $\tau = \{(\mathsf{id}, \iota_1)\}$, $\tau = \{(\mathsf{id}, \iota_2)\}$, $\tau = \{(\mathsf{id}, c)\}$, $\tau \in [\![\delta_1(s,\sigma)]\!]$, or $\tau \in [\![\delta_2(s,\sigma)]\!]$. Property (b) follows from our induction hypothesis.

Finally, we need to deal with the case $\alpha = \langle \pi^* \rangle\, \mathsf{tt}$. For this, we assume that the above lemma holds for $\mathcal{M}_{\langle \pi \rangle \mathsf{tt}} = (S', \delta', \iota', c', \kappa')$. Again, it can be easily verified that $\mathsf{ms}(\mathcal{M}_\alpha) = \{\iota, c\} \cup \mathsf{ms}(\mathcal{M}_{\langle \pi \rangle \mathsf{tt}})$. Thus, property (a) holds. Now, let $\tau \in [\![\delta(s,\sigma)]\!]$ for some $s \in S$ and $\sigma \in \Sigma$. We have $\tau = \{(\mathsf{id}, \iota)\}$, $\tau = \{(\mathsf{id}, \iota')\}$, $\tau = \{(\mathsf{id}, c)\}$, or $\tau \in [\![\delta'(s,\sigma)]\!]$. Property (b) follows from our induction hypothesis. □

Proposition 4.5. *Let α be a path formula and $\rho = (C, E, r, \mu, \nu)$ be an accepting run of $\mathcal{M}_\alpha = (S, \delta, \iota, c, \kappa)$. There exists exactly one configuration from C denoted by $\mathsf{cs}(\rho)$ with $\mu(\mathsf{cs}(\rho)) = c$.*

Proof. It follows from Lemma 4.4 that all configurations $x \in C$ with $\mu(x) \in \mathsf{ms}(\mathcal{M}_\alpha)$ form a unique branch $b = x_1 x_2 x_3 \ldots \in C^\infty$ of ρ. Since ρ is accepting, b must be accepting. It follows from Remark 4.3 that $\mu(b) \in (\mathsf{ms}(\mathcal{M}_\alpha) \setminus \{c\})^* \{c\}$. Therefore, every accepting run of $\mathcal{M}_\alpha$ contains exactly one configuration labelled by c. □

4.3. **Correctness.** Let α be a local formula. We show by induction over the construction of α that $L(\mathcal{M}_\alpha) = L(\alpha)$. The following claim is used as the induction hypothesis of our proof. Recall that $\mathsf{reach}_M(v, \pi)$ is the set of all events which can be reached from v by a path described by π in the MSC M.

Claim 4.6. *Let α be a path formula. For all MSCs M, events $v, v' \in V^M$, we have $v' \in \mathsf{reach}_M(v, \alpha)$ if and only if there exists an accepting run ρ of $\mathcal{M}_\alpha$ on (M, v) with $\nu(\mathsf{cs}(\rho)) = v'$.*

The following four technical lemmas deal with the correctness of the constructions of the local MSCAs $\mathcal{M}_{\langle \pi_1 ; \pi_2 \rangle \mathsf{tt}}$ and $\mathcal{M}_{\langle \pi^* \rangle \mathsf{tt}}$.

Lemma 4.7. *Let M be an MSC, $v_1, v' \in V^M$, and π_1, π_2 be path expressions. If Claim 4.6 holds for $\langle \pi_1 \rangle\, \mathsf{tt}$ and $\langle \pi_2 \rangle\, \mathsf{tt}$ and we have $v' \in \mathsf{reach}_M(v_1, \pi_1 ; \pi_2)$, then there exists an accepting run $\rho = (C, E, r, \mu, \nu)$ of $\mathcal{M}_{\langle \pi_1 ; \pi_2 \rangle \mathsf{tt}}$ on (M, v_1) with $\nu(\mathsf{cs}(\rho)) = v'$.*

Proof. Let $M_{\langle \pi_1 ; \pi_2 \rangle \mathsf{tt}} = (S, \delta, \iota, c, \kappa)$ and $M_{\langle \pi_i \rangle \mathsf{tt}} = (S_i, \delta_i, \iota_i, c_i, \kappa_i)$ for all $i \in [2]$. If we have $v' \in \mathsf{reach}_M(v_1, \pi_1 ; \pi_2)$, then, by definition, there exists an event $v_2 \in V^M$ such that $v_2 \in \mathsf{reach}_M(v_1, \pi_1)$ and $v' \in \mathsf{reach}_M(v_2, \pi_2)$. It follows from our assumption that there exists an accepting run $\rho_1 = (C_1, E_1, r_1, \mu_1, \nu_1)$ of the local MSCA $\mathcal{M}_{\langle \pi_1 \rangle \mathsf{tt}}$ on (M, v_1) with $\nu_1(\mathsf{cs}(\rho_1)) = v_2$ and that there exists an accepting run $\rho_2 = (C_2, E_2, r_2, \mu_2, \nu_2)$ of $\mathcal{M}_{\langle \pi_2 \rangle \mathsf{tt}}$ on (M, v_2) with $\nu_2(\mathsf{cs}(\rho_2)) = v'$. Consider the S-labelled tree $\rho = (C_1 \uplus C_2, E, r_1, \mu, \nu)$ where $E = E_1 \cup E_2 \cup \{(\mathsf{cs}(\rho_1), r_2)\}$, $\mu = \mu_1 \cup \mu_2$, and $\nu = \nu_1 \cup \nu_2$. It can be easily checked that ρ is a run of the local MSCA $\mathcal{M}_{\langle \pi_1 ; \pi_2 \rangle \mathsf{tt}}$ on (M, v_1) with $\mathsf{cs}(\rho) = \mathsf{cs}(\rho_2)$. In Fig. 9, the run ρ is depicted where $\mathsf{cs}(\rho_i)$ is denoted by x_i for $i \in [2]$.

It remains to show that ρ is accepting. Let b be a branch of ρ. We distinguish two cases: If $b \in C_1^\infty$, then b is also a branch from ρ_1. Since ρ_1 is accepting, the branch b is accepting in $\mathcal{M}_{\langle \pi_1 \rangle \mathsf{tt}}$. Since $\kappa_1 \subseteq \kappa$ and $\mu_1 \subseteq \mu$, b is accepting in $\mathcal{M}_{\langle \pi_1 ; \pi_2 \rangle \mathsf{tt}}$, too. Otherwise (i.e., if

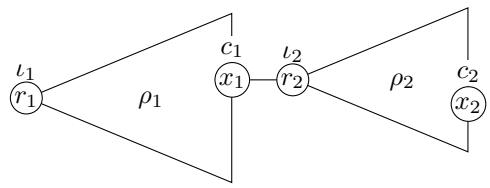

Figure 9: The run ρ of $\mathcal{M}_{\langle \pi_1;\pi_2\rangle \mathsf{tt}}$.

$b \in C_1^+\{r_2\}C_2^\infty$), there exists a suffix of b which is an accepting branch in ρ_2. Because of this fact, $\mu_2 \subseteq \mu$, and $\kappa_2 \subseteq \kappa$, the branch b is also accepting in $\mathcal{M}_{\langle \pi_1;\pi_2\rangle \mathsf{tt}}$. Hence, ρ is an accepting run of the automaton $\mathcal{M}_{\langle \pi_1;\pi_2\rangle \mathsf{tt}}$ on (M, v_1) with $\nu(\mathsf{cs}(\rho)) = \nu(\mathsf{cs}(\rho_2)) = v'$. □

Lemma 4.8. *Let M be an MSC, $v, v' \in V^M$, and π_1, π_2 be path expressions. If Claim 4.6 holds for $\langle \pi_1\rangle\, \mathsf{tt}$ and $\langle \pi_2\rangle\, \mathsf{tt}$ and there exists an accepting run $\rho = (C, E, r_1, \mu, \nu)$ of $\mathcal{M}_{\langle \pi_1;\pi_2\rangle \mathsf{tt}}$ on (M, v) with $\nu(\mathsf{cs}(\rho)) = v'$, then $v' \in \mathsf{reach}_M(v, \pi_1; \pi_2)$.*

Proof. Let $M_{\langle \pi_1;\pi_2\rangle \mathsf{tt}} = (S, \delta, \iota, c, \kappa)$ and $M_{\langle \pi_i\rangle \mathsf{tt}} = (S_i, \delta_i, \iota_i, c_i, \kappa_i)$ for all $i \in [2]$. Since $\mu(\mathsf{cs}(\rho)) = c$ and $c = c_2 \in S_2$, there has to exist a configuration $r_2 \in C$ with $\mu(r_2) = \iota_2$. Towards a contradiction, let us assume that there exists another configuration $r_3 \in C$ with $\mu(r_3) = \iota_2$. Because ι_2 is a main state in $\mathcal{M}_{\langle \pi_1;\pi_2\rangle \mathsf{tt}}$ (see the proof of Lemma 4.4) and due to Lemma 4.4, r_2 and r_3 must occur in a branch of ρ. This is a contradiction to $\iota_2 \notin \mathsf{src}_{\mathcal{M}_{\langle \pi_1;\pi_2\rangle \mathsf{tt}}}(\iota_2)$, i.e., ι_2 is not reachable from ι_2 in $\mathcal{M}_{\langle \pi_1;\pi_2\rangle \mathsf{tt}}$. The latter fact follows by simple inspection of the transition relation of $\mathcal{M}_{\langle \pi_1;\pi_2\rangle \mathsf{tt}}$. Let $C_2 = \{y \in C \mid (r_2, y) \in E^*\}$ and $C_1 = C \setminus C_2$. It can be easily checked that the S-labelled tree $\rho_i = \rho \restriction C_i$ is a run of $\mathcal{M}_{\langle \pi_i\rangle \mathsf{tt}}$ for all $i \in [2]$. From the definition of the transition function δ, it follows that there exists a configuration $x_1 \in C_1$ with $\mu(x_1) = c_1$, $E_\rho(x_1) = \{r_2\}$, $\nu(x_1) = \nu(r_2)$. Figure 9 shows a depiction of the run ρ consisting of ρ_1 and ρ_2 where $\mathsf{cs}(\rho) = x_2$.

If b is a branch of ρ_2 and b' is the unique path in ρ from r_1 to x_1, then $b'b$ is a branch of ρ. Since ρ is accepting, $\mu_2 \subseteq \mu$, and $\kappa_2 \subseteq \kappa$, b is accepting in $\mathcal{M}_{\langle \pi_2\rangle \mathsf{tt}}$. Hence, ρ_2 is an accepting run of $\mathcal{M}_{\langle \pi_2\rangle \mathsf{tt}}$ on $(M, \nu(x_1))$ with $\mathsf{cs}(\rho_2) = \mathsf{cs}(\rho)$. Now, let b be a branch of ρ_1. If $b \in (C \setminus \{x_1\})^\infty$, then b is a branch of ρ with $\mu(b) \in S_1^\infty$. Since ρ is accepting, $\mu_1 \subseteq \mu$ and $\kappa_1 \subseteq \kappa$, b is accepting in $\mathcal{M}_{\langle \pi_1\rangle \mathsf{tt}}$. Otherwise (i.e., if $b \in C^*\{x_1\}$), b ends in a configuration labelled by c_1. By Remark 4.3, $\kappa_1(c_1)$ is even and, therefore, b is accepting in ρ_1. Hence, ρ_1 is an accepting run of $\mathcal{M}_{\langle \pi_1\rangle \mathsf{tt}}$ on (M, v) with $\mathsf{cs}(\rho_1) = x_1$. By our assumption, it follows that $\nu(x_1) \in \mathsf{reach}_M(v, \pi_1)$ and $v' \in \mathsf{reach}_M(\nu(x_1), \pi_2)$. Therefore, we have $v' \in \mathsf{reach}_M(v, \pi_1; \pi_2)$. □

Lemma 4.9. *Let M be an MSC, $v, v' \in V^M$, and π be a path expression. If Claim 4.6 holds for $\langle \pi\rangle\, \mathsf{tt}$ and we have $v' \in \mathsf{reach}_M(v, \pi^*)$, then there exists an accepting run $\rho = (C, E, r, \mu, \nu)$ of $\mathcal{M}_{\langle \pi^*\rangle \mathsf{tt}}$ on (M, v) with $\nu(\mathsf{cs}(\rho)) = v'$.*

Proof. Let $\mathcal{M}_{\langle \pi^*\rangle \mathsf{tt}} = (S, \delta, \iota, c, \kappa)$ and $\mathcal{M}_{\langle \pi\rangle \mathsf{tt}} = (S', \delta', \iota', c', \kappa')$. Since $v' \in \mathsf{reach}_M(v, \pi^*)$, there exist an $n \geq 0$ and events $v_1, v_2, \ldots, v_{n+1} \in V^M$ such that $v_1 = v$, $v_{n+1} = v'$, and $v_{i+1} \in \mathsf{reach}_M(v_i, \pi)$ for all $i \in [n]$. If $n = 0$, then the lemma follows by easy inspection of the construction of $\mathcal{M}_{\langle \pi^*\rangle \mathsf{tt}}$. Now, let us assume that $n \geq 1$. Since Claim 4.6 holds for $\langle \pi\rangle\, \mathsf{tt}$, we can assume that there exist, for all $i \in [n]$, accepting runs $\rho_i = (C_i, E_i, r_i, \mu_i, \nu_i)$ of the automaton $\mathcal{M}_{\langle \pi\rangle \mathsf{tt}}$ on the pointed MSC (M, v_i) with $\nu_i(\mathsf{cs}(\rho_i)) = v_{i+1}$. Without loss

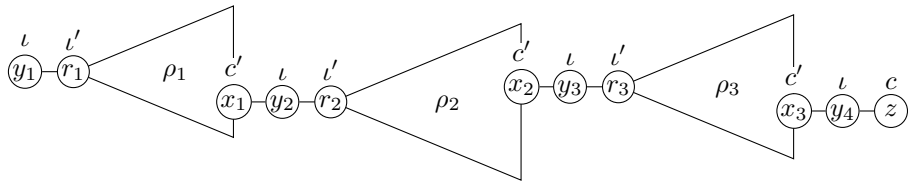

Figure 10: The run ρ of $\mathcal{M}_{\langle\pi^*\rangle\mathsf{tt}}$ consisting of three runs of $\mathcal{M}_{\langle\pi\rangle\mathsf{tt}}$ ($n = 3$).

of generality, we may assume that $C_i \cap C_j = \emptyset$ for all $i, j \in [n]$ (note that we can enforce $C_i \cap C_j = \emptyset$ by renaming the nodes of the C_i's). Let $x_i = \mathsf{cs}(\rho_i)$ for all $i \in [n]$. The S-labelled tree $\rho = (C, E, y_1, \mu, \nu)$ where

$$\begin{aligned}
C &= \textstyle\bigcup_{i\in[n]} C_i \uplus \{y_1, y_2, \ldots, y_{n+1}, z\}\,,\\
E &= \textstyle\bigcup_{i\in[n]} E_i \cup \{(y_i, r_i) \mid i \in [n]\} \cup \{(x_i, y_{i+1}) \mid i \in [n]\} \cup \{(y_{n+1}, z)\}\,,\\
\mu &= \textstyle\bigcup_{i\in[n]} \mu_i \cup \{(y_i, \iota) \mid i \in [n+1]\} \cup \{(z, c)\}\,,\\
\nu &= \textstyle\bigcup_{i\in[n]} \nu_i \cup \{(y_i, v_i) \mid i \in [n+1]\} \cup \{(z, v')\}
\end{aligned}$$

is a run of $\mathcal{M}_{\langle\pi^*\rangle\mathsf{tt}}$ on (M, v) with $\mathsf{cs}(\rho) = v'$ — this follows by an easy inspection of the construction of $\mathcal{M}_{\langle\pi^*\rangle\mathsf{tt}}$. Figure 10 shows a depiction of ρ for the case $n = 3$.

It remains to show that ρ is accepting. Let b be a branch of ρ. If $b \in (C \setminus \{z\})^\infty$, then there exist a suffix b' of b and an index $i \in [n]$ such that b' is a branch of ρ_i. Note that we have $\mu(b') \in (S' \setminus \{c'\})^\infty$. Since ρ_i is accepting, b' is accepting in $\mathcal{M}_{\langle\pi\rangle\mathsf{tt}}$. Since $\mu_i \subseteq \mu$ and $\kappa' \restriction (S' \setminus \{c'\}) \subseteq \kappa$, it follows that b is accepting in $\mathcal{M}_{\langle\pi^*\rangle\mathsf{tt}}$. Otherwise (i.e., if $b \in C^*\{z\}$), we have $\mu(b) = S^*\{c\}$. Since $\kappa(c)$ is even, b is accepting in $\mathcal{M}_{\langle\pi^*\rangle\mathsf{tt}}$. Hence, ρ is an accepting run of $\mathcal{M}_{\langle\pi^*\rangle\mathsf{tt}}$ on the pointed MSC (M, v) with $\nu(\mathsf{cs}(\rho)) = v'$. □

Lemma 4.10. *Let M be an MSC, $v, v' \in V^M$, and π be a path expression. If Claim 4.6 holds for $\langle\pi\rangle\,\mathsf{tt}$ and there exists an accepting run $\rho = (C, E, y_1, \mu, \nu)$ of $\mathcal{M}_{\langle\pi^*\rangle\mathsf{tt}}$ on (M, v) with $\nu(\mathsf{cs}(\rho)) = v'$, then $v' \in \mathsf{reach}_M(v, \pi^*)$.*

Proof. Let $\mathcal{M}_{\langle\pi^*\rangle\mathsf{tt}} = (S, \delta, \iota, c, \kappa)$ and $\mathcal{M}_{\langle\pi\rangle\mathsf{tt}} = (S', \delta', \iota', c', \kappa')$. Let R be the set of all configurations from C labelled by ι'. If $R = \emptyset$, then ρ consists of exactly one branch $b = y_1 z$ with $\mu(b) = \iota c$ and $\nu(y_1) = \nu(z)$. This can be easily verified by inspecting the transition function δ. From $\nu(y_1) = v$ and $\nu(z) = \nu(\mathsf{cs}(\rho)) = v'$, it follows that $v = v'$. Hence, $v' \in \mathsf{reach}_M(v, \pi^*)$.

Now, let us assume that $R \neq \emptyset$. It follows from $\iota' \in \mathsf{ms}(\mathcal{M}_{\langle\pi^*\rangle\mathsf{tt}})$ (see the last paragraph of the proof of Lemma 4.4) and Lemma 4.4 that all configurations from R occur in a unique finite branch $b = z_1 z_2 \ldots z_\ell$ of ρ. Without loss of generality, we can assume that $R = \{r_1, r_2, \ldots, r_n\}$ such that there exist $i_1 < i_2 < \ldots < i_n$ with $z_{i_k} = r_k$ for all $k \in [n]$. By examining the transition function δ, one can see that:

- For every $i \in [n]$, there exists a $y_i \in C$ with $E_\rho(y_i) = \{r_i\}$ and $\mu(y_i) = \iota$.
- There exists a configuration $y_{n+1} \in C$ with $E_\rho(y_{n+1}) = \{\mathsf{cs}(\rho)\}$ and $\mu(y_{n+1}) = \iota$.
- For every $i \in [n]$, there exists a configuration x_i with $E_\rho(x_i) = \{y_{i+1}\}$ and $\mu(x_i) = c'$.

Let $C_i = (\{x \in C \mid (r_i, x) \in E^*\} \setminus \{x \in C \mid (y_{i+1}, x) \in E^*\})$ for all $i \in [n]$. It can be easily verified that the S-labelled tree $\rho_i = (C_i, E_i, r_i, \mu_i, \nu_i) = \rho \restriction C_i$ is a run of $\mathcal{M}_{\langle\pi\rangle\mathsf{tt}}$ on

$(M, \nu(r_i))$ with $\mathsf{cs}(\rho_i) = x_i$. Figure 10 shows a depiction of the runs $\rho_1, \rho_2, \ldots, \rho_n$ forming the run ρ for the case $n = 3$.

We now show that ρ_i is an accepting run for every $i \in [n]$. Let $i \in [n]$ and b be a branch of ρ_i. If $b \in (C_i \setminus \{x_i\})$, then there exists a path b' from y_1 to y_i in ρ such that $b'b$ is a branch of ρ. Since ρ is accepting, $b'b$ is accepting in $\mathcal{M}_{\langle \pi^* \rangle \mathsf{tt}}$. Because of $\mu_i \subseteq \mu$, $\mu(b) \in (S' \setminus \{c'\})^\infty$ and $\kappa' \restriction (S' \setminus \{c'\}) \subseteq \kappa$, b is accepting in $\mathcal{M}_{\langle \pi \rangle \mathsf{tt}}$. If $b \in C^*\{x_i\}$, then $\mu(b) \in S'^*\{c'\}$. By Remark 4.3, $\kappa'(c')$ is even and, therefore, b is accepting in $\mathcal{M}_{\langle \pi \rangle \mathsf{tt}}$. Hence ρ_i is an accepting run of $\mathcal{M}_{\langle \pi \rangle \mathsf{tt}}$ on $(M, \nu(r_i))$ with $\mathsf{cs}(\rho_i) = x_i$.

Since Claim 4.6 holds for $\langle \pi \rangle\, \mathsf{tt}$, we can assume that $\nu(x_i) \in \mathsf{reach}_M(\nu(r_i), \pi)$. By checking the definition of the transition function δ, one can easily verify that $\nu(x_i) = \nu(y_{i+1})$ and $\nu(y_i) = \nu(r_i)$ holds for every $i \in [n]$. Hence, we have $\nu(y_{i+1}) \in \mathsf{reach}_M(\nu(y_i), \pi)$ for every $i \in [n]$. From $\nu(y_{n+1}) = \nu(\mathsf{cs}(\rho))$ it follows that $v' \in \mathsf{reach}_M(v, \pi^*)$. □

The following lemma dealing with the correctness of the construction of $\mathcal{M}_{\langle \pi \rangle^\omega}$ finishes the preparatory work needed in order to proof Theorem 4.1.

Lemma 4.11. *Let M be an MSC, $v \in V^M$, and π be a path expression. If Claim 4.6 holds for $\langle \pi \rangle\, \mathsf{tt}$, then*

$$M, v \models \langle \pi \rangle^\omega \iff (M, v) \in L(\mathcal{M}_{\langle \pi \rangle^\omega})$$

Proof. Let us assume that $M, v \models \langle \pi \rangle^\omega$. There exist $v_1, v_2, v_3, \ldots \in V^M$ such that $v_1 = v$ and $v_{i+1} \in \mathsf{reach}(v_i, \pi)$ for all $i \geq 1$. Since Claim 4.6 holds for $\langle \pi \rangle\, \mathsf{tt}$, there exists an accepting run $\rho_i = (C_i, E_i, r_i, \mu_i, \nu_i)$ of $\mathcal{M}_{\langle \pi \rangle \mathsf{tt}}$ on (M, v_i) with $\nu(\mathsf{cs}(\rho_i)) = v_{i+1}$ for every $i \geq 1$. The S-labelled tree $\rho = (C, E, r_1, \mu, \nu)$ with $C = \biguplus_{i \geq 1} C_i$, $E = \bigcup_{i \geq} E_i \cup \{(\mathsf{cs}(\rho_i), r_{i+1}) \mid i \geq 1\}$, $\mu = \bigcup_{i \geq 1} \mu_i$, and $\nu = \bigcup_{i \geq 1} \nu_i$ is a run of $\mathcal{M}_{\langle \pi \rangle^\omega}$ on (M, v). Let b be a branch of ρ. If there exists an $i > 1$ such that $b \in (C \setminus \{r_i\})^\infty$, then there exists a suffix b' of b such that b' is an accepting branch of ρ_j for some j with $1 \leq j < i$. Hence, b is accepting in ρ. Otherwise (i.e., b is a branch going through r_i for every $i \geq 1$), it follows from Remark 4.3 that we have $\min\{\kappa(s) \mid s \in \mathsf{inf}(b)\} = \kappa(c) = 0$. Hence, b is accepting and, therefore, ρ is accepting. The converse can be shown analogously. □

We are now able to prove our main theorem of this section.

Proof of Theorem 4.1. By an easy analysis of our construction, one can see that, for all local formulas α, the automaton $\mathcal{M}_\alpha$ can be constructed in polynomial time and that its size is linear in the size of α.

Now, we inductively show that $L(\alpha) = L(\mathcal{M}_\alpha)$ for every local formula α. Let us first consider the base cases. If $\alpha = \sigma$ with $\sigma \in \Sigma$, then it is easily checked that $L(\alpha) = L(\mathcal{M}_\alpha)$. By simple inspection, it also follows that Claim 4.6 holds for $\alpha = \langle D \rangle\, \mathsf{tt}$ with $D \in \mathbb{M}$. Regarding the induction step, we need to distinguish the following cases: If $\alpha = \neg\beta$, the claim follows from Theorem 3.6. By Lemma 4.11, we have $L(\alpha) = L(\mathcal{M}_\alpha)$ for $\alpha = \langle \pi \rangle^\omega$. Claim 4.6 holds for $\alpha = \langle \pi_1; \pi_2 \rangle\, \mathsf{tt}$ and $\alpha = \langle \pi^* \rangle\, \mathsf{tt}$ because of the lemmas 4.7, 4.8, 4.9, and 4.10. Analogously, it can be shown that Claim 4.6 is also true for the cases $\alpha = \langle \{\beta\} \rangle\, \mathsf{tt}$ and $\alpha = \langle \pi_1 + \pi_2 \rangle\, \mathsf{tt}$. Note that we have $M, v \models \alpha$ if and only if $\mathsf{reach}_M(v, \alpha) \neq \emptyset$ for all pointed MSCs (M, v). Hence, $L(\alpha) = L(\mathcal{M}_\alpha)$ holds for the above path formulas. □

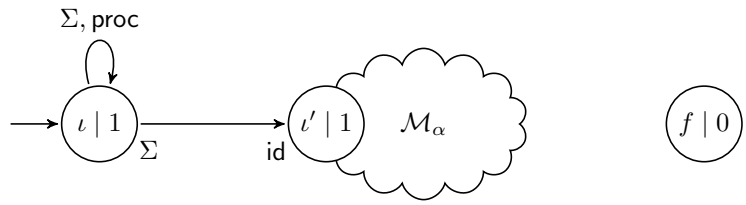

Figure 11: Illustration of the local MSCA $\mathcal{M}_{\mathsf{E}\alpha}$.

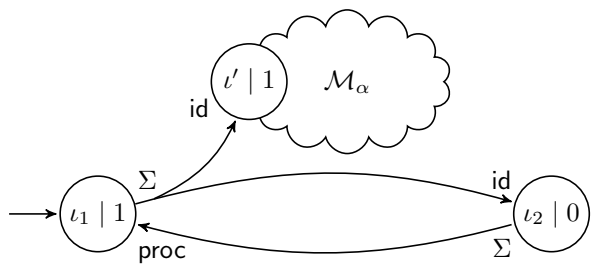

Figure 12: Illustration of the local MSCA $\mathcal{M}_{\mathsf{A}\alpha}$.

5. Translation of Global CRPDL Formulas

In this section, we demonstrate that, for every global CRPDL formula φ of the form $\mathsf{E}\alpha$ or $\mathsf{A}\alpha$, one can compute a global MSCA $\mathcal{G}_\varphi$ in polynomial time which exactly accepts the set of models of φ. Let φ be a global formula of the above form.

Case $\varphi = \mathsf{E}\alpha$. If $\mathcal{M}_\alpha = (S', \delta', \iota', c, \kappa')$, then we set $\mathcal{M}_{\mathsf{E}\alpha} = (S, \delta, \iota, c, \kappa)$ where $S = S' \uplus \{\iota, f\}$, $\kappa(s) = \kappa'(s)$ for all $s \in S'$, $\kappa(\iota) = 1$, $\kappa(f) = 0$, and, for all $s \in S$ and $\sigma \in \Sigma$,

$$\delta(s,\sigma) = \begin{cases} (\mathsf{proc}, \iota) \vee (\mathsf{id}, \iota') & \text{if } s = \iota \\ \bot & \text{if } s = f \\ \delta'(s,\sigma) & \text{otherwise} \end{cases}$$

Intuitively, the automaton $\mathcal{M}_{\mathsf{E}\alpha}$ (depicted in Fig. 11) moves forward on a process finitely many times. At some event v, it nondeterministically decides to start the automaton $\mathcal{M}_\alpha$ to check whether $(M, v) \models \alpha$ holds.

Now, $\mathcal{G}_{\mathsf{E}\alpha} = (\mathcal{M}, I)$ is meant to work as follows: it nondeterministically chooses a process on which it executes a copy of $\mathcal{M}_{\mathsf{E}\alpha}$ in state ι. On all the other processes it accepts immediately by starting $\mathcal{M}_{\mathsf{E}\alpha}$ in the sink state f with rank 0. More formally, we let $\mathcal{G}_{\mathsf{E}\alpha} = (\mathcal{M}_{\mathsf{E}\alpha}, I)$ where

$$I = \{(s_1, s_2, \ldots, s_{|\mathbb{P}|}) \mid \text{there exists } p \in \mathbb{P} \text{ such that } s_p = \iota \text{ and } s_q = f \text{ for all } p \neq q\}.$$

Case $\varphi = \mathsf{A}\alpha$. If $\mathcal{M}_\alpha = (S', \delta', \iota', c, \kappa')$, we set $\mathcal{M}_{\mathsf{A}\alpha} = (S, \delta, \iota_1, c, \kappa)$ where $S = S' \uplus \{\iota_1, \iota_2\}$, $\kappa(s) = \kappa'(s)$ for all $s \in S$, $\kappa(\iota_1) = 1$, $\kappa(\iota_2) = 0$, and

$$\delta(s, \sigma) = \begin{cases} (\mathsf{id}, \iota_2) \wedge (\mathsf{id}, \iota') & \text{if } s = \iota_1 \\ (\mathsf{proc}, \iota_1) & \text{if } s = \iota_2 \\ \delta(s, \sigma) & \text{otherwise} \end{cases}$$

Informally speaking, the automaton $\mathcal{M}_{\mathsf{A}\alpha}$ (depicted in Fig. 12) moves forward on a certain process p and checks, for every event $v \in V_p^M$ of this process, if $(M, v) \models \alpha$ holds. Note that, if $\mathcal{M}_{\mathsf{A}\alpha}$ is in state ι_2 at an event v such that there exists a successor v' of v on the same process, then $\mathcal{M}_{\mathsf{A}\alpha}$ is forced to move to v' and to change into the state ι_1. That is due to the fact that runs of local MSCAs are maximal by definition (see Definition 2.21) and because we have $\{(\mathsf{proc}, \iota_1)\} \in [\![\delta(\iota_2, \sigma)]\!]$ for every $\sigma \in \Sigma$.

We define $\mathcal{G}_{\mathsf{A}\alpha} = (\mathcal{M}_{\mathsf{A}\alpha}, I)$ where $I = \{(\iota_1, \iota_1, \ldots, \iota_1)\}$. That means $\mathcal{G}_{\mathsf{A}\alpha}$ ensures $(M, v) \models \alpha$ for every $v \in M$ by starting $\mathcal{M}_{\mathsf{A}\alpha}$ in the state ι_1 on every process.

Using Theorem 4.1 and by simple inspection of the above construction, the following theorem can be shown.

Theorem 5.1. *From a global formula φ of the form $\varphi = \mathsf{E}\alpha$ or $\varphi = \mathsf{A}\alpha$, one can construct in time $\mathsf{poly}(|\varphi|)$ a global MSCA $\mathcal{G}_\varphi$ such that, for all MSCs M, we have $M \models \varphi$ if and only if $M \in L(\mathcal{G}_\varphi)$. The size of $\mathcal{G}_\varphi$ is linear in the size of φ.*

If φ is an arbitrary global formula, then we can also construct an equivalent global MSCA $\mathcal{G}_\varphi = (\mathcal{M}, I)$. However, this time the space needed for our construction is exponential in the number of "global" conjunctions occurring in φ. In fact, the size of $\mathcal{M}$ is still linear in φ but $|I|$ is exponential in the number of conjunctive connectives occurring outside of subformulas of the form $\mathsf{E}\alpha$ and $\mathsf{A}\alpha$, respectively.

When constructing a global MSCA from an arbitrary global formula, we need to distinguish the following two additional cases:

Case $\varphi = \varphi_1 \vee \varphi_2$. Let $\mathcal{G}_{\varphi_i} = (\mathcal{M}_i, I_i)$ and $\mathcal{M}_i = (S_i, \delta_i, \iota_i, \kappa_i)$ for all $i \in [2]$. Then we define $\mathcal{G}_{\varphi_1 \vee \varphi_2} = (\mathcal{M}, I)$ where $\mathcal{M} = (S, \delta, \iota_1, \kappa)$, $S = S_1 \uplus S_2$, $\delta = \delta_1 \cup \delta_2$, $\kappa = \kappa_1 \cup \kappa_2$, and $I = I_1 \cup I_2$.

Case $\varphi = \varphi_1 \wedge \varphi_2$. Let $\mathcal{G}_{\varphi_i} = (\mathcal{M}_i, I_i)$ and $\mathcal{M}_i = (S_i, \delta_i, \iota_i, c_i, \kappa_i)$ for all $i \in [2]$. We define $\mathcal{G}_{\varphi_1 \wedge \varphi_2} = (\mathcal{M}, I)$ where

$$I = \left\{ \left((s_1, s_1'), (s_2, s_2'), \ldots, (s_{|\mathbb{P}|}, s_{|\mathbb{P}|}') \right) \mid (s_1, s_2, \ldots, s_{|\mathbb{P}|}) \in I_1, (s_1', s_2', \ldots, s_{|\mathbb{P}|}') \in I_2 \right\},$$

$\mathcal{M} = (S_1 \uplus S_2 \uplus S, \delta, \iota_1, c_1, \kappa)$, $S = \{(s_1, s_2) \mid s_1 \in S_1, s_2 \in S_2\}$, $\kappa = \kappa_1 \cup \kappa_2 \cup \{(s, 1) \mid s \in S\}$, and, for all $s \in S_1 \cup S_2 \cup S$, $s_1 \in S_1$, $s_2 \in S_2$, and $\sigma \in \Sigma$:

$$\delta(s, \sigma) = \begin{cases} (\mathsf{id}, s_1) \wedge (\mathsf{id}, s_2) & \text{if } s = (s_1, s_2) \in S \\ \delta_1(s, \sigma) & \text{if } s \in S_1 \\ \delta_2(s, \sigma) & \text{if } s \in S_2 \end{cases}$$

Together with Theorem 5.1, we obtain:

Corollary 5.2. *From a global formula φ, one can construct in time $2^{\mathsf{poly}(|\varphi|)}$ a global MSCA $\mathcal{G}_\varphi$ such that, for all MSCs M, we have $M \models \varphi$ if and only if $M \in L(\mathcal{G}_\varphi)$. The size of $\mathcal{G}_\varphi$ is exponential in the size of φ.*

6. The Satisfiability Problem

We strive for an algorithm that decides, given a global formula φ, whether $L(\varphi) \neq \emptyset$ holds. Unfortunately, the satisfiability problem of CRPDL is undecidable. This follows from results concerning Lamport diagrams which can be easily transferred to MSCs [17]. However, if one only considers existentially B-bounded MSCs [20, 16, 10, 9], then the problem becomes decidable. Intuitively, an MSC M is *existentially B-bounded* if its events can be scheduled in such a way that at every moment no communication channel contains more than B pending messages (see definition below). The rest of this section prepares the proof of our main theorem which is stated in the following. The proof itself can be found on page 32.

Theorem 6.1. *The following problem is PSPACE-complete:*

Input: $B \in \mathbb{N}$ (given in unary) and a global CRPDL formula φ
Question: Is there an existentially B-bounded MSC satisfying φ?

6.1. From MSCAs to Word Automata.

In order to be able to give uniform definitions of automata over MSCs and words, respectively, we also consider words over an alphabet Γ as labelled relational structures. For this, we fix the set $\mathbb{W} = \{\mathsf{prev}, \mathsf{next}, \mathsf{id}\}$ of directions.

Definition 6.2. Let Γ be an arbitrary alphabet. A *word-like structure over* Γ is a structure $W = (V^W, \mathsf{next}^W, \lambda^W)$ where

- V^W is a set of *positions*,
- $\mathsf{next}^W \subseteq (V^W \times V^W)$,
- $\lambda^W \colon V^W \to \Gamma$ is a labeling function,
- next^W is the direct successor relation of a linear order $\preceq^W$ on V^W,
- $(V^W, \preceq^W)$ is finite or isomorphic to $(\mathbb{N}, \leq)$

The word-like structure W induces a partial function $\eta_W \colon (V^W \times V^W) \to \mathbb{W}$. For all $v, v' \in V^W$, we define

$$\eta_W(v, v') = \begin{cases} \mathsf{next} & \text{if } (v, v') \in \mathsf{next}^W \\ \mathsf{prev} & \text{if } (v', v) \in \mathsf{next}^W \\ \mathsf{id} & \text{if } v = v' \\ \text{undefined} & \text{otherwise} \end{cases}$$

Every finite word $W = \gamma_1\gamma_2 \dots \gamma_n \in \Gamma^*$ gives rise to a unique (up to isomorphism) word-like structure $\overline{W}$ where $V^{\overline{W}} = [n]$, $\mathsf{next}^{\overline{W}} = \{(i, i+1) \mid 1 \leq i < n\}$, and $\lambda^{\overline{W}}(i) = \gamma_i$ for all $i \in [n]$. Analogously, every infinite word $W \in \Gamma^\omega$ induces a word-like structure $\overline{W}$. In the following, we identify W and $\overline{W}$ for every word $W \in \Gamma^\infty$.

We now formalize the notion of existentially B-bounded MSCs.

Definition 6.3. If M is an MSC and W is a word, then W is a *linearization* of M if $V^M = V^W$, $\lambda^M = \lambda^W$, and $(\mathsf{msg}^M \cup \bigcup_{p\in\mathbb{P}} \mathsf{proc}_p^M)^* \subseteq \preceq^W$. The word W is *B-bounded* if we have

$$|\{v' \mid v' \preceq^W v, \lambda^W(v') = p!q\}| - |\{v' \mid v' \preceq^W v, \lambda^W(v') = q?p\}| \;\leq\; B\,,$$

for every $v \in V^W$ and $(p,q) \in \mathsf{Ch}$. An MSC M is *existentially B-bounded* if there exists a B-bounded linearization of M, i.e., if it allows for an execution with B-bounded channels.

Example 6.4. Let M be the MSC from Fig. 1. The word

$$W = (1!2)\,(1!2)\,(1!2)\,(2?1)\,(2!1)\,(2?1)\,(2!1)\,(2?1)\,(2!1)\,(1?2)\,(1?2)\,(1?2) \;\in\; \Sigma^*$$

is a 3-bounded linearization of M. Note that parentheses have been introduced for readability. There is even a 1-bounded linearization of M:

$$W' = (1!2)\,(2?1)\,(1!2)\,(2!1)\,(2?1)\,(1!2)\,(1?2)\,(2!1)\,(2?1)\,(1?2)\,(2!1)\,(1?2) \;\in\; \Sigma^*$$

Hence, W' witnesses the fact that M is existentially 1-bounded.

We define two-way alternating automata over words in the style of local MSCAs.

Definition 6.5. A *two-way alternating parity automaton* (or *2APA* for short) is a quadruple $\mathcal{P} = (S, \delta, \iota, \kappa)$ where

- S is a finite set of states,
- $\delta\colon (S \times \Sigma) \to \mathcal{B}^+(\mathbb{W} \times S)$ is a transition function,
- $\iota \in S$ is an initial state, and
- $\kappa\colon S \to \{0, 1, \ldots, m-1\}$ is a ranking function with $m \in \mathbb{N}$.

The *size* of $\mathcal{P}$ is $|S| + |\delta|$. If $|\tau| = 1$ for all $\tau \in [\![\delta(s,\sigma)]\!]$, $s \in S$, and $\sigma \in \Sigma$ (i.e., $\mathcal{P}$ does not make use of universal branching), then $\mathcal{P}$ is called a *two-way parity automaton* (or *2PA*). If W is a word, then the definition of an S-labelled tree over W is analogous to the definition of an S-labelled tree over a pointed MSC (cf. Definition 2.17). Furthermore, an (accepting) run of a 2APA is defined in a similar way as it is defined for a local MSCA (cf. Definitions 2.20, 2.21, and 2.22). By $L(\mathcal{P})$, we denote the set of words W for which there exists an accepting run of $\mathcal{P}$ on (W, v) where v is the minimal element from V^W with respect to $\preceq^W$.

Now, let us fix a channel bound $B \in \mathbb{N}$ and the alphabet $\Gamma = \Sigma \times \{0, 1, \ldots, B-1\}$.

Definition 6.6. If W is a B-bounded word over Σ, then we associate with W the unique B-bounded word W_B over Γ where $V^W = V^{W_B}$, $\mathsf{next}^W = \mathsf{next}^{W_B}$, and, for every $v \in V^W$, we have $\lambda^{W_B}(v) = (\lambda^W(v), i)$ with $i = |\{v' \in V^W \mid v' \prec^W v, \lambda^W(v) = \lambda^W(v')\}| \bmod B$.

That means that, in the second component of the labels in W_B, we count events labelled by the same action modulo B.

Example 6.7. Let W and W' be the words from Example 6.4. For instance, W_3 is the word

$$(1!2,0)\,(1!2,1)\,(1!2,2)\,(2?1,0)\,(2!1,0)\,(2?1,1)\,(2!1,1)\,(2?12)\,(2!1,2)\,(1?2,0)\,(1?2,1)\,(1?2,2)$$

whereas W'_2 is given by:

$$(1!2,0)\,(2?1,0)\,(1!2,1)\,(2!1,0)\,(2?1,1)\,(1!2,0)\,(1?2,0)\,(2!1,1)\,(2?1,0)\,(1?2,1)\,(2!1,0)\,(1?2,0)$$

In W_B, we are able to quickly locate matching send and receive events. For example, if v is a send event of W_B labelled by $(p!q, i)$, we just need to move to the smallest event $v' \in V^{W_B}$ (with respect to $\preceq^W$) with $v \preceq^{W_B} v'$ and $\lambda^{W_B}(v') = (q?p, i)$.

Let $\mathcal{G} = (\mathcal{M}, I)$ be a global MSCA. We can construct a 2APA $\mathcal{P}_\mathcal{G} = (S, \delta, \iota, \kappa)$ that accepts exactly the set of words W_B where W is a B-bounded linearization of an MSC from $L(\mathcal{G})$. In order to construct $\mathcal{P}_\mathcal{G}$, there is one issue which needs to be addressed. Let M be an MSC and W be a B-bounded linearization of M. If $v, v' \in V^M$ with $\eta_M(v, v') = \mathsf{proc}$, then a local MSCA is capable of directly moving to v'. In general, this cannot be accomplished by a 2APA running on W_B since there may exist events $v'' \in V^M$ with $v \prec^{W_B} v'' \prec^{W_B} v'$. To circumvent this limitation, the idea is to introduce transitions which allow the 2APA to move forward on W_B and skip non-relevant events until it reaches the event v'. Of course, we have to analogously deal with proc^{-1}, msg, and msg^{-1} transitions of local MSCAs.

More precisely, regarding the 2APA $\mathcal{P}_\mathcal{G}$, we use states of the form (s, p, next) to remember that we are searching for the next event on process p in the next-direction. In contrast, a state of the form $(s, p!q, i, \mathsf{prev})$ means that we are looking for the nearest send event $p!q$ indexed by i in the prev-direction. The first component is always used to remember the state from which we need to continue the simulation of the local MSCA $\mathcal{M}$ after finding the correct event. If $\mathcal{M} = (S', \delta', \iota', \kappa')$, then the set of states of $\mathcal{P}_\mathcal{G}$ is the following:

$$\begin{aligned} S = &\{\iota, t\} \cup S' \cup \{(s, p, \mathsf{prev}), (s, p, \mathsf{next}) \mid s \in S', p \in \mathbb{P}\} \\ &\cup \{(s, \sigma, i, \mathsf{prev}), (s, \sigma, i, \mathsf{next}) \mid s \in S', \sigma \in \Sigma, 0 \leq i < B\} \end{aligned}$$

The intuition for the states from I is as follows: From the initial state ι, the 2APA $\mathcal{P}_\mathcal{G}$ nondeterministically changes into a global initial state $(\iota_1, \iota_2, \ldots, \iota_{|\mathbb{P}|})$ from I. That way, it simulates $|\mathbb{P}|$ many copies of $\mathcal{M}$ where the p-th copy of $\mathcal{M}$ is started in the state ι_p in the minimal event of process p (with respect to $\preceq_p^M$). More formally, for all $\gamma \in \Gamma$, we define

$$\delta(\iota, \gamma) = \bigvee_{(\iota_1, \ldots, \iota_{|\mathbb{P}|}) \in I} \big(\mathsf{id}, (\iota_1, 1, \mathsf{next})\big) \wedge \big(\mathsf{id}, (\iota_2, 2, \mathsf{next})\big) \wedge \ldots \wedge \big(\mathsf{id}, (\iota_{|\mathbb{P}|}, |\mathbb{P}|, \mathsf{next})\big) .$$

Assume that the automaton $\mathcal{P}_\mathcal{G}$ is in a state of the form (s, p, D) resp. (s, σ, i, D) at an event v. If $\lambda^M(v) \notin \Sigma_p \times \{0, \ldots, B-1\}$ resp. $\lambda^M(v) \neq (\sigma, i)$, i.e., if v is not the event at which the simulation of $\mathcal{M}$ needs to be continued, then we stay in the current state and move into direction D. Otherwise, we simulate a transition $\tau \in [\![\delta'(s, \lambda^M(v))]\!]$ of the local MSCA $\mathcal{M}$ in the following manner: If $(\mathsf{proc}, s) \in \tau$, then we change into the state (s, p, next) and move along the next-direction. If $(\mathsf{proc}^{-1}, s) \in \tau$, then we act analogously in the prev-direction. Now, let us assume that $(\mathsf{msg}, s) \in \tau$. If $\lambda^M(v)$ is of the form $(p!q, i)$, then we change into $(s, q?p, i, \mathsf{next})$ and move along the next-direction. In contrast, if v is a receive event, then the local MSCA $\mathcal{M}$ is unable to execute the movement (msg, s). To simulate this behavior, we change into the sink state t and stay at v. From state t, the 2APA $\mathcal{P}_\mathcal{G}$ is unable to accept. If $(\mathsf{msg}^{-1}, s) \in \tau$, then we proceed similarly. Formally, for all $s \in S'$, $p \in \mathbb{P}$, $\sigma \in \Sigma$,

$i \in \{0, \dots, B-1\}$, $D \in \mathbb{W}$, and $\gamma \in \Gamma$, we have

$$\delta\big((s,p,D),\gamma\big) = \begin{cases} \big(D,(s,p,D)\big) & \text{if } \gamma \notin \Sigma_p \times \{0,\dots,B-1\} \\ (\mathsf{id}, s) & \text{if } \gamma \in \Sigma_p \times \{0,\dots,B-1\} \end{cases}$$

$$\delta\big((s,\sigma,i,D),\gamma\big) = \begin{cases} \big(D,(s,\sigma,i,D)\big) & \text{if } \gamma \neq (\sigma,i) \\ (\mathsf{id}, s) & \text{if } \gamma = (\sigma,i) \end{cases}$$

$$\delta(s,\gamma) = g(s,\gamma)$$

$$\delta(t,\gamma) = \bot$$

where, for all $s \in S'$ and $\gamma = (p\theta q, i) \in \Gamma$, $g(s,\gamma)$ is the positive Boolean expression which is obtained from $\delta'(s, p\theta q)$ by applying the following substitutions: for all $s' \in S'$, we exchange

- (proc, s') by $\big(\mathsf{next}, (s', p, \mathsf{next})\big)$,
- (proc^{-1}, s') by $\big(\mathsf{prev}, (s', p, \mathsf{prev})\big)$,
- (msg, s') by (id, t) if $\theta = ?$,
- (msg, s') by $\big(\mathsf{next}, (s', q?p, i, \mathsf{next})\big)$ if $\theta = !$,
- (msg^{-1}, s') by (id, t) if $\theta = !$, and
- (msg, s') by $\big(\mathsf{prev}, (s', q!p, i, \mathsf{prev})\big)$ if $\theta = ?$.

It remains to define the ranking function κ of $\mathcal{P}_\mathcal{G}$. For all $s \in S'$, we define $\kappa(s) = \kappa'(s)$. If $s \in S \setminus S'$, then we set $\kappa(s) = m$ where m is the smallest odd natural number larger than $\max_{s \in S'} \kappa'(s)$.

Theorem 6.8. *Let M be an MSC and W some B-bounded linearization of M. We have $M \in L(\mathcal{G})$ if and only if $W_B \in L(\mathcal{P}_\mathcal{G})$. The size of $\mathcal{P}_\mathcal{G}$ is polynomial in B and the size of $\mathcal{G}$.*

Proof sketch. If ρ is a successful run of $\mathcal{P}_\mathcal{G}$ on an MSC M, then ρ immediately splits into $|\mathbb{P}|$ many subtrees ρ_q. By easy inspection of the transition function of $\mathcal{P}_\mathcal{G}$ it follows that there exists a global initial state $(\iota_1, \dots, \iota_{|\mathbb{P}|}) \in I$ such that, for every $q \in \mathbb{P}$, there is exactly one subtree $\rho_q = (C_q, E_q, r_q, \mu_q, \nu_q)$ with $\mu_q(r_q) = (\iota_q, q, \mathsf{next})$. Each of these subtrees ρ_q can be pruned in such a way that one obtains an accepting run ρ'_q of $\mathcal{M}$ starting in state ι_q from the minimal event of V_q^M (with respect to $\preceq_q^M$). Thus, M is accepted by $\mathcal{G}$. Note that we obtain ρ'_q from ρ_q by essentially removing all configurations x with $\mu_q(x) \notin S'$; of course, we need to update E_q accordingly.

The converse can be shown analogously. Basically, one only needs to pad and combine the accepting runs of the local MSCA $\mathcal{M}$ on the different processes in order to obtain a successful run of $\mathcal{P}_\mathcal{G}$. □

6.2. **Checking the Emptiness of 2APAs.** In order to solve the emptiness problem for a 2APA $\mathcal{P}$, we transform $\mathcal{P}$ into a Büchi automaton.

Definition 6.9. Formally, a *Büchi automaton* (or *BA*) over the alphabet Σ is a tuple $\mathcal{B} = (S, \Delta, \iota, F)$ where S is a finite set of states, ι is the initial state, $F \subseteq S$ is the set of final states, and $\Delta \subseteq S \times \Sigma \times S$ is the transition relation. The *size* of $\mathcal{B}$ is $|S| + |\Delta|$. Let $W = \sigma_0\sigma_1 \ldots \in \Sigma^\infty$ be a word of length $n \in \mathbb{N} \cup \{\infty\}$. The mapping $r \colon \mathbb{N} \to S$ is a *run of $\mathcal{B}$ on W* if $r(0) = \iota$ and $\big(r(i), \sigma_i, r(i+1)\big) \in \Delta$ for all $i < n$. A word W is *accepted* by $\mathcal{B}$ if there exists a run r such that $r(0)r(1)r(2)\ldots \in S^\infty$ is *Büchi accepting*, i.e., if one of the following conditions is fulfilled:

(1) $n \in \mathbb{N}$ and $r(n) \in F$
(2) $\mathsf{inf}\big(r(0)r(1)r(2)\dots\big) \cap F \neq \emptyset$

By $L(\mathcal{B})$, we denote the set of words which are accepted by $\mathcal{B}$.

In contrast to common definitions of Büchi automata, item (1) allows $\mathcal{B}$ to accept finite words as well. In the following, we also need to deal with two-way (alternating) Büchi automata (2ABA and 2BA for short) which are defined analogously to 2APA and 2PA but implement the Büchi acceptance condition instead of the parity acceptance condition.

Definition 6.10. More precisely, a *two-way alternating Büchi automaton* (*2ABA* for short) is a tuple $\mathcal{B} = (S, \delta, \iota, F)$ where S, δ, and ι are defined as for 2APA's and $F \subseteq S$ is the set of final states. An (accepting) run of $\mathcal{B}$ is defined in a similar way as it is defined for a 2APA with the following modification: A sequence of states $(s_i)_{i\geq 1} \in S^\infty$ is accepting if and only if it is Büchi accepting. A *two-way Büchi automaton* (*2BA*) is defined analogously to a 2PA.

Remark 6.11. Note that using the ideas from [14], a 2APA $\mathcal{P}$ can be transformed into a 2ABA $\mathcal{B}$ in polynomial space such that the size of $\mathcal{B}$ is polynomial in the size of $\mathcal{P}$ and $L(\mathcal{P}) = L(\mathcal{B})$.

In [4], Dax and Klaedtke showed the following:

Theorem 6.12 ([4])**.** *From a 2APA $\mathcal{P}$, one can construct a BA $\mathcal{B}$ whose size is exponential in the size of $\mathcal{P}$ such that $L(\mathcal{P}) = L(\mathcal{B})$.*

Note that Dax and Klaedtke actually stated that one can construct a BA of size $2^{O((nk)^2)}$ where n is the size of $\mathcal{P}$ and $2k$ is the maximal rank of a state from $\mathcal{P}$. Since we can assume that the maximal rank of a state of $\mathcal{P}$ is linear in the number of states of $\mathcal{P}$, it follows that the size of $\mathcal{B}$ is exponential in the size of $\mathcal{P}$. Furthermore, in [4], only infinite words are considered. Nevertheless, it can be easily seen that the result also applies to automata recognizing infinite and finite words at the same time.

In the following, we recall parts of the proof of Theorem 6.12 and adapt it to our setting in order to be able to prove Prop. 6.13. Let $\mathcal{B} = (S, \delta, \iota, F)$ be an 2ABA over the alphabet Σ and let Γ be an abbreviation of the function space $S \to 2^{\mathbb{W}\times S}$. If W is a word and $\rho = (C, E, r, \mu, \nu)$ is an accepting run of $\mathcal{P}$ on W, then the authors of [4] argue that we can assume without loss of generality that all nodes x and y of ρ with $\mu(x) = \mu(y)$ and $\nu(x) = \nu(y)$ exhibit isomorphic subtrees. Hence, ρ can be thought of as a directed acyclic graph (DAG) which can be represented as a (possibly infinite) word of functions $f = f_1 f_2 \dots \in \Gamma^\infty$ where $f_j(q) = \mathsf{tr}_\rho(x)$ (cf. Def. 2.21), $\mu(x) = q$, and $\nu(x)$ is the j-th position of W with respect to next^W. From $\mathcal{B}$, an intermediate 2BA $\mathcal{B}' = (S, \delta', \iota, S \setminus F)$ over the alphabet $\Sigma \times \Gamma$ is constructed where, for all $s \in S$, $\sigma \in \Sigma$, and $f \in \Gamma$, we have

$$\delta'\big(s, (\sigma, f)\big) = \begin{cases} \bigvee_{(D,s')\in f(s)} (D, s') & \text{if } f(s) \in [\![\delta(s,\sigma)]\!] \\ (\mathsf{next}, s) & \text{otherwise.} \end{cases}$$

Note that the automaton $\mathcal{B}'$ is of exponential size since the size of the alphabet Γ is exponential in the size of $\mathcal{B}$. However, the set of states of $\mathcal{B}'$ equals the set of states of $\mathcal{B}$. It is shown that $\mathcal{B}'$ rejects exactly those words $(\sigma_0, f_0)(\sigma_1, f_1)\dots \in (\Sigma \times \Gamma)^\infty$ where the function word $(f_i)_{i\geq 0}$ represents an accepting run of $\mathcal{B}$ on $(\sigma_i)_{i\geq 0}$. In the course of the proof of Theorem 6.12, using [24, Theorem 4.3], a Büchi automaton $\mathcal{B}''$ whose size is exponential

in $\mathcal{B}$ with $L(\mathcal{B}'') = (\Sigma \times \Gamma)^\infty \setminus L(\mathcal{B}')$ is constructed. It is shown that the projection of $L(\mathcal{B}'')$ to the alphabet Σ equals $L(\mathcal{B})$.

We also recall the essential parts of the proof of Theorem 4.3 of [24], apply a minor correction and adapt it to our setting. Let $\mathcal{B} = (S, \delta, \iota, F)$ be a 2BA. By $\mathsf{bwl}(\mathcal{B})$, we denote the set $S^2 \times \{0,1\}$. Intuitively, a triple $(s,t,b) \in \mathsf{bwl}(\mathcal{B})$ expresses that at the current position there is a backward loop starting in state s and ending in state t. We have $b = 1$ if and only if this loop visits a final state. A word $(\sigma_1\sigma_2\ldots, m_0m_1\ldots, n_0n_1\ldots) \in (\Sigma \times \mathsf{bwl}(\mathcal{B}) \times 2^S)^\infty$ of length $h \in \mathbb{N} \cup \{\infty\}$ is $\mathcal{B}$-*legal* if and only if there exists a sequence $\ell_0\ell_1\ldots \in \mathsf{bwl}(\mathcal{B})^\infty$ of length h such that the following conditions are fulfilled:

- $(s,t,0) \in \ell_i$ if and only if either $\{(\mathsf{id},t)\} \in [\![\delta(s,\sigma_i)]\!]$ or $i \geq 1$ and there are states $s', t' \in S$ and $b \in \{0,1\}$ such that $(s',t',b) \in m_{i-1}$, $\{(s',\mathsf{prev})\} \in [\![\delta(s,\sigma_i)]\!]$, and $\{(t,\mathsf{next})\} \in [\![\delta(t',\sigma_{i-1})]\!]$
- $(s,t,1) \in \ell_i$ if and only if either $\{(\mathsf{id},t)\} \in [\![\delta(s,\sigma_i)]\!]$ and $t \in F$ or $i \geq 1$ and there are states $s', t' \in S$ and $b \in \{0,1\}$ such that $(s',t',b) \in m_{i-1}$, $\{(s',\mathsf{prev})\} \in [\![\delta(s,\sigma_i)]\!]$, $\{(t,\mathsf{next})\} \in [\![\delta(t',\sigma_{i-1})]\!]$, and in addition either $b = 1$ or $\{s',t',t\} \cap F \neq \emptyset$
- $(s,t,0) \in m_i$ if and only if there are $s_0, s_1, \ldots, s_k \in S$ and $b_0, b_1, \ldots, b_{k-1} \in \{0,1\}$ with $k > 0$ such that $s_0 = s$, $s_k = t$, and $(s_j, s_{j+1}, b_j) \in \ell_i$ for all $0 \geq j > k$
- $(s,t,1) \in m_i$ if and only if there are $s_0, s_1, \ldots, s_k \in S$ and $b_0, b_1, \ldots, b_{k-1} \in \{0,1\}$ with $k > 0$ such that $s_0 = s$, $s_k = t$, $(s_j, s_{j+1}, b_j) \in \ell_i$ for all $0 \geq j > k$, and $\{b_0, b_1, \ldots, b_{k_1}\} \cap \{1\} \neq \emptyset$
- $s \in n_i$ if and only if there exists a state $s' \in S$ and $b \in \{0,1\}$ such that $(s,s',b) \in m_i$ and one of the following conditions holds:
  - $(s',s',1) \in m_i$
  - $s' \in F$ and $\mathcal{B}$ cannot make a transition at position i in state s'
  - $i \geq 1$ and there exists a state $s'' \in S$ such that $\{(s'',\mathsf{prev})\} \in [\![\delta(s',\sigma_i)]\!]$ and $s'' \in n_{i-1}$

Note that the ℓ_i's are only used to simplify the definition of the m_i's. The introduction of the n_i's is a minor correction of the proof of Theorem 4.3. Intuitively, we have $s \in n_i$ if there exists a position $j \leq i$ and a state $s' \in S$ such that there exists a backward run starting in s allowing $\mathcal{B}$ to visit the j-th position of the input word in state s' such that the following holds: either $s' \in F$ and $\mathcal{B}$ cannot make a transition at position j in state s' or, at position j in state s', the automaton $\mathcal{B}$ can enter infinitely often a loop containing a final state. Note that without the information contained in the n_i's, we would not capture accepting runs of $\mathcal{B}$ which do not visit all positions of the input word but, at some position i, go backward and then accept without returning to i again.

From the 2BA $\mathcal{B} = (S, \delta, \iota, F)$, we can construct a BA $\mathcal{B}_1$ recognizing the set of all $\mathcal{B}$-legal words. Let $\mathcal{B}_1 = (S_1, \Delta_1, \iota_1, F_1)$ be the BA where $S_1 = 2^{S^2} \times \mathsf{bwl}(\mathcal{B}) \times 2^S$, $\iota_1 = (\emptyset, \emptyset, \emptyset)$, $F_1 = S_1$ and, for all $(p', \overline{m}', \overline{n}'), (p, \overline{m}, \overline{n}) \in S_1$ and $(\sigma, m, n) \in \Sigma \times \mathsf{bwl}(\mathcal{B}) \times S$, we have $\big((p', \overline{m}', \overline{n}'), (\sigma, m, n), (p, \overline{m}, \overline{n})\big) \in \Delta_1$ if and only if there exists $\ell \subseteq \mathsf{bwl}(\mathcal{B})$ such that the following conditions hold:

- $\overline{m} = m$, $\overline{n} = n$,
- $(s,t) \in p$ if and only if $\{(\mathsf{next},t)\} \in [\![\delta(s,\sigma)]\!]$
- $(s,t,0) \in \ell$ if and only if $\{(\mathsf{id},t)\} \in [\![\delta(s,\sigma)]\!]$ or there are states $s', t' \in S$ and $b \in \{0,1\}$ such that $(s',t',b) \in \overline{m}'$, $\{(\mathsf{prev},s')\} \in [\![\delta(s,\sigma)]\!]$, and $(t',t) \in p'$
- $(s,t,1) \in \ell$ if and only if $\{(\mathsf{id},t)\} \in [\![\delta(s,\sigma)]\!]$ and $t \in F$ or there are states $s', t' \in S$ and $b \in \{0,1\}$ such that $(s',t',b) \in \overline{m}'$, $\{(\mathsf{prev},s')\} \in [\![\delta(s,\sigma)]\!]$, $(t',t) \in p'$, and in addition either $b = 1$ or $\{s',t',t\} \cap F \neq \emptyset$

- $(s,t,0) \in m$ if and only if there are $s_0, s_1, \ldots, s_k \in S$ and $b_0, b_1, \ldots, b_{k-1} \in \{0,1\}$ with $k > 0$ such that $s_0 = s$, $s_k = t$, and $(s_j, s_{j+1}, b_j) \in \ell$ for all $0 \geq j > k$
- $(s,t,1) \in m$ if and only if there are $s_0, s_1, \ldots, s_k \in S$ and $b_0, b_1, \ldots, b_{k-1} \in \{0,1\}$ with $k > 0$ such that $s_0 = s$, $s_k = t$, $(s_j, s_{j+1}, b_j) \in \ell$ for all $0 \geq j > k$, and $\{b_0, b_1, \ldots, b_{k-1}\} \cap \{1\} \neq \emptyset$
- $s \in n$ if and only if there exists a state $s' \in S$ and $b \in \{0,1\}$ such that $(s, s', b) \in m$ and one of the following holds:
  - $(s', s', 1) \in m$
  - $s' \in F$ and $\mathcal{B}$ cannot make a transition at the current position in state s'
  - there exists a state $s'' \in S$ such that $\{(s'', \mathsf{prev})\} \in [\![\delta(s', \sigma)]\!]$ and $s'' \in \overline{n}'$

It remains to specify a BA $\mathcal{B}_2$ such that a $\mathcal{B}$-legal word $(\sigma_0\sigma_1 \ldots, m_0 m_1 \ldots, n_0 n_1 \ldots) \in (\Sigma \times \mathsf{bwl}(\mathcal{B}) \times 2^S)^\infty$ is accepted by $\mathcal{B}_2$ if and only if $(\sigma_i)_{i \geq 0}$ is accepted by the 2BA $\mathcal{B}$. Let $\mathcal{B}_2 = (S_2, \Delta_2, \iota_2, F_2)$ where

- $S_2 = (S \cup \{\bot\}) \times \{0,1\}$,
- $\iota_2 = (\iota, b)$ with $b = 1$ if and only if $\iota \in F$,
- $F_2 = (S \cup \{\bot\}) \times \{1\}$, and,
- we have $\big((s,b), (\sigma, m, n), (s', b')\big) \in \Delta_2$ if and only if one of the following conditions is fulfilled:
  - there exist $s'' \in S$ and $b'' \in \{0,1\}$ such that $(s, s'', b'') \in m$, $\{(s', \mathsf{next})\} \in [\![\delta(s'', \sigma)]\!]$, and ($b' = 1$ if and only if $b'' = 1$ or $s' \in F$)
  - $s \in n$ and $s' = \bot$
  - $s' = t' = \bot$

Intuitively, the states of $\mathcal{B}_2$ come with a flag. The flag is set to 1 if and only if the simulated automaton $\mathcal{B}$ just visited a final state. It can be shown that the projection of $L(\mathcal{B}_1) \cap L(\mathcal{B}_2)$ to the alphabet Σ equals the language of $\mathcal{B}$.

Proposition 6.13. *If $\mathcal{P}$ is a 2APA, then one can check the emptiness of $L(\mathcal{P})$ in polynomial space.*

Proof sketch. By Remark 6.11, we can transform $\mathcal{P}$ in polynomial space into a 2ABA recognizing the same language. By Theorem 6.12, we can construct a BA $\mathcal{B}' = (S, \Delta, \iota, F)$ whose size is exponential in the size of $\mathcal{P}$ such that $L(\mathcal{B}') = L(\mathcal{P})$. Clearly, remembering a state of $\mathcal{B}'$ requires only polynomial space. By inspecting the construction of $\mathcal{B}'$, one can see that $\mathcal{B}'$ can be obtained in space polynomial in the size of $\mathcal{P}$. This means in particular: given two states $s, t \in S$ and $\sigma \in \Sigma$, one can check in polynomial space whether $(s, \sigma, t) \in \Delta$ holds. Since $L(\mathcal{B}')$ is non-empty if there exists a final state $s \in F$ which is reachable from ι (recall that our Büchi automata also accept finite words), the emptiness problem of $\mathcal{P}$ can be solved in polynomial space. □

6.3. **The Decision Procedure.** We are now able to prove our main theorem:

Proof of Theorem 6.1. The global formula φ is a positive Boolean combination of global formulas $\varphi_1, \ldots, \varphi_n$ where, for every $i \in [n]$, φ_i is of the form $\mathsf{A}\alpha_i$ or $\mathsf{E}\alpha_i$ for some local formula α_i. It follows from Theorem 5.1 that we can construct in polynomial space a global MSCAs $\mathcal{G}_i$ such that $L(\varphi_i) = L(\mathcal{G}_i)$ and the size of $\mathcal{G}_i$ is linear in the size of φ_i for every $i \in [n]$. By Theorem 6.8, we can construct, for every $i \in [n]$, a 2APA $\mathcal{P}_i$ such that, for all MSCs M and B-bounded linearizations W of M, we have $M \in L(\varphi_i)$ if and only if

$W_B \in L(\mathcal{P}_i)$. By simple inspection of the construction of Sect. 6.1, one can see that $\mathcal{P}_i$ can be obtained in polynomial space. The number of states of $\mathcal{P}_i$ is also polynomial. Using standard automata constructions for alternating automata, we can combine the automata $\mathcal{P}_1, \ldots, \mathcal{P}_n$ according to the construction of φ to obtain a 2APA $\mathcal{P}_\varphi$ such that, for all MSCs M and B-bounded linearizations W of M, we have $W_B \in L(\mathcal{P}_\varphi)$ if and only if $M \in L(\varphi)$. This can be accomplished in polynomial space and the number of states of $\mathcal{P}_\varphi$ is also polynomial in B and the size of φ. Clearly, φ is satisfiable by an existentially B-bounded MSC if and only if $L(\mathcal{P})$ is non-empty. Hence, by Prop. 6.13, the satisfiability problem of φ can be decided in polynomial space. The hardness result follows from the PSPACE-hardness of the satisfiability problem of LTL. □

7. The Model Checking Problem

A communicating finite-state machine (also known as message-passing automaton) is well suited to model the behavior of a distributed system. It consists of a finite number of finite automata communicating using order-preserving channels. To be more precise, we recapitulate the definition from [1].

Definition 7.1. A *communicating finite-state machine* (or *CFM* for short) is a structure $\mathcal{C} = (H, (\mathcal{T}_p)_{p\in\mathbb{P}}, F)$ where

- H is a finite set of *message contents*,
- for every $p \in \mathbb{P}$, $\mathcal{T}_p = (S_p, \rightarrow_p, \iota_p)$ is a finite labelled transition system over the alphabet $\Sigma_p \times H$ (i.e., $\rightarrow_p \subseteq S_p \times \Sigma_p \times H \times S_p$) with initial state $\iota_p \in S_p$,
- $F \subseteq \prod_{p\in\mathbb{P}} S_p$ is a set of global final states.

Let $\mathcal{C}$ be a CFM and M be an MSC. A *run* of $\mathcal{C}$ on M is a pair (ζ, χ) of mappings $\zeta : V^M \to \bigcup_{p\in\mathbb{P}} S_p$ and $\chi : V^M \to H$ such that, for all $v \in V^M$,

- $\chi(v) = \chi(v')$ if there exists $v' \in V^M$ with $\eta_M(v, v') = \mathsf{msg}$,
- $(\zeta(v'), \lambda(v), \chi(v), \zeta(v)) \in \rightarrow_{P_M(v)}$ if there exists $v' \in V^M$ with $\eta_M(v', v) = \mathsf{proc}$, and $(\iota_p, \lambda(v), \chi(v), \zeta(v)) \in \rightarrow_{P_M(v)}$ otherwise.

Let $\mathsf{cofin}_\zeta(p) = \{s \in S_p \mid \forall v \in V_p^M \exists v' \in V_p^M : v \prec_p^M v' \wedge \zeta(v') = s\}$. The run (ζ, χ) is *accepting* if there is some $(s_p)_{p\in\mathbb{P}} \in F$ such that $s_p \in \mathsf{cofin}_\zeta(p)$ for all $p \in \mathbb{P}$. The *language* of $\mathcal{C}$ is the set $L(\mathcal{C})$ of all MSCs M for which there exists an accepting run.

We now demonstrate that the bounded model checking problem for CFMs and CRPDL is PSPACE-complete.

Theorem 7.2. *The following problem is PSPACE-complete:*

Input: $B \in \mathbb{N}$ (given in unary), CFM $\mathcal{C}$, and a global CRPDL formula φ.
Question: Is there an existentially B-bounded MSC $M \in L(\mathcal{C})$ with $M \models \varphi$?

Proof. In [1], it was shown that one can construct in polynomial space a Büchi automaton $\mathcal{B}_\mathcal{C}$ from $\mathcal{C}$ which recognizes exactly the set of all B-bounded linearizations of the MSCs from $L(\mathcal{C})$. Its number of states is polynomial in the maximal number of local states a transition system of $\mathcal{C}$ has and exponential in B. In the proof of Theorem 6.1, we already constructed in polynomial space a Büchi automaton $\mathcal{B}_\varphi$ of exponential size accepting the set of all B-bounded linearizations of the MSCs satisfying φ. Hence, the model checking problem can be decided in polynomial space. The PSPACE-hardness follows from the PSPACE-hardness of the satisfiability problem. □

Remark 7.3. The model checking problem for CRPDL and high-level message sequence charts (HMSCs) asks, given an HMSC $\mathcal{H}$ and a global CRPDL formula φ, is there an MSC $M \in L(\mathcal{H})$ with $M \models \varphi$. Using techniques from [1] and the ideas from the proof of Theorem 7.2, it can be shown that this problem is also PSPACE-complete.

8. Open Questions

It is an interesting open question whether the bounded model checking problem of CFMs and CRPDL enriched with the intersection operator [11, 1] is still in PSPACE. It also needs to be investigated whether PDL is a proper fragment of CRPDL and if CRPDL and global MSCAs are expressively equivalent. Furthermore, we would like to know more about the expressive power of CRPDL and global MSCAs in general, especially in comparison with the existential fragment of monadic second-order logic (EMSO).

References

[1] B. Bollig, D. Kuske, and I. Meinecke. Propositional dynamic logic for message-passing systems. *Logical Methods in Computer Science*, 6(3), 2010.

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

[3] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, volume 131 of *LNCS*, pages 52–71. Springer, 1981.

[4] C. Dax and F. Klaedtke. Alternation elimination by complementation (extended abstract). In *LPAR*, volume 5330 of *LNCS*, pages 214–229. Springer, 2008.

[5] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979.

[6] P. Gastin and D. Kuske. Uniform satisfiability in PSPACE for local temporal logics over Mazurkiewicz traces. *Fundam. Inform.*, 80(1-3):169–197, 2007.

[7] P. Gastin and D. Kuske. Uniform satisfiability problem for local temporal logics over Mazurkiewicz traces. *Inf. Comput.*, 208(7):797–816, 2010.

[8] P. Gastin and D. Oddoux. LTL with past and two-way very-weak alternating automata. In *MFCS*, volume 2747 of *LNCS*, pages 439–448. Springer, 2003.

[9] B. Genest, D. Kuske, and A. Muscholl. A Kleene theorem and model checking algorithms for existentially bounded communicating automata. *Inf. Comput.*, 204(6):920–956, 2006.

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

[11] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[12] S. Katz and D. Peled. Interleaving set temporal logic. *Theor. Comput. Sci.*, 75(3):263–287, 1990.

[13] Y. Kesten, A. Pnueli, and L. Raviv. Algorithmic verification of linear temporal logic specifications. In *ICALP*, volume 1443 of *LNCS*, pages 1–16. Springer, 1998.

[14] V. King, O. Kupferman, and M. Y. Vardi. On the complexity of parity word automata. In F. Honsell and M. Miculan, editors, *FoSSaCS*, volume 2030 of *Lecture Notes in Computer Science*, pages 276–286. Springer, 2001.

[15] R. Küsters. Memoryless determinacy of parity games. In *Automata, Logics, and Infinite Games*, volume 2500 of *LNCS*, pages 95–106. Springer, 2001.

[16] P. Madhusudan and B. Meenakshi. Beyond message sequence graphs. In *FSTTCS*, volume 2245 of *LNCS*, pages 256–267. Springer, 2001.

[17] B. Meenakshi and R. Ramanujam. Reasoning about layered message passing systems. *Computer Lang., Systems & Structures*, 30(3-4):171–206, 2004.

[18] R. Mennicke. Propositional dynamic logic with converse and repeat for message-passing systems. In M. Koutny and I. Ulidowski, editors, *CONCUR*, volume 7454 of *LNCS*, pages 531–546. Springer, 2012.

[19] D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *Theor. Comput. Sci.*, 54:267–276, 1987.

[20] D. Peled. Specification and verification of message sequence charts. In *FORTE*, volume 183 of *IFIP Conference Proceedings*, pages 139–154. Kluwer, 2000.
[21] A. Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.
[22] V. R. Pratt. Semantical considerations on Floyd-Hoare logic. In *FOCS*, pages 109–121. IEEE, 1976.
[23] R. S. Streett. Propositional dynamic logic of looping and converse. In *STOC*, pages 375–383. ACM, 1981.
[24] M. Y. Vardi. A temporal fixpoint calculus. In J. Ferrante and P. Mager, editors, *POPL*, pages 250–259. ACM Press, 1988.
[25] M. Y. Vardi. Alternating automata and program verification. In *Computer Science Today*, volume 1000 of *LNCS*, pages 471–485. Springer, 1995.