Communicating Finite-State Machines and Two-Variable Logic

Benedikt Bollig
LSV, CNRS & ENS Paris-Saclay, Université Paris-Saclay, France

Marie Fortin
LSV, CNRS & ENS Paris-Saclay, Université Paris-Saclay, France

Paul Gastin
LSV, CNRS & ENS Paris-Saclay, Université Paris-Saclay, France

Abstract

Communicating finite-state machines are a fundamental, well-studied model of finite-state processes that communicate via unbounded first-in first-out channels. We show that they are expressively equivalent to existential MSO logic with two first-order variables and the order relation.

2012 ACM Subject Classification Theory of computation → Concurrency, Theory of computation → Logic and verification

Keywords and phrases Communicating Finite-state Machines, MSO logic, Message Sequence Charts

Digital Object Identifier 10.4230/LIPIcs.STACS.2018.17

Related Version A full version of the paper is available at `https://arxiv.org/abs/1709.09991`.

1 Introduction

The study of logic-automata connections has ever played a key role in computer science, relating concepts that are a priori very different. Its motivation is at least twofold. First, automata may serve as a tool to decide logical theories. Beginning with the work of Büchi, Elgot, and Trakhtenbrot, who established expressive equivalence of monadic second-order (MSO) logic and finite automata [8, 9, 26], the "automata-theoretic" approach to logic has been successfully applied, for example, to MSO logic on trees [23], temporal logics [27], and first-order logic with two variables over words with an equivalence relation (aka *data words*) [4]. Second, automata serve as models of various kind of state-based systems. Against this background, Büchi-like theorems lay the foundation of *synthesis*, i.e., the process of transforming high-level specifications (represented as logic formulas) into faithful system models. In this paper, we provide a Büchi theorem for *communicating finite-state machines*, which are a classical model of concurrent message-passing systems.

One of the simplest system models are finite automata. They can be considered as single finite-state processes and, therefore, serve as a model of *sequential* systems. Their executions are words, which, seen as a logical structure, consist of a set of positions (also referred to as *events*) that carry letters from a finite alphabet and are *linearly* ordered by some binary relation $\leq$. The simple MSO (even first-order) formula $\forall x.\big(a(x) \implies \exists y.(x \leq y \wedge b(y))\big)$ says that every "request" a is eventually followed by an "acknowledgment" b. In fact, Büchi's theorem allows one to turn any logical MSO specification into a finite automaton. The latter

35th Symposium on Theoretical Aspects of Computer Science (STACS 2018).
Editors: Rolf Niedermeier and Brigitte Vallée; Article No. 17; pp. 17:1–17:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

can then be considered correct by construction. Though the situation quickly becomes more intricate when we turn to other automata models, Büchi theorems have been established for expressive generalizations of finite automata that also constitute natural system models. In the following, we will discuss some of them.

Data automata accept (in the context of system models, we may also say *generate*) words that, in addition to the linear order $\leq$ and its direct-successor relation, are equipped with an equivalence relation $\sim$ [4]. Positions (events) that belong to the same equivalence class may be considered as being executed by one and the same process, while $\leq$ reflects a sort of global control. It is, therefore, convenient to also include a predicate that connects *successive* events in an equivalence class. Bojańczyk et al. showed that data automata are expressively equivalent to existential MSO logic with two first-order variables [4]. A typical formula is $\neg\exists x.\exists y.(x \neq y \wedge x \sim y)$, which says that every equivalence class is a singleton. It should be noted that data automata scan a word twice and, therefore, can hardly be seen as a system model. However, they are expressively equivalent to *class-memory automata*, which distinguish between a global control (modeling, e.g., a shared variable) and a local control for every process [3].

Unlike finite automata and data automata, *asynchronous automata* are a model of *concurrent* shared-memory systems, with a finite number of processes. Their executions are Mazurkiewicz traces, where the relation $\leq$ is no longer a total, but a partial order. Thus, there may be *parallel* events x and y, for which neither $x \leq y$ nor $y \leq x$ holds. A typical logical specification is the mutual exclusion property, which can be expressed in MSO logic as $\neg\exists x.\exists y.(CS(x) \wedge CS(y) \wedge x \parallel y)$ where the parallel operator $x \parallel y$ is defined as $\neg(x \leq y) \wedge \neg(y \leq x)$. Note that this is even a first-order formula that uses only two first-order variables, x and y. It says that there are no two events x and y that access a critical section simultaneously. Asynchronous automata are closed under complementation [29] so that the inductive approach to translating formulas into automata can be applied to obtain a Büchi theorem [24]. Note that complementability is also the key ingredient for MSO characterizations of *nested-word automata* [1] and *branching automata* running over series-parallel posets (aka N-free posets) [18, 2].

The situation is quite different in the realm of *communicating finite-state machines (CFMs)*, aka *communicating automata* or *message-passing automata*, where finitely many processes communicate by exchanging messages through *unbounded* FIFO channels [7]. A CFM accepts/generates message-sequence charts (MSCs) which are also equipped with a partial order $\leq$. Additional binary predicates connect (i) the emission of a message with its reception, and (ii) successive events executed by one and the same process. Unfortunately, CFMs are not closed under complementation [6] so that an inductive translation of MSO logic into automata will fail. In fact, they are strictly less expressive than MSO logic. Two approaches have been adopted to overcome these problems. First, when channels are (existentially or universally) bounded, closure under complementation is recovered so that CFMs are expressively equivalent to MSO logic [17, 19, 11, 12]. Note that, however, the corresponding proofs are much more intricate than in the case of finite automata. Second, CFMs with unbounded channels have been shown to be expressively equivalent to *existential* MSO logic when dropping the order $\leq$ [6]. The proof relies on Hanf's normal form of first-order formulas on structures of *bounded degree* (which is why one has to discard $\leq$) [15]. However, it is clear that many specifications (such as mutual exclusion) are easier to express in terms of $\leq$. But, to the best of our knowledge, a convenient specification language that is exactly as expressive as CFMs has still been missing.

It is the aim of this paper to close this gap, i.e., to provide a logic that

- *matches* exactly the expressive power of *unrestricted* CFMs (in particular, every specification should be *realizable* as an automaton), and
- *includes* the order $\leq$ so that one can easily express natural properties like mutual exclusion.

We show that existential MSO logic with two first-order variables is an appropriate logic. To translate a formula into an automaton, we first follow the approach of [4] for data automata and consider its Scott normal form (cf. [13]). However, while data automata generate total orders, the main difficulty in our proof comes from the fact that $\leq$ is a partial order. Actually, our main technical contribution is a CFM that, running on an MSC, marks precisely those events that are in parallel to some event of a certain type.

It should be noted that message-passing automata can also be used as acceptors of the underlying (graph) architecture. In that case, logical characterizations have been obtained in terms of MSO and modal logics [20, 16, 21, 22]. However, in our framework, the architecture is fixed and we rather reason about the set of *executions* of a CFM.

The paper is structured as follows. In Section 2, we recall the classical notions of CFMs and MSO logic. Section 3 states our main result, describes our proof strategy, and settles several preliminary lemmas. The main technical part is contained in Section 4. We conclude in Section 5. Missing proofs can be found at: `https://arxiv.org/abs/1709.09991`.

2 Preliminaries

Let Σ be a finite alphabet. The set of finite words over Σ is denoted by Σ^*, which includes the empty word ε. For $w \in \Sigma^*$, let $|w|$ denote its length. In particular, $|\varepsilon| = 0$. The inverse of a binary relation R is defined as $R^{-1} = \{(f, e) \mid (e, f) \in R\}$. We denote the size of a finite set A by $|A|$. The disjoint union of sets A and B is denoted by $A \uplus B$.

2.1 Communicating Finite-State Machines

Communicating finite-state machines are a natural model of communicating systems where a finite number of processes communicate through a priori unbounded FIFO channels [7]. Every process is represented as a *finite transition system* (S, ι, Δ) over some finite alphabet Γ, i.e., S is a finite set of states with initial state $\iota \in S$, and $\Delta \subseteq S \times \Gamma \times S$ is the transition relation. An element from Γ will describe the action that is performed when taking a transition (e.g., "send a message to some process" or "perform a local computation").

A communicating finite-state machine is a *collection* of finite transition systems, one for each process. We assume a finite set $P = \{p, q, r, \ldots\}$ of *processes* and a finite alphabet $\Sigma = \{a, b, c, \ldots\}$ of *labels*. We suppose that there is a channel between any two distinct processes. Thus, the set of *channels* is $Ch = \{(p, q) \in P \times P \mid p \neq q\}$.

▶ **Definition 1.** A *communicating finite-state machine (CFM)* over P and Σ is a tuple $\mathcal{A} = ((\mathcal{A}_p)_{p \in P}, Msg, Acc)$ where

- Msg is a finite set of *messages*,
- $\mathcal{A}_p = (S_p, \iota_p, \Delta_p)$ is a finite transition system over $\Sigma \cup (\Sigma \times \{!, ?\} \times Msg \times (P \setminus \{p\}))$, and
- $Acc \subseteq \prod_{p \in P} S_p$ is the set of *global accepting states*.[1]

[1] We may also include several *global* initial states without changing the expressive power, which is convenient in several of the forthcoming constructions.

Let $t = (s, \alpha, s') \in \Delta_p$ be a transition of process p. We call s the *source state* of t, denoted by $source(t)$, and s' its *target state*, denoted $target(t)$. Moreover, α is the *action* executed by t. If $\alpha \in \Sigma$, then t is said to be *internal*, and we let $label(t) = \alpha$. The label from Σ may provide some more information about an event (such as "enter critical section"). When α is of the form $(a, !, \mathrm{m}, q)$, then t is a send transition, which writes message m into the channel (p, q). Accordingly, we let $msg(t) = \mathrm{m}$, $receiver(t) = q$, and $label(t) = a$. Finally, performing $\alpha = (a, ?, \mathrm{m}, q)$ removes message m from channel (q, p). In that case, we set $msg(t) = \mathrm{m}$, $sender(t) = q$, and $label(t) = a$.

If there is only one process, i.e., P is a singleton, then all transitions are internal so that a CFM is simply a finite automaton accepting a regular set of words over the alphabet Σ. In the presence of several processes, a single behavior is a *tuple* of words over Σ, one for every process. However, these words are not completely independent (unless all transitions are internal and there is no communication), since the sending of a message can be linked to its reception. This is naturally reflected by a binary relation $\lhd$ that connects word positions on distinct processes. The resulting structure is called a message sequence chart.

▶ **Definition 2.** A *message sequence chart (MSC)* over P and Σ is a tuple $M = ((w_p)_{p\in P}, \lhd)$ where $w_p \in \Sigma^*$ for every $p \in P$. We require that at least one of these words be non-empty. By $E_p = \{p\} \times \{1, \dots, |w_p|\}$, we denote the set of *events* that are executed by process p. Accordingly, the (disjoint) union $E = \bigcup_{p\in P} E_p$ is the set of all events. Implicitly, we obtain the labeling $\lambda : E \to (P \times \Sigma)$ defined by $\lambda((p, i)) = (p, a)$ where a is the i-th letter of w_p, and the process-edge relation $\to \subseteq \bigcup_{p\in P}(E_p \times E_p)$, which connects successive events that are executed by one and the same process: $(p, i) \to (p, i+1)$ for all $p \in P$ and $i \in \{1, \dots, |w_p|-1\}$. Now, $\lhd \subseteq \bigcup_{(p,q)\in Ch}(E_p \times E_q)$ is a set of *message edges*, satisfying the following:

- $(\to \cup \lhd)$ is acyclic (intuitively, messages cannot travel backwards in time), and the associated partial order is denoted $\leq = (\to \cup \lhd)^*$ with strict part $< = (\to \cup \lhd)^+$,
- each event is part of at most one message edge, and
- for all $(p, q) \in Ch$ and $(e, f), (e', f') \in \lhd \cap (E_p \times E_q)$, we have $e \to^* e'$ iff $f \to^* f'$ (which guarantees a FIFO behavior).

An event that does not belong to a message edge is called *internal*. We say that two events $e, f \in E$ are *parallel*, written $e \parallel f$, if neither $e \leq f$ nor $f \leq e$. The set of all MSCs is denoted $\mathbb{MSC}(P, \Sigma)$.

▶ **Example 3.** An example MSC over $P = \{p, q, r\}$ and $\Sigma = \{a, b, c\}$ is depicted in Figure 1. For the moment, we ignore the colored areas and annotations. We have $w_p = aacaaaaa$, $w_r = aaaaaaaaaa$, and $w_q = abbaacaaa$ (note that q is the bottom process). Consider the events $f = (p, 4)$, $e = (p, 5)$, and $g = (q, 2)$. That is, $f \to e$ and $g \lhd e$. Moreover, $(p, 3) \parallel (q, 6)$ (i.e., the two c-labeled events are parallel), while $(p, 3) \leq (q, 8)$.

Let $M = ((w_p)_{p\in P}, \lhd)$ be an MSC over P and Σ. A *run* of the CFM $\mathcal{A}$ on M is given by a mapping ρ that associates with every event $e \in E_p$ ($p \in P$) the transition $\rho(e) \in \Delta_p$ that is executed at e. We require that

1. for every $e \in E$ with $\lambda(e) = (p, a)$, we have $label(\rho(e)) = a$,
2. for every process $p \in P$ such that $E_p \neq \emptyset$, we have $source(\rho((p, 1))) = \iota_p$,
3. for every process edge $(e, f) \in \to$, we have $target(\rho(e)) = source(\rho(f))$,
4. for every internal event $e \in E$, $\rho(e)$ is an internal transition, and
5. for every message edge $(e, f) \in \lhd$ with $e \in E_p$ and $f \in E_q$, $\rho(e) \in \Delta_p$ is a send transition and $\rho(f) \in \Delta_q$ is a receive transition such that $msg(\rho(e)) = msg(\rho(f))$, $receiver(\rho(e)) = q$, and $sender(\rho(f)) = p$.

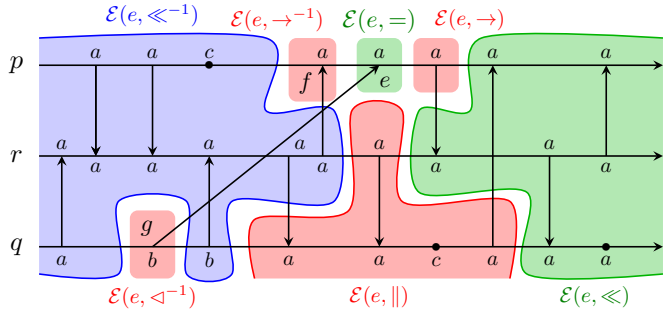

Figure 1 An MSC (cf. Example 3) and the partition determined by an event e (cf. Example 10).

Note that, when $|P| = 1$, Condition 5. becomes meaningless and Conditions 1.–4. emulate the behavior of a finite automaton.

It remains to define when ρ is accepting. To this aim, we collect the final states of each process p. If $E_p \neq \emptyset$, then let s_p be the target state of $\rho((p, |w_p|))$, i.e., of the last transition taken by p. Otherwise, let $s_p = \iota_p$. Now, we say that ρ is *accepting* if $(s_p)_{p \in P} \in Acc$.

Finally, the *language* of $\mathcal{A}$ is defined as $L(\mathcal{A}) = \{M \in \mathbb{MSC}(P, \Sigma) \mid \text{there is an accepting run of } \mathcal{A} \text{ on } M\}$.

2.2 MSO and Two-Variable Logic

While CFMs serve as an operational model of concurrent systems, MSO logic can be considered as a high-level specification language. It uses first-order variables $x, y, \ldots$ to quantify over events, and second-order variables $X, Y, \ldots$ to represent sets of events. The logic MSO (we assume that P and Σ are understood from the context) is defined by the following grammar:

$$\varphi ::= p(x) \mid a(x) \mid x \in X \mid x = y \mid x \to y \mid x \lhd y \mid x \leq y \mid \varphi \vee \varphi \mid \neg\varphi \mid \exists x.\varphi \mid \exists X.\varphi$$

where x and y are first-order variables, X is a second-order variable, $p \in P$, and $a \in \Sigma$. We use the usual operator precedence. For convenience, we allow usual abbreviations such as conjunction $\varphi \wedge \psi$, universal quantification $\forall x.\varphi$, implication $\varphi \implies \psi$, etc. The atomic formulas $p(x)$ and $a(x)$ are interpreted as "x is located on process p" and, respectively, "the label of event x is a". The binary predicates are self-explanatory, and the boolean connectives and quantification are interpreted as usual. The size $|\varphi|$ of a formula $\varphi \in$ MSO is the length of φ seen as a string.

A variable that occurs free in a formula requires an interpretation in terms of an event/a set of events from the given MSC. We will write, for example, $M, x \mapsto e, y \mapsto f \models \varphi$ if M satisfies φ provided x is interpreted as e and y as f. If φ is a *sentence* (i.e., does not contain any free variable), then we write $M \models \varphi$ to denote that M satisfies φ. With a sentence φ, we associate the MSC language $L(\varphi) = \{M \in \mathbb{MSC}(P, \Sigma) \mid M \models \varphi\}$.

The set FO of *first-order formulas* is the fragment of MSO that does not make use of second-order quantification $\exists X$. The two-variable fragment of FO, denoted by FO^2, allows only for two first-order variables, x and y (which, however, can be quantified and reused arbitrarily often). Moreover, formulas from EMSO, the *existential fragment* of MSO, are of the form $\exists X_1 \ldots \exists X_n.\varphi$ where $\varphi \in$ FO. Accordingly, EMSO^2 is the set of EMSO formulas whose first-order kernel is in FO^2.

The expressive power of all these fragments heavily depends on the set of binary predicates among $\{\rightarrow, \lhd, \leq\}$ that are actually allowed. For a logic $\mathcal{C} \in \{\mathrm{MSO}, \mathrm{EMSO}, \mathrm{EMSO}^2, \mathrm{FO}, \mathrm{FO}^2\}$ and a set $R \subseteq \{\rightarrow, \lhd, \leq\}$, let $\mathcal{C}[R]$ be the logic $\mathcal{C}$ restricted to the binary predicates from R (however, we always allow for equality, i.e., formulas of the form $x = y$). In particular, $\mathrm{MSO} = \mathrm{MSO}[\rightarrow, \lhd, \leq]$. As the transitive closure of a binary relation is definable in terms of second-order quantification, $\mathrm{MSO}[\rightarrow, \lhd, \leq]$ and $\mathrm{MSO}[\rightarrow, \lhd]$ have the same expressive power (over MSCs). On the other hand, $\mathrm{MSO}[\leq]$ is strictly less expressive [6].

▶ **Example 4.** Suppose $P = \{p, q, r\}$ and $\Sigma = \{a, b, c\}$. The (mutual exclusion) formula $\neg\exists x.\exists y.(c(x) \wedge c(y) \wedge x \parallel y)$, where $x \parallel y$ is defined as $\neg(x \leq y) \wedge \neg(y \leq x)$, is in $\mathrm{FO}^2[\leq]$. It is not satisfied by the MSC from Figure 1, as the two c-labeled internal events are parallel.

Let us turn to the relative expressive power of CFMs and logic. We say that CFMs and a logic $\mathcal{C}$ are *expressively equivalent* if,

- for every CFM $\mathcal{A}$, there exists a sentence $\varphi \in \mathcal{C}$ such that $L(\mathcal{A}) = L(\varphi)$, and
- for every sentence $\varphi \in \mathcal{C}$, there exists a CFM $\mathcal{A}$ such that $L(\mathcal{A}) = L(\varphi)$.

Now, the Büchi-Elgot-Trakhtenbrot theorem can be stated as follows:

▶ **Theorem 5** ([8, 9, 26])**.** *If $|P| = 1$, then CFMs (i.e., finite automata) and* MSO *are expressively equivalent.*

Unfortunately, when several processes are involved, MSO is too expressive to be captured by CFMs, unless one restricts the logic:

▶ **Theorem 6** ([6])**.** *CFMs and* $\mathrm{EMSO}[\rightarrow, \lhd]$ *are expressively equivalent.*

3 Two-Variable Logic and CFMs

The logic $\mathrm{EMSO}[\rightarrow, \lhd]$ is not very convenient as a specification language, as it does not allows us to talk, explicitly, about the order of an MSC. It should be noted that CFMs and MSO are expressively equivalent if one restricts to MSCs that are *channel-bounded* [17, 11, 19]. Our main result allows one to include $\leq$ in the unbounded case, too, though we have to restrict to two first-order variables:

▶ **Theorem 7.** *CFMs and* $\mathrm{EMSO}^2[\rightarrow, \lhd, \leq]$ *are expressively equivalent. More precisely:*

1. *Given a CFM $\mathcal{A}$, we can effectively construct a sentence $\varphi_{\mathcal{A}} \in \mathrm{EMSO}^2[\rightarrow, \lhd, \leq]$ of polynomial size such that $L(\mathcal{A}) = L(\varphi_{\mathcal{A}})$.*
2. *Given a sentence $\varphi \in \mathrm{EMSO}^2[\rightarrow, \lhd, \leq]$, we can effectively construct a CFM $\mathcal{A}_\varphi$ with $2^{2^{\mathcal{O}(|\varphi|+|P|\log|P|)}}$ states (per process) such that $L(\mathcal{A}_\varphi) = L(\varphi)$.*

Translating a CFM into an EMSO^2 formula is standard: Second-order variables represent an assignment of transitions to events. The first-order kernel then checks whether this guess is consistent with the definition of an accepting run.

Before dwelling on the involved proof of the converse translation, i.e., Theorem 7(2), we start with some comments. The CFM $\mathcal{A}_\varphi$ is inherently *nondeterministic* (for the definition of a deterministic CFM, cf. [12]). Already for FO^2, this is unavoidable: CFMs are in general not determinizable, as witnessed by an FO^2-definable language in [12, Proposition 5.1]. Note that the number of states of $\mathcal{A}_\varphi$ is, in fact, independent of the number of letters from Σ that do *not* occur in the formula. This is why Theorem 7(2) mentions only $|\varphi|$ rather than $|\Sigma|$. Actually, the doubly exponential size of $\mathcal{A}_\varphi$ is necessary, even for $\mathrm{FO}^2[\rightarrow]$ or $\mathrm{FO}^2[\leq]$ sentences and a small number of processes. The following can be shown using known techniques [14, 28]:

▶ **Theorem 8.** *(i) Assume* $|P| = 1$ *and* $|\Sigma| = 2$*. For all* $n \in \mathbb{N}$*, there is a sentence* $\varphi \in \mathrm{FO}^2[\rightarrow]$ *of size* $\mathcal{O}(n^2)$ *such that no CFM with less than* 2^{2^n} *states recognizes* $L(\varphi)$*. (ii) Assume* $|P| = 2$ *and* $|\Sigma| = n$ *with* $n \geq 2$*. There is a sentence* $\varphi \in \mathrm{FO}^2[\leq]$ *of size* $\mathcal{O}(n)$ *such that no CFM with less than* $2^{2^{n-1}}$ *states on every process recognizes* $L(\varphi)$*.*

The rest of this paper is devoted to the proof of Theorem 7(2). In a first step, we translate the given formula into Scott normal form (cf. [13]):

▶ **Lemma 9** (Scott Normal Form)**.** *Every sentence from* $\mathrm{EMSO}^2[\rightarrow, \lhd, \leq]$ *is effectively equivalent to a linear-size sentence of the form* $\exists X_1 \ldots \exists X_m.\psi$ *where* $\psi = \forall x.\forall y.\varphi \wedge \bigwedge_{i=1}^{\ell} \forall x.\exists y.\varphi_i \in \mathrm{FO}^2[\rightarrow, \lhd, \leq]$ *with* $\varphi, \varphi_1, \ldots, \varphi_\ell$ *quantifier-free.*

As the class of languages accepted by CFMs is closed under projection, it remains to deal with the first-order part ψ. Note that ψ contains free occurrences of second-order variables $X_1, \ldots, X_m$. To account for an interpretation of these variables, we extend the alphabet Σ towards the alphabet $\Sigma' = \Sigma \times \{0,1\}^m$ of exponential size. When an event e is labeled with $(a, b_1, \ldots, b_m) \in \Sigma'$, we consider that $e \in X_i$ iff $b_i = 1$.

As the class of languages accepted by CFMs is closed under intersection, too, the proof of Theorem 7(2) comes down to the translation of the formulas of the form $\forall x.\forall y.\eta$ or $\forall x.\exists y.\eta$ where η is a quantifier-free formula with free variables among $X_1, \ldots, X_m, x, y$. Notice that, given an MSC $M \in \mathbb{MSC}(P, \Sigma')$ and events e and f in M, whether $M, x \mapsto e, y \mapsto f \models \eta$ holds or not only depends on the labels of e and f, and their relative position. This is formalized below in terms of *types*.

Types. Let $M = ((w_p)_{p \in P}, \lhd) \in \mathbb{MSC}(P, \Sigma')$ be an MSC. Towards the definition of the type of an event, we define another binary relation $\ll \;=\; < \setminus (\rightarrow \cup \lhd)$. Let Ω be the set of *relation symbols* $\{=, \rightarrow, \lhd, \parallel, \rightarrow^{-1}, \lhd^{-1}, \ll, \ll^{-1}\}$. Given an event $e \in E$ and a relation symbol $\bowtie \in \Omega$, we let $\mathcal{E}(e, \bowtie) = \{f \in E \mid e \bowtie f\}$. In particular, $\mathcal{E}(e, \ll^{-1}) = \{f \in E \mid f < e \wedge \neg(f \rightarrow e) \wedge \neg(f \lhd e)\}$. Notice that all these sets form a partition of E, i.e., $E = \biguplus_{\bowtie \in \Omega} \mathcal{E}(e, \bowtie)$ (some sets may be empty, though). The $\bowtie$-*type* and the *type* of an event $e \in E$ are respectively defined by

$$\mathsf{type}^{\bowtie}_M(e) = \{\lambda(f) \mid f \in \mathcal{E}(e, \bowtie)\} \subseteq P \times \Sigma' \quad \text{and} \quad \mathsf{type}_M(e) = \big(\mathsf{type}^{\bowtie}_M(e)\big)_{\bowtie \in \Omega}.$$

By $\mathbb{T}_{P,\Sigma'} = \prod_{\bowtie \in \Omega} 2^{P \times \Sigma'}$, we denote the (finite) set of possible types. Thus, we actually deal with functions $\mathsf{type}^{\bowtie}_M : E \to 2^{P \times \Sigma'}$ and $\mathsf{type}_M : E \to \mathbb{T}_{P,\Sigma'}$.

▶ **Example 10.** Consider Figure 1 and the distinguished event e. Suppose $a, b, c \in \Sigma'$. The sets $\mathcal{E}(e, \bowtie)$, which form a partition of the set of events, are indicated by the colored areas. Note that, since e is a receive event, $\mathcal{E}(e, \lhd) = \emptyset$. Moreover, $\mathsf{type}^{\rightarrow}_M(e) = \mathsf{type}^{=}_M(e) = \mathsf{type}^{\rightarrow^{-1}}_M(e) = \{(p, a)\}$ and $\mathsf{type}^{\ll^{-1}}_M(e) = \{(p, a), (p, c), (r, a), (q, a), (q, b)\}$.

In fact, it is enough to know the type of every event to (effectively) evaluate ψ. To formalize this, let η be a quantifier-free formula with free variables among $X_1, \ldots, X_m, x, y$. Assume that we are given $M \in \mathbb{MSC}(P, \Sigma')$ and two events e and f that are labeled with $(p, \sigma), (p', \sigma') \in P \times \Sigma'$, respectively, where $\sigma = (a, b_1, \ldots, b_m)$ and $\sigma' = (a', b'_1, \ldots, b'_m)$. Let $\bowtie \in \Omega$ be the unique relation such that $e \bowtie f$. To decide whether $M, x \mapsto e, y \mapsto f \models \eta$, we rewrite η into a propositional formula $[\![\eta]\!]^{\bowtie}_{(p,\sigma),(p',\sigma')}$ that can be evaluated to *true* or *false*: Replace the formulas $p(x)$, $a(x)$, $p'(y)$, $a'(y)$, $x \in X_i$ with $b_i = 1$, and $y \in X_i$ with $b'_i = 1$ by *true*. All other unary predicates become *false* (we consider $z \in X_i$ to be unary). Formulas $z \sqsubset z'$ with $z, z' \in \{x, y\}$ and $\sqsubset \in \{=, \rightarrow, \lhd, \leq\}$ can be evaluated to *true* or *false* based on the assumption that $x \bowtie y$. By an easy induction, we obtain:

▶ **Lemma 11.** *For all $\eta \in \{\varphi, \varphi_1, \ldots, \varphi_\ell\}$, $M \in \mathbb{MSC}(P, \Sigma')$, and events e of M:*

- $M, x \mapsto e \models \exists y.\eta$ *iff* $[\![\eta]\!]^{\bowtie}_{\lambda(e),(p',\sigma')}$ *is true for some* $\bowtie \in \Omega$ *and* $(p', \sigma') \in \mathsf{type}^{\bowtie}_M(e)$.
- $M, x \mapsto e \models \forall y.\eta$ *iff* $[\![\eta]\!]^{\bowtie}_{\lambda(e),(p',\sigma')}$ *is true for all* $\bowtie \in \Omega$ *and* $(p', \sigma') \in \mathsf{type}^{\bowtie}_M(e)$.

Therefore, we start by constructing a CFM $\mathcal{A}_{\mathsf{types}}$ that "labels" each event with its type. Formally, $\mathcal{A}_{\mathsf{types}}$ runs on *extended* MSCs, whose events have additional labels from the finite alphabet $\mathbb{T}_{P,\Sigma'}$. More generally, given a finite alphabet Γ, it will be convenient to consider a Γ-extended MSC from $\mathbb{MSC}(P, \Sigma' \times \Gamma)$, in the obvious way, as a pair (M, γ) where $M \in \mathbb{MSC}(P, \Sigma')$ and γ is a function from the set of events E to Γ.

▶ **Theorem 12.** *There is a CFM $\mathcal{A}_{\mathsf{types}}$ over P and $\Sigma' \times \mathbb{T}_{P,\Sigma'}$ with $2^{|\Sigma'| \cdot 2^{\mathcal{O}(|P| \log |P|)}}$ states such that $L(\mathcal{A}_{\mathsf{types}}) = \{(M, \mathsf{type}_M) \mid M \in \mathbb{MSC}(P, \Sigma')\}$.*

The proof of Theorem 12 is given in Section 4. Before this, we show that Theorem 12 (together with Lemmas 9 and 11) implies Theorem 7(2).

Proof of Theorem 7(2). We first convert the given sentence $\xi \in \mathrm{EMSO}^2[\rightarrow, \lhd, \leq]$ into the linear-size Scott normal form $\exists X_1 \ldots \exists X_m.\psi$ where $\psi = \forall x.\forall y.\varphi \wedge \bigwedge_{i=1}^{\ell} \forall x.\exists y.\varphi_i$ (Lemma 9). According to Lemma 11, the CFM for ψ is obtained from $\mathcal{A}_{\mathsf{types}}$ by restricting the transition relation: We keep a transition of process p with label $(\sigma, (\tau_{\bowtie})_{\bowtie \in \Omega}) \in \Sigma' \times \mathbb{T}_{P,\Sigma'}$ if $[\![\varphi]\!]^{\bowtie}_{(p,\sigma),(p',\sigma')}$ is *true* for all $\bowtie \in \Omega$ and $(p', \sigma') \in \tau_{\bowtie}$, and for all $1 \leq i \leq n$, $[\![\varphi_i]\!]^{\bowtie}_{(p,\sigma),(p',\sigma')}$ is *true* for some $\bowtie \in \Omega$ and $(p', \sigma') \in \tau_{\bowtie}$. Moreover, the new transition label will just be σ (the type is projected away). Finally, we project the extended alphabet $\Sigma' = \Sigma \times \{0,1\}^m$ to Σ. The resulting CFM $\mathcal{A}_\xi$ is equivalent to the given sentence ξ. ◀

4 CFM $\mathcal{A}_{\mathsf{types}}$ Checking the Type of Events

We obtain $\mathcal{A}_{\mathsf{types}}$ as the product of CFMs $\mathcal{A}^{\bowtie}$ over P and $\Sigma' \times 2^{P \times \Sigma'}$ such that $L(\mathcal{A}^{\bowtie}) = \{(M, \mathsf{type}^{\bowtie}_M) \mid M \in \mathbb{MSC}(P, \Sigma')\}$. Thus, it only remains to construct $\mathcal{A}^{\bowtie}$, for all $\bowtie \in \Omega$. The cases $\bowtie \in \{=, \rightarrow, \lhd, \rightarrow^{-1}, \lhd^{-1}\}$ are straightforward.

▶ **Lemma 13.** *There is a CFM $\mathcal{A}^{\ll^{-1}}$ over P and $\Sigma' \times 2^{P \times \Sigma'}$ with $2^{\mathcal{O}(|P \times \Sigma'|)}$ states such that $L(\mathcal{A}^{\ll^{-1}}) = \{(M, \mathsf{type}^{\ll^{-1}}_M) \mid M \in \mathbb{MSC}(P, \Sigma')\}$.*

Proof sketch. Consider Figure 1 and suppose $a, b, c \in \Sigma'$. At the time of reading event e, the CFM $\mathcal{A}^{\ll^{-1}}$ should deduce $\mathsf{type}^{\ll^{-1}}_M(e) = \{(p,a), (p,c), (r,a), (q,a), (q,b)\}$. To do so, it collects all labelings from $P \times \Sigma'$ that it has seen in the past, i.e., $\mathsf{type}^{\ll^{-1}}_M(e) \cup \mathsf{type}^{\rightarrow^{-1}}_M(e) \cup \mathsf{type}^{\lhd^{-1}}_M(e)$. Naively, one would then just remove the labels (p,a) and (q,b) of the predecessors f and g of e. However, this leads to the wrong result, since both (p,a) and (q,b) are contained in $\mathsf{type}^{\ll^{-1}}_M(e)$. In particular, there is another (q,b)-labeled event in $\mathcal{E}(e, \ll^{-1})$. The solution is to count the number of occurrences of each label up to 2. When reading e, the CFM will have seen (p,a) and (q,b) at least twice so that it can safely conclude that both are contained in $\mathsf{type}^{\ll^{-1}}_M(e)$. ◀

We then obtain $\mathcal{A}^{\ll}$ by symmetry. To complete the proof of Theorem 12, we construct below the CFM $\mathcal{A}^{\|}$ such that $L(\mathcal{A}^{\|}) = \{(M, \mathsf{type}^{\|}_M) \mid M \in \mathbb{MSC}(P, \Sigma')\}$.

For all $p, q \in P$ with $p \neq q$ and $a \in \Sigma'$, we define

$$L^0_{p,q,a} = \left\{(M,\gamma) \in \mathbb{MSC}(P, \Sigma' \times \{0,1\}) \mid \forall e \in E_p : \big(\gamma(e) = 0 \implies (q,a) \notin \mathsf{type}^{\|}_M(e)\big)\right\},$$
$$L^1_{p,q,a} = \left\{(M,\gamma) \in \mathbb{MSC}(P, \Sigma' \times \{0,1\}) \mid \forall e \in E_p : \big(\gamma(e) = 1 \implies (q,a) \in \mathsf{type}^{\|}_M(e)\big)\right\}.$$

▶ **Lemma 14.** *For all $p, q \in P$ with $p \neq q$ and $a \in \Sigma'$, there are CFMs $\mathcal{A}^0_{p,q,a}$ and $\mathcal{A}^1_{p,q,a}$ over P and $\Sigma' \times \{0,1\}$ with $2^{2^{\mathcal{O}(|P| \log |P|)}}$ states such that $L(\mathcal{A}^0_{p,q,a}) = L^0_{p,q,a}$ and $L(\mathcal{A}^1_{p,q,a}) = L^1_{p,q,a}$.*

Before proving Lemma 14, we explain how to derive the CFM $\mathcal{A}^{\|}$. Consider an extended MSC $(M, \tau) \in \mathbb{MSC}(P, \Sigma' \times \mathbb{T}_{P,\Sigma'})$. For all $p, q \in P$ with $p \neq q$ and $a \in \Sigma'$, define $\gamma^{M,\tau}_{p,q,a} \colon E \to \{0,1\}$ by $\gamma^{M,\tau}_{p,q,a}(e) = 1$ iff $e \in E_p$ and $(q,a) \in \tau_{\|}$ assuming $\tau(e) = (\tau_{\bowtie})_{\bowtie \in \Omega}$. From the CFMs $\mathcal{A}^0_{p,q,a}$ and $\mathcal{A}^1_{p,q,a}$, we easily derive a CFM $\mathcal{A}_{p,q,a}$ over P and $\Sigma' \times \mathbb{T}_{P,\Sigma'}$ such that $(M, \tau) \in L(\mathcal{A}_{p,q,a})$ iff $(M, \gamma^{M,\tau}_{p,q,a}) \in L^0_{p,q,a} \cap L^1_{p,q,a}$. We obtain $\mathcal{A}^{\|}$ as the intersection of the CFMs $\mathcal{A}_{p,q,a}$.

Construction of $\mathcal{A}^0_{p,q,a}$. Let $(M, \gamma) \in \mathbb{MSC}(P, \Sigma' \times \{0,1\})$ be an MSC. We can easily check that $(M, \gamma) \in L^0_{p,q,a}$ iff there is a path ν in M (i.e., a path in the directed graph $(E, \to \cup \lhd)$) such that all events e on process p with $\gamma(e) = 0$ and all events f such that $\lambda(f) = (q,a)$ are on ν. Therefore, the CFM $\mathcal{A}^0_{p,q,a}$ will try to guess such a path ν. This path is represented by a token moved along the MSC. Initially, exactly one process has the token. At each event, the automaton may choose to pass along the token to the next event of the current process, or (if the event is a write) to send the token to another process. Formally, (non)-possession of the token is represented by two states, s_{token} and $s_{\overline{\mathsf{token}}}$, and movements of the token from one process to another by messages. All global states are accepting. Notice that $\mathcal{A}^0_{p,q,a}$ has only two states per process. Process p may read an event labeled 0 only if it has the token, and process q may read a's only if it has the token. Clearly, $\mathcal{A}^0_{p,q,a}$ has an accepting run on (M, γ) iff $(M, \gamma) \in L^0_{p,q,a}$.

Construction of $\mathcal{A}^1_{p,q,a}$. Let $M = ((w_p)_{p \in P}, \lhd)$ be an MSC. For $e \in E$ and $F \subseteq E$, let $\|_p(e) = \{f \in E_p \mid f \parallel e\}$ and $\|_p(F) = \{e \in E_p \mid e \parallel f \text{ for some } f \in F\}$. Moreover, given $e \in E$, define $\downarrow_p(e) = \{f \in E_p \mid f < e\}$ and $\uparrow_p(e) = \{f \in E_p \mid e < f\}$. An *interval* in M is a (possibly empty) finite set of events $\{e_1, \dots, e_k\}$ such that $e_1 \to \cdots \to e_k$. For all $e, f \in E_p$, we denote by $[e, f]$ the interval $\{g \in E_p \mid e \leq g \leq f\}$.

▶ Remark. For all $e \in E_q$ (recall that $p \neq q$), the sets $\downarrow_p(e)$, $\|_p(e)$, and $\uparrow_p(e)$ are intervals (possibly empty) of events on process p, such that $E_p = \downarrow_p(e) \uplus \|_p(e) \uplus \uparrow_p(e)$.

The idea is that $\mathcal{A}^1_{p,q,a}$ will guess a set of intervals covering all 1-labeled events on process p, and check that, for each interval I, there exists an event f such that $\lambda(f) = (q,a)$ and $I = \|_p(f)$. We first show that it will be sufficient for $\mathcal{A}^1_{p,q,a}$ to guess two sequences of disjoint intervals:

▶ **Lemma 15.** *Let $M = ((w_p)_{p \in P}, \lhd) \in \mathbb{MSC}(P, \Sigma')$ and $F = \{f \in E \mid \lambda(f) = (q,a)\}$. There exist subsets $F_1, F_2 \subseteq F$ such that the following hold:*

- $\|_p(F_1) \cup \|_p(F_2) = \|_p(F)$.
- *For $i \in \{1,2\}$, the intervals in $\|_p(F_i)$ are pairwise disjoint, and not adjacent: if $f, f' \in F_i$ and $f \neq f'$, then $\|_p(f) \cup \|_p(f')$ is not an interval.*

Proof. We first construct a set $F' \subseteq F$ by iteratively removing redundant events from F, i.e., until there remains no event f such that $\|_p(f) \subseteq \|_p(F' \setminus \{f\})$. Now, consider three events $f, f', f'' \in F'$ such that $f < f' < f''$. If $\|_p(f) \cup \|_p(f'')$ is an interval, we have $\|_p(f') \subseteq \|_p(f) \cup \|_p(f'')$, a contradiction with $f, f', f'' \in F'$.

Therefore, for each event $f \in F'$, there is at most one event $f' \in F'$ such that $f < f'$ and $\|_p(f) \cup \|_p(f')$ is an interval. We deduce that the set F' can be divided into two sets F_1 and F_2 satisfying the requirements of the lemma. ◀

So, $\mathcal{A}^1_{p,q,a}$ will proceed as follows. It will guess the sets F_1, F_2, $\|_p(F_1)$ and $\|_p(F_2)$, that is, label some events on process q with "F_1" or "F_2", and some events on process p with "F_1" and/or "F_2". This labeling must be such that on process q, only events initially labeled a may be labeled "F_1" or "F_2" (the sets guessed for F_1 and F_2 contain only events labeled a), and that on process p, all events initially labeled 1 must be labeled either "F_1", "F_2", or both (the sets guessed for $\|_p(F_1)$ and $\|_p(F_2)$ cover all events labeled 1 on process p). Then, for each $i \in \{1,2\}$, $\mathcal{A}^1_{p,q,a}$ will check condition C_i: for each non-empty maximal interval I of events marked F_i on process p, there exists an event f marked F_i on process q such that $I = \|_p(f)$. The CFM $\mathcal{A}^1_{p,q,a}$ will check C_1 and C_2 with two copies of a CFM $\mathcal{A}_{\mathsf{parallel}}$ defined below. But first, notice that if $\mathcal{A}^1_{p,q,a}$ has an accepting run on M then $M \in L^1_{p,q,a}$. Conversely, if $M \in L^1_{p,q,a}$, then if $\mathcal{A}^1_{p,q,a}$ guesses correctly the sets F_1, F_2, $\|_p(F_1)$ and $\|_p(F_2)$, it accepts.

The MSC language L_{parallel}. Define the language L_{parallel} of extended MSCs $(M,\zeta) \in \mathbb{MSC}(P, \Sigma' \times \{0,1\})$ with $\zeta\colon E \to \{0,1\}$ such that

- for each non-empty maximal interval I of 1-labeled events on process p, there exists exactly one 1-labeled event f on process q such that $\|_p(f) = I$,
- and conversely, for all 1-labeled events f on process q, there exists a non-empty maximal interval I of 1-labeled events on process p such that $\|_p(f) = I$.

We show below how to construct a CFM $\mathcal{A}_{\mathsf{parallel}}$ for the language L_{parallel}. Notice that $\mathcal{A}^1_{p,q,a}$ can check condition C_i ($i \in \{1,2\}$) by running $\mathcal{A}_{\mathsf{parallel}}$ on the MSC (M,ζ) where the events labeled 1 by ζ are those labeled F_i by $\mathcal{A}^1_{p,q,a}$.

We can decompose this problem one last time. Let Π (respectively, $\Pi_{p,q}$) be the set of process sequences $\pi = p_1 \dots p_n$ (respectively, with $p_1 = p$ and $p_n = q$) such that $n \geq 1$ and $p_i \neq p_j$ for $i \neq j$. For $\pi = p_1 \dots p_n \in \Pi$ and $e, f \in E$, we write $e \rightsquigarrow_\pi f$ if there exist events $e = e_1, f_1, e_2, f_2, \dots, e_n, f_n = f$ such that, for all i, we have $e_i, f_i \in E_{p_i}$, $e_i \to^* f_i$, and $f_i \lhd e_{i+1}$. For all events $e \in E$ such that $\{f \in E \mid f \rightsquigarrow_\pi e\}$ (respectively, $\{f \in E \mid e \rightsquigarrow_\pi f\}$) is non-empty, we let

$$\mathsf{pred}_\pi(e) = \max\{f \in E \mid f \rightsquigarrow_\pi e\} \quad \text{and} \quad \mathsf{succ}_\pi(e) = \min\{f \in E \mid e \rightsquigarrow_\pi f\}\,.$$

This is well-defined since all events in $\{f \in E \mid f \rightsquigarrow_\pi e\}$ (respectively, $\{f \in E \mid e \rightsquigarrow_\pi f\}$) are on the same process, hence are ordered. Note that, if $\pi = p$ consists of a single process, then, for all $e \in E_p$, we have $\mathsf{pred}_\pi(e) = e = \mathsf{succ}_\pi(e)$. Moreover, notice that $\leq = \bigcup_{\pi \in \Pi} \rightsquigarrow_\pi$.

- Let $L_{\mathsf{intervals}}$ be the set of MSCs (M,ζ) where the mapping $\zeta\colon E \to \{0,1\}$ defines (non-empty maximal) intervals $[e_1, e'_1], \dots, [e_k, e'_k]$ of 1-labeled events on process p and a sequence of 1-labeled events $f_1 < \dots < f_k$ on process q, such that, for all $1 \leq i \leq k$, we have $\downarrow_p(e_i) \subseteq \downarrow_p(f_i)$ and $\uparrow_p(e'_i) \subseteq \uparrow_p(f_i)$. This is illustrated in Figure 2.
- Let L_{left} be the set of all MSCs in $L_{\mathsf{intervals}}$ such that, for all $1 \leq i \leq k$ and $\pi \in \Pi_{p,q}$, if $\mathsf{pred}_\pi(f_i)$ is defined, then $\mathsf{pred}_\pi(f_i) < e_i$.
- Let L_{right} be the set of all MSCs in $L_{\mathsf{intervals}}$ such that, for all $1 \leq i \leq k$ and $\pi \in \Pi_{q,p}$, if $\mathsf{succ}_\pi(f_i)$ is defined, then $e'_i < \mathsf{succ}_\pi(f_i)$.

Note that $L_{\mathsf{parallel}} \subseteq L_{\mathsf{intervals}}$. The converse inclusion does not hold in general, since the intervals in MSCs from $L_{\mathsf{intervals}}$ may be too large. However, we have:

▶ **Lemma 16.** $L_{\mathsf{parallel}} = L_{\mathsf{left}} \cap L_{\mathsf{right}}$.

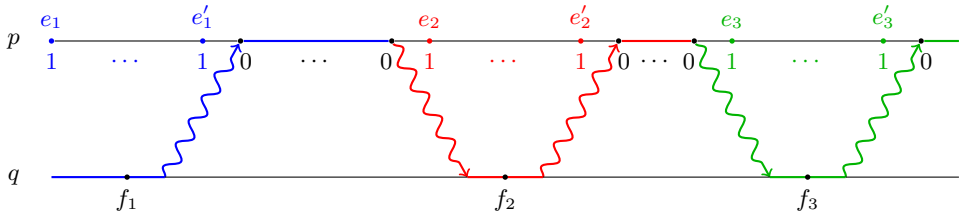

Figure 2 Constructions of $\mathcal{A}_{\mathsf{intervals}}$ and $\mathcal{A}_{\mathsf{parallel}}$.

Proof. Let $(M,\zeta) \in L_{\mathsf{parallel}}$ and $i \in \{1,\dots,k\}$. By definition, $[e_i, e'_i] = \|_p(f_i)$. Since $[e_i, e'_i]$ is non-empty, we have $\downarrow_p(e_i) = \downarrow_p(f_i)$ and $\uparrow_p(e'_i) = \uparrow_p(f_i)$. Hence, $(M,\zeta) \in L_{\mathsf{left}} \cap L_{\mathsf{right}}$.

Now, let $(M,\zeta) \in L_{\mathsf{left}} \cap L_{\mathsf{right}}$. Since $(M,\zeta) \in L_{\mathsf{intervals}}$, we have $\|_p(f_i) \subseteq [e_i, e'_i]$ for all i. Assume that there is $e \in [e_i, e'_i] \setminus \|_p(f_i)$. For instance $e \in \downarrow_p(f_i)$. Then, there exists $\pi \in \Pi_{p,q}$ such that $e \rightsquigarrow_\pi f_i$. Hence $e \le \mathsf{pred}_\pi(f_i)$, a contradiction with $(M,\zeta) \in L_{\mathsf{left}}$. ◀

A CFM for L_{parallel}. The last piece of the puzzle is a CFM $\mathcal{A}_{\mathsf{parallel}}$ such that $L(\mathcal{A}_{\mathsf{parallel}}) = L_{\mathsf{parallel}}$. It is built as the product (intersection) of CFMs $\mathcal{A}_{\mathsf{intervals}}$, $\mathcal{A}_{\mathsf{left}}$, and $\mathcal{A}_{\mathsf{right}}$.

▶ **Lemma 17.** *There is a CFM $\mathcal{A}_{\mathsf{intervals}}$ with a constant number of states such that we have $L(\mathcal{A}_{\mathsf{intervals}}) = L_{\mathsf{intervals}}$.*

Proof. Again, we implement a sort of token passing, which is illustrated in Figure 2. The token starts on process p iff the first p-event is labeled 0; otherwise, it must start on q. Similarly, the token ends on process p iff the last p-event is labeled 0; otherwise, it must end on q. Process p reads 0's when it holds the token, and 1's when it does not. Moreover, after sending the token, process p must read some 1-labeled events. When sent by p (respectively q), the token must reach q (respectively p) before returning to p (respectively q). Finally, process q reads only 0-labeled events when it does not hold the token. Moreover, process q checks that, within every maximal interval where it holds the token, there is exactly one 1-labeled event.

It is easy to check that $(M,\zeta) \in L_{\mathsf{intervals}}$ iff there exists a path along which the token is passed and satisfying the above conditions. ◀

We now show that there exists a CFM $\mathcal{A}_{\mathsf{left}}$ that accepts an MSC $(M,\zeta) \in L_{\mathsf{intervals}}$ iff $(M,\zeta) \in L_{\mathsf{left}}$. The idea is that $\mathcal{A}_{\mathsf{left}}$ guesses a coloring of the intervals of marked events such that checking $\mathsf{pred}_\pi(f_i) < e_i$ can be replaced with checking that $\mathsf{pred}_\pi(f_i)$ is not in an interval with the same color as $[e_i, e'_i]$. We need to prove that such a coloring exists, and that the colors associated with the $\mathsf{pred}_\pi(f_i)$ can be computed by the CFM.

▶ **Lemma 18.** *Let $(M,\zeta) \in L_{\mathsf{left}}$, let $[e_1, e'_1], \dots, [e_k, e'_k]$ be the sequence of maximal intervals of 1-labeled events on process p, and $f_1 < \cdots < f_k$ the corresponding events labeled 1 on process q. There exists a coloring $\chi\colon \{1,\dots,k\} \to \{1,\dots,|\Pi_{p,q}|+1\}$ such that, for all $i,j \in \{1,\dots,k\}$ and $\pi \in \Pi_{p,q}$, $\mathsf{pred}_\pi(f_j) \in [e_i, e'_i]$ implies $\chi(i) \neq \chi(j)$.*

Proof. We write $i \rightsquigarrow j$ when there exists $\pi \in \Pi_{p,q}$ such that $\mathsf{pred}_\pi(f_j) \in [e_i, e'_i]$. Notice that if $i \rightsquigarrow j$, then $i < j$ (otherwise, we would have $e_i \le \mathsf{pred}_\pi(f_j) < f_j < f_i$, hence, $e_i \le \mathsf{pred}_\pi(f_j) \le \mathsf{pred}_\pi(f_i)$, a contradiction with $(M,\zeta) \in L_{\mathsf{left}}$). So we can define χ by successively choosing colors for $1,\dots,k$: For all j, it suffices to choose a color $\chi(j) \in \{1,\dots,|\Pi_{p,q}|+1\}$ distinct from the at most $|\Pi_{p,q}|$ colors of indices $i < j$ such that $i \rightsquigarrow j$. ◀

▶ **Lemma 19.** *Let Θ be a finite set. There exists a (deterministic) CFM with $|\Theta|^{\mathcal{O}(|P|!)}$ states recognizing the set of doubly extended MSCs (M, θ, ξ) such that, for all events e, $\xi(e)$ is the partial function from Π to Θ such that $\xi(e)(\pi) = \theta(\mathsf{pred}_\pi(e))$.*

Proof. The CFM stores the label $\xi(e)$ of an event e in its state, and includes it in the message if e is a send event. At an event e on process u, the CFM checks that $\xi(e)(u) = \theta(e)$. Moreover, the CFM checks that:

- If e has no $\rightarrow$-predecessor and no $\lhd$-predecessor, then $\xi(e)(\pi)$ is undefined for all $\pi \neq u$.
- If e has one $\rightarrow$-predecessor f but no $\lhd$-predecessor, then $\xi(e)(\pi) = \xi(f)(\pi)$ for $\pi \neq u$.
- If e has one $\lhd$-predecessor g on process r, but no $\rightarrow$-predecessor, then $\xi(e)(\pi r u) = \xi(g)(\pi r)$, and $\xi(e)(\pi)$ is undefined if $\pi \neq u$ and π does not end with ru.
- If e has one $\rightarrow$-predecessor f and one $\lhd$-predecessor g on process r, then $\xi(e)(\pi r u) = \xi(g)(\pi r)$, and $\xi(e)(\pi) = \xi(f)(\pi)$ if $\pi \neq u$ and π does not end with ru. ◀

▶ **Lemma 20.** *There is a CFM $\mathcal{A}_{\mathsf{left}}$ with $2^{2^{\mathcal{O}(|P| \log |P|)}}$ states such that we have $L(\mathcal{A}_{\mathsf{left}}) \cap L_{\mathsf{intervals}} = L_{\mathsf{left}}$.*

Proof. Let $(M, \zeta) \in L_{\mathsf{intervals}}$ with $[e_1, e_1'], \ldots, [e_k, e_k']$ the non-empty maximal intervals of 1-labeled events on process p, and let $f_1 < \cdots < f_k$ be the corresponding 1-labeled events on process q. We can slightly modify $\mathcal{A}_{\mathsf{intervals}}$ so that on input (M, ζ), it guesses a coloring $\chi \colon \{1, \ldots, k\} \to \{1, \ldots, |\Pi_{p,q}| + 1\}$, and labels each event in $[e_i, e_i']$ with $\chi(i)$. The color of the upcoming interval $[e_i, e_i']$ is passed along with the token, so that at each f_i, the CFM has access to the color $\chi(i)$ (see Figure 2).

We can then compose that automaton with the CFM from Lemma 19, to compute, at each f_i and for all $\pi \in \Pi_{p,q}$ such that $\mathsf{pred}_\pi(f_i)$ is defined, the value $\zeta(\mathsf{pred}_\pi(f_i))$ and the color associated with $\mathsf{pred}_\pi(f_i)$. The CFM $\mathcal{A}_{\mathsf{left}}$ then checks that for all i and π, either $\mathsf{pred}_\pi(f_i)$ is undefined, or $\zeta(\mathsf{pred}_\pi(f_i)) = 0$, or the color associated with $\mathsf{pred}_\pi(f_i)$ is different from $\chi(i)$.

Suppose $(M, \zeta) \in L(\mathcal{A}_{\mathsf{left}}) \cap L_{\mathsf{intervals}}$. Then, for all i and π, $\mathsf{pred}_\pi(f_i)$ cannot be in an interval colored $\chi(i)$. In particular, this implies $\mathsf{pred}_\pi(f_i) \notin [e_i, e_i']$. Since $\uparrow_p(e_i') \subseteq \uparrow_p(f_i)$ and $\mathsf{pred}_\pi(f_i) \notin \uparrow_p(f_i)$, we deduce that $\mathsf{pred}_\pi(f_i) < e_i$ and $(M, \zeta) \in L_{\mathsf{left}}$. Conversely, suppose $(M, \zeta) \in L_{\mathsf{left}}$. Then, by Lemma 18, there exists a run in which the coloring guessed along the token passing is such that $\mathcal{A}_{\mathsf{left}}$ accepts. ◀

Finally, we obtain $\mathcal{A}_{\mathsf{parallel}}$ as the product (intersection) of $\mathcal{A}_{\mathsf{intervals}}$, $\mathcal{A}_{\mathsf{left}}$, and the "reversal" (or mirror) $\mathcal{A}_{\mathsf{right}}$ of $\mathcal{A}_{\mathsf{left}}$, which recognizes L_{right}. In fact, it is easy to see that CFMs are closed under reversal languages, in which both the process and the edge relations are inverted.

▶ **Lemma 21.** *There is a CFM $\mathcal{A}_{\mathsf{parallel}}$ with $2^{2^{\mathcal{O}(|P| \log |P|)}}$ states such that $L(\mathcal{A}_{\mathsf{parallel}}) = L_{\mathsf{parallel}}$.*

5 Conclusion

We showed that every EMSO^2 formula over MSCs can be effectively translated into an equivalent CFM of doubly exponential size, which is optimal. At the heart of our construction is a CFM $\mathcal{A}_{\mathsf{types}}$ of independent interest, which "outputs" the type of each event of an MSC. In particular, $\mathcal{A}_{\mathsf{types}}$ can be applied to other logics such as propositional dynamic logic (PDL), which combines modal operators and regular expressions [10]. It has been shown in [5] that every PDL formula can be translated into an equivalent CFM. We can extend this result by adding a modality $\langle \| \rangle$ to PDL, which "jumps" to some parallel event. For example, the formula $\neg \mathsf{E}(CS \wedge \langle \| \rangle CS)$ says that no two parallel events access a critical section. Note that

[5] considers infinite MSCs. However, it is easy to see that all our constructions can be extended to infinite MSCs.

A major open problem is whether every sentence from $\mathrm{FO}[\rightarrow, \lhd, \leq]$, with arbitrarily many variables, is equivalent to some CFM. To the best of our knowledge, the question is even open for the logic $\mathrm{FO}[\leq]$. Generally, it would be worthwhile to identify large classes of acyclic graphs of bounded degree such that all FO- or FO^2-definable languages (including the transitive closure of the edge relation) are "recognizable" (e.g., by a graph acceptor [25]).

References

1. R. Alur and P. Madhusudan. Adding nesting structure to words. *Journal of the ACM*, 56(3):1–43, 2009.
2. N. Bedon. Logic and branching automata. *Logical Methods in Computer Science*, 11(4), 2015.
3. H. Björklund and T. Schwentick. On notions of regularity for data languages. *Theoretical Computer Science*, 411(4-5):702–715, 2010.
4. M. Bojanczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Transactions on Computational Logic*, 12(4):27, 2011.
5. B. Bollig, D. Kuske, and I. Meinecke. Propositional dynamic logic for message-passing systems. *Logical Methods in Computer Science*, 6(3:16), 2010.
6. B. Bollig and M. Leucker. Message-passing automata are expressively equivalent to EMSO logic. *Theoretical Computer Science*, 358(2-3):150–172, 2006.
7. D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2), 1983.
8. J. Büchi. Weak second order logic and finite automata. *Z. Math. Logik, Grundlag. Math.*, 5:66–62, 1960.
9. C. C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98:21–52, 1961.
10. M. J. Fischer and R. E. Ladner. Propositional Dynamic Logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.
11. B. Genest, D. Kuske, and A. Muscholl. A Kleene theorem and model checking algorithms for existentially bounded communicating automata. *Information and Computation*, 204(6):920–956, 2006.
12. B. Genest, D. Kuske, and A. Muscholl. On communicating automata with bounded channels. *Fundamenta Informaticae*, 80(1-3):147–167, 2007.
13. E. Grädel and M. Otto. On logics with two variables. *Theor. Comput. Sci.*, 224(1-2):73–113, 1999.
14. M. Grohe and N. Schweikardt. Comparing the succinctness of monadic query languages over finite trees. In *Proceedings of CSL'03*, volume 2803 of *Lecture Notes in Computer Science*, pages 226–240. Springer, 2003.
15. W. Hanf. Model-theoretic methods in the study of elementary logic. In J. W. Addison, L. Henkin, and A. Tarski, editors, *The Theory of Models*. North-Holland, Amsterdam, 1965.
16. Lauri Hella, Matti Järvisalo, Antti Kuusisto, Juhana Laurinharju, Tuomo Lempiäinen, Kerkko Luosto, Jukka Suomela, and Jonni Virtema. Weak models of distributed computing, with connections to modal logic. *Distributed Computing*, 28(1):31–53, 2015. doi:10.1007/s00446-013-0202-3.
17. J. G. Henriksen, M. Mukund, K. Narayan Kumar, M. Sohoni, and P. S. Thiagarajan. A theory of regular MSC languages. *Information and Computation*, 202(1):1–38, 2005.
18. D. Kuske. Infinite series-parallel posets: Logic and languages. In *Proceedings of ICALP'00*, volume 1853 of *LNCS*, pages 648–662. Springer, 2000.

19 D. Kuske. Regular sets of infinite message sequence charts. *Information and Computation*, 187:80–109, 2003.

20 Antti Kuusisto. Modal logic and distributed message passing automata. In Simona Ronchi Della Rocca, editor, *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, volume 23 of *LIPIcs*, pages 452–468. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013. doi:10.4230/LIPIcs.CSL.2013.452.

21 Fabian Reiter. Distributed graph automata. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 192–201. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.27.

22 Fabian Reiter. Asynchronous distributed automata: A characterization of the modal mu-fragment. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 100:1–100:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.100.

23 J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, 1968.

24 W. Thomas. On logical definability of trace languages. In *Proceedings of Algebraic and Syntactic Methods in Computer Science (ASMICS)*, Report TUM-I9002, Technical University of Munich, pages 172–182, 1990.

25 W. Thomas. Elements of an automata theory over partial orders. In *Proceedings of POMIV'96*, volume 29 of *DIMACS*. AMS, 1996.

26 B. A. Trakhtenbrot. Finite automata and monadic second order logic. *Siberian Math. J*, 3:103–131, 1962. In Russian; English translation in *Amer. Math. Soc. Transl.* 59, 1966, 23–55.

27 M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings of LICS'86*, pages 332–344. IEEE Computer Society, 1986.

28 P. Weis. *Expressiveness and Succinctness of First-order Logic on Finite Words*. Ph.D. thesis, University of Massachusetts Amherst, 2011.

29 W. Zielonka. Notes on finite asynchronous automata. *R.A.I.R.O. — Informatique Théorique et Applications*, 21:99–135, 1987.