A Hierarchy of Implementable MSC Languages

Benedikt Bollig[1] and Martin Leucker[2]

[1] Lehrstuhl für Informatik II, RWTH Aachen, Germany
bollig@informatik.rwth-aachen.de
[2] Institut für Informatik, TU Munich, Germany
leucker@in.tum.de

Abstract. We develop a unifying theory of message-passing automata (MPAs) and MSC languages. We study several variants of *regular* as well as *product* MSC languages, their closure under finite union and their intersection. Furthermore, we analyse the expressive power of several variants of MPAs and characterize the language classes of interest by the corresponding classes of MPAs.

1 Introduction

A common design practice when developing communicating systems is to start with drawing scenarios showing the intended interaction of the system to be. The standardized notion of *message sequence charts* (MSCs, [ITU99]) is widely used in industry to formalize such typical behaviors.

A message sequence chart defines a set of processes and a set of communication actions between these processes. In the visual representation of an MSC, processes are drawn as vertical lines and interpreted as time axes. A labeled arrow from one line to a second corresponds to the communication event of sending a message from the first process to the second. Collections of MSCs are used to capture the scenarios that a designer might want the system to follow or to avoid. Figure 1 shows four simple MSCs.

The next step in the design process is to come up with a high-level model of the system to be. Such a model is usually formalized as a state machine or as an automaton. In the setting of MSCs, where the notion of distributed processes is central, one asks for distributed automata models. The components of such a distributed automaton should communicate by message passing to reflect the message flow indicated by MSCs. Thus, we are after message-passing automata (MPA) *realizing* or *implementing* the behavior given in form of scenarios.

In the setting of finite words, there has been an extensive study of several classes of languages and corresponding characterizations by means of automata. Algebraic and logical characterizations have been obtained as well.

In the setting of MSCs, however, a correspondence of languages and characterizing automata models or characterizing logical specifications is still at the beginning. In this paper, we provide a comprehensive study of MSC languages and message-passing automata. Our work can be summarized in two pictures, Figure 2 and Figure 3, which we explain in the rest of this introduction.

F. Wang (Ed.): FORTE 2005, LNCS 3731, pp. 53–67, 2005.

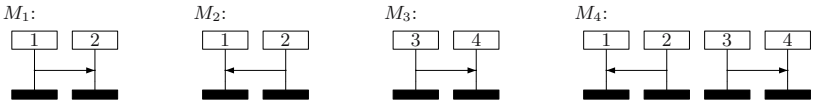


Fig. 1. The language of a finite MPA

When realizing communicating systems, two dimensions are important: *finiteness* and *independence*.

Finiteness. Most often, the intended system is required to have a finite set of (control) states. In the case of words, *regular* languages admit a characterization in terms of *finite* state machines. In general, the notion of *regularity* aims at *finite* representations. Thus, as in the word case, one is interested in *regular* MSC languages. While for the word case a consensus is reached for the notion of a regular language, this is not the case for MSC languages.

There have been several proposals for the *right* notion of regularity for MSC languages. Henriksen et al. started in [HMKT00], proposing that an MSC language is regular when its set of linearizations is regular. We denote this class of languages by $\mathcal{R}$ (see Figure 2).

One could likewise argue to call a set of MSCs regular, when it is definable by monadic second-order (MSO) logic (adjusted to MSCs), since this important property for verification holds for regular sets of words. This view was also pursued in [HMKT00] and it is shown that this class coincides with $\mathcal{R}$ when formulas are supposed to define *bounded* MSC languages. Intuitively, an MSC language is bounded iff its structure exhibits a bound on the number of send events that have not been received yet by the receiving process. For example, the MSC language $\{M_1\}^*$ (where M_1 is taken from Figure 1) is not bounded and hence not regular. It induces the set of MSCs that send arbitrarily many messages from process 1 to process 2. The set of all its linearizations gives rise to the set of all words that have the same number of send and receive events, where, for every prefix, the number of send events is larger or equal to the number of receive events. This language is not regular, since, intuitively, we would need an unbounded counter for the send events. In [GMK04], a more general notion of regularity was studied, which gives rise to *existentially* bounded MSC languages. Those languages require any MSC to exhibit at least one linearization that is compatible with a fixed channel capacity.

Independence. On the other hand, the systems exemplified by MSCs are distributed in nature and the notion of a *process* is central. It is therefore natural to consider every process to be represented as a single state machine or transition system. Furthermore, one defines a notion of communication, describing the way these parallel systems work together.

Languages defined by finite transition systems working in parallel are known as *product languages* and were initially studied in [Thi95] for Mazurkiewicz traces. That paper discusses finite-state systems. There is a common initial state and two different notions of acceptance condition. Either the acceptance condition is *local*, i.e. every process decides on its own when to stop, or it is *global*,

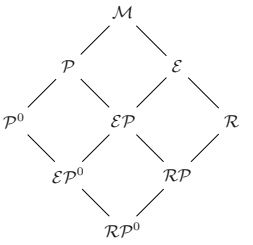


Fig. 2. The hierarchy of regular and product MSC languages

which allows to select certain combinations of local final states to be a valid point for termination. It is clear that (provided the system has a single initial state) the latter notion of acceptance is more expressive than local acceptance.

In the context of MSC languages, [AEY00] studied classes of MSCs taking up the idea of product behavior. Languages of MSCs are required to be closed under *inference*. The idea can be described looking at the setting in Figure 1. When realizing the behavior described by M_2 and by M_3, it is argued that the behavior of M_4 is also a possible one: Since processes 1 and 2 do not synchronize with processes 3 and 4, the four processes do not know whether the behavior of M_2 should be realized or that of M_3. We call the class of MSC languages that is closed under such inference *weak product MSC languages* and denote it by $\mathcal{P}^0$ (see Figure 2). Note that, in [AEY00], no finiteness condition was studied.

In simple words, *product languages* respect *independence.*

Extensions. Let us study extensions of the two existing approaches. When thinking about an automata model realizing MSC languages, the allowance of different initial states or global final states will give us classes of languages closed under finite union. For example, one could realize exactly the set consisting of M_2 and M_3 (without M_4). Thus, when considering finite unions of sets of $\mathcal{P}^0$ languages, one obtains the richer class of *product MSC languages*, denoted by $\mathcal{P}$. Combining the ideas of *independence* and *finiteness*, we get $\mathcal{RP}^0$ or $\mathcal{RP}$.

The drawback of the regularity notion used for $\mathcal{R}$ is that the simple language $\{M_1\}^*$ is not regular, as mentioned before. Let us once again turn to the logical point of view. Recall that MSO logic interpreted over bounded MSCs captures regular and, thus, bounded behavior. It was shown in [BL04] that MSO logic interpreted over the whole class of MSCs turns out to be too expressive to be compatible with certain finite message-passing automata as introduced beneath. These automata, though they employ finite state spaces for each process (but not globally), are capable of generating unbounded behavior using a priori unbounded channels. In [BL04], existential MSO, a fragment of MSO was shown to be expressively equivalent to finite message-passing automata. We therefore introduce the class of *EMSO-definable languages* ($\mathcal{E}$), which lifts the boundedness restriction without abandoning the existence of a finite automata-theoretic counterpart. Together with independence, we obtain the class $\mathcal{EP}$ or, when starting from $\mathcal{RP}^0$, the class, $\mathcal{EP}^0$.

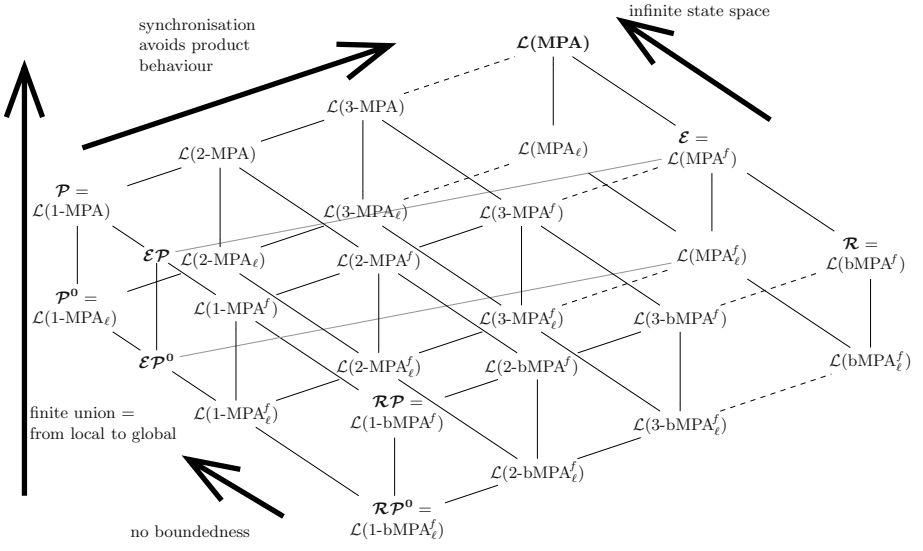


Fig. 3. A Hierarchy of MSC Languages

This completes the explanation of the languages shown in Figure 1, and, one main result of the paper is to show that languages actually form the hierarchy that is suggested in the figure.

Automata Models. So let us exhibit corresponding machine models and call our attention to Figure 3. We study several variants of message-passing automata (MPA), which consist of components that communicate through channels. All studied systems have a single initial state. We use (variations of) n-bMPA$^f_\ell$ to indicate several classes of MPAs. Dropping the requirement that the local state spaces are finite is indicated by the missing superscript f. When considering finite unions of languages, we have to move towards global acceptance conditions (rather than local), indicated by a missing ℓ.

When extending the expressiveness from regularity to EMSO-definable languages, we drop the boundedness condition of MSCs. This is represented in Figure 3 by a missing b. It can be shown that, when realizing regular languages, one needs so-called *synchronization messages* that are used to tell other components which transition was taken. They are used in [HMKT00] and [GMSZ02] to extend expressiveness. We show that the more of these messages are allowed, the more expressive power we have. The restriction to n synchronization messages is described by a preceding n- in Figure 3.

MPAs are subject to active research. However, there is no agreement which automata model is the right one to make an MSC language *implementable*: While, for example, [GMSZ02] is based on MPA$^f_\ell$, [AEY00, Mor02] focus on 1-MPA$_\ell$. In [HMKT00], though a priori unbounded channels are allowed, the bounded model bMPA$^f_\ell$ suffices to implement regular MSC languages. We pro-

vide a unifying framework and classify automata models according to their state-space, synchronization behavior, acceptance mode, and, on a rather semantical level, whether they generate bounded or unbounded behavior.

Our second main result is to show the correspondence indicated in Figure 3.

2 Preliminaries

Let us recall some basic definitions and let Σ be an alphabet. A finite *Σ-labeled partial order* is a triple $\mathcal{P} = (E, \leq, \ell)$ where E is a finite set, $\leq$ is a partial-order relation on E, i.e., it is reflexive, transitive, and antisymmetric, and $\ell : E \to \Sigma$ is a *labeling function*. For $e, e' \in E$, we write $e \lessdot e'$ if both $e < e'$ and, for any $e'' \in E$, $e < e'' \leq e'$ implies $e'' = e'$. A linearization of $\mathcal{P}$ is an extension $(E, \leq', \ell)$ of $\mathcal{P}$ such that $\leq' \supseteq \leq$ is a linear order. As we will identify isomorphic structures in the following, a linearization of $\mathcal{P}$ will be seen as a word over Σ. The set of linearizations of $\mathcal{P}$ is denoted by $Lin(\mathcal{P})$.

Let us fix a finite set *Proc* of at least two *processes*, which communicate with one another via message passing.[1] Communication proceeds through channels via executing communication actions. We denote by *Ch* the set $\{(p, q) \mid p, q \in Proc,\ p \neq q\}$ of reliable FIFO *channels*. Given a process $p \in Proc$, we furthermore set $\mathcal{C}_p$ to be $\{\mathtt{send}(p, q) \mid (p, q) \in Ch\} \cup \{\mathtt{rec}(p, q) \mid (p, q) \in Ch\}$, the set of *actions* of process p. The action $\mathtt{send}(p, q)$ is to be read as "p sends a message to q", while $\mathtt{rec}(q, p)$ is the complementary action of receiving a message sent from p to q. Accordingly, we set $Com := \{(\mathtt{send}(p, q), \mathtt{rec}(q, p)) \mid (p, q) \in Ch\}$. Moreover, let $\mathcal{C}$ stand for the union of the $\mathcal{C}_p$. Observe that an action $p\theta q$ ($\theta \in \{!, ?\}$) is performed by process p, which is indicated by $P(p\theta q) = p$. A *message sequence chart* (MSC) (over *Proc*) is a tuple $(E, \{\leq_p\}_{p \in Proc}, <_c, \ell)$ such that

- E is a finite set of *events*,
- ℓ is a mapping $E \to \mathcal{C}$,
- for any $p \in Proc$, $\leq_p$ is a linear order on $E_p := \ell^{-1}(\mathcal{C}_p)$,
- $<_c \subseteq E \times E$ such that both, for any $e \in E$, there is $e' \in E$ satisfying $e <_c e'$ or $e' <_c e$ and, for any $(e_1, e_1') \in <_c$, there are $p, q \in Proc$ satisfying
  - $\ell(e_1) = \mathtt{send}(p, q)$
  - $\ell(e_1') = \mathtt{rec}(q, p)$
  - for any $(e_2, e_2') \in <_c$ with $\ell(e_1) = \ell(e_2)$, it holds $e_1 \leq_p e_2$ iff $e_1' \leq_q e_2'$,
- $\leq := \left(<_c \cup \bigcup_{p \in Proc} \leq_p\right)^*$ is a partial-order relation on E.

The set of MSCs is denoted by $\mathbb{MSC}$. (As *Proc* will be fixed in the following, a corresponding reference is omitted.) Let $M = (E, \{\leq_p\}_{p \in Proc}, <_c, \ell) \in \mathbb{MSC}$. The behavior of M might be split into its components $M \restriction p := (E_p, \leq_p, \ell_{|E_p})$, $p \in Proc$, each of which represents the behavior of one single agent and can be seen as a word over $\mathcal{C}_p$. In turn, given a collection of words $w_p \in \mathcal{C}_p^*$, there is at most one MSC M such that, for any $p \in Proc$, $w_p = M \restriction p$. We will write $Lin(M)$ to denote $Lin((E, \leq, \ell))$, which extends to sets of MSCs as usual. Given

[1] In proofs, we sometimes silently assume the existence of more than two processes.

a set of words $L' \subseteq \mathcal{C}^*$, we say L' is an *MSC word language* if $L' = Lin(L)$ for some $L \subseteq \mathbb{MSC}$. In turn, a set $L \subseteq \mathbb{MSC}$ is uniquely determined by $Lin(L)$.

Let $B \geq 1$. We call a word $w \in \mathcal{C}^*$ *B-bounded* if, for any prefix v of w and any $(p,q) \in Ch$, $|v|_{\mathtt{send}(p,q)} - |v|_{\mathtt{rec}(q,p)} \leq B$ where $|v|_\sigma$ denotes the number of occurrences of σ in v. An MSC $M \in \mathbb{MSC}$ is called *B-bounded* if, for any $w \in Lin(M)$, w is B-bounded. The set of B-bounded MSCs is denoted by $\mathbb{MSC}_B$. An MSC language $L \subseteq \mathbb{MSC}$ is called B-bounded if $L \subseteq \mathbb{MSC}_B$. Moreover, we call L bounded if it is B-bounded for some B. In other words, boundedness is safe in the sense that any possible execution sequence does not claim more memory than some given upper bound (whereas existential boundedness, which, however, is not considered in this paper, allows an MSC to be executed even if this does not apply to each of its linear extensions [GMK04]).

3 Implementable MSC Languages

3.1 Regular MSC Languages

There have been several proposals for the *right* notion of regularity for MSC languages. In their seminal work [HMKT00, HMK+04], Henriksen et al. consider an MSC language to be regular if its set of linearizations forms a regular word language. For example, the MSC language $\{M_1\}^*$ (where M_1 is taken from Figure 1), which allows to concatenate M_1 arbitrarily often[2], is not bounded and hence cannot be regular. It induces the set of MSCs that send arbitrarily many messages from process 1 to process 2. The corresponding set of linearizations gives rise to a set of words that show the same number of send and receive events. This language is not recognizable in the free word monoid. In contrast, the language $\{M_1 \cdot M_2\}^*$ is regular, as its word language can be easily realized by a finite automaton. Thus, regularity aims at finiteness of the underlying *global* system, which incorporates the state of a communication channel.

Definition 1 ([HMKT00]). *A set $L \subseteq \mathbb{MSC}$ is called* regular *if $Lin(L)$ is a regular word language over $\mathcal{C}$.*

The class of regular MSC languages is denoted by $\mathcal{R}$.

Corollary 1 ([HMKT00]). *Any regular MSC language is bounded.*

As mentioned above, $\{M_1\}^*$ with M_1 again taken from Figure 1 is not regular. However, it is *existentially*-bounded [GMK04] and, as we will see in the next section, there is a simple finite message-passing automaton accepting $\{M_1\}^*$. Thus, we are looking for another, extended notion of *regularity*.

3.2 (E)MSO-Definable MSC Languages

Formulas from monadic second-order (MSO) logic (over *Proc*) involve first-order variables $x, y, \ldots$ for events and second-order variables $X, Y, \ldots$ for sets of events.

[2] Here, concatenation is meant to be asynchronous.

They are built up from the atomic formulas $\ell(x) = \sigma$ (for $\sigma \in \mathcal{C}$), $x \in X$, $x \lessdot_p y$ (for $p \in Proc$), $x <_c y$, and $x = y$ and furthermore allow the connectives $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$ as well as the quantifiers $\exists$, $\forall$, which can be applied to either kind of variable. Formulas without free variables, which do not occur within the scope of a quantifier, are called sentences. Given an MSC $M = (E, \{\leq_p\}_{p \in Proc}, <_c, \ell)$ and an MSO sentence φ, the validity of the satisfaction relation $M \models \varphi$ is defined canonically with the understanding that first-order variables range over events from E and second-order variables over subsets of E. The *language* of φ, denoted by $L(\varphi)$, is the set of MSCs M with $M \models \varphi$. The class of subsets of $\mathbb{MSC}$ that can be defined by some MSO sentence φ is denoted by $\mathcal{MSO}$. An important fragment of MSO logic is captured by *existential* MSO (EMSO) formulas, which are of the form $\exists X_1 \ldots \exists X_n \varphi$ where φ does not contain any set quantifier. In many cases, the restriction to EMSO formulas suffices to characterize recognizability in terms of automata, e.g., in the domains of words, trees, and Mazurkiewicz traces. Sometimes, however, we even have to restrict to EMSO formulas not to exceed recognizability in terms of automata, because full MSO logic is too expressive in general. In fact, the latter applies to MSCs [BL04]. The class of EMSO-definable MSC languages will be denoted by $\mathcal{E}$.

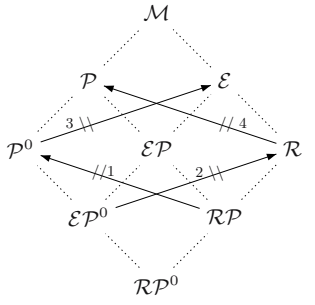
3.3 Product MSC Languages

Languages defined by finite transition systems working in parallel are known as *product languages* and were initially studied by Thiagarajan in [Thi95] in the domain of Mazurkiewicz traces where distributed components communicate executing actions simultaneously rather than sending messages. Taking up the idea of product behavior, [AEY00] considers MSC languages that are closed under *inference*, which can be described by the setting depicted in Figure 1. Attempting to realize the MSC language $\{M_2, M_3\}$, one might argue that the behavior of M_4 is a feasible one, too. As processes 1 and 2 do not get in touch with processes 3 and 4, it is not clear to a single process whether to realize the behavior of M_2 or that of M_3 so that, finally, M_4 might be *inferred* from $\{M_2, M_3\}$. We call a set of MSCs that is closed under such an inference a *weak product MSC language*. Let us be more precise and, given $L \subseteq \mathbb{MSC}$ and $M \in \mathbb{MSC}$, write $L \vdash_{Proc} M$ if $\forall p \in Proc : \ \exists M' \in L : \ M' \restriction p = M \restriction p$.

Definition 2. *A set $L \subseteq \mathbb{MSC}$ is called a* weak product MSC language *if, for any $M \in \mathbb{MSC}$, $L \vdash_{Proc} M$ implies $M \in L$ [AEY00]. The finite union of weak product MSC languages is called a* product MSC language.

We let $\mathcal{P}^0$ and $\mathcal{P}$ denote the classes of weak product languages and, respectively, product languages.

In other words, an MSC language L is a weak product MSC language if every MSC that agrees on each process line with some MSC from L is contained in L, too. Getting back to Figure 1, M_4 agrees with M_2 on the first two process lines and with M_3 on the remaining two. Thus, M_4 belongs to any weak product language containing both M_2 and M_3. As global knowledge of an underlying system, one often allows several global initial or final states. This is the reason for



1	$\{M_2, M_3\}$
2	$\{M_2\}^*$
3	[BL04]
4	L_{N+1} with $N \geq 1$ (cf. Lemma 10)

Fig. 4. Strictness and incomparability in the hierarchy

considering finite unions of weak product languages. For example, $\{M_2, M_3\}$ is a product MSC language, while $\{M_4\}^*$ is not. Let us bring together the concepts of product behavior and regularity.

Definition 3. *We call $\mathcal{R} \cap \mathcal{P}^0$ the class of* weak regular product MSC languages *and denote it by $\mathcal{RP}^0$. Furthermore, an MSC language L is a* regular product MSC language*, denoted by $L \in \mathcal{RP}$, if it is the finite union of sets from $\mathcal{RP}^0$.*

Let us now extend our study towards product languages in combination with EMSO-definable languages. As the class of EMSO-definable languages turned out to capture exactly the class of languages implementable in terms of a finite message-passing automaton, we rather concentrate on EMSO-definable languages than on MSO-definable ones [BL04].

Definition 4. *We call $\mathcal{E} \cap \mathcal{P}^0$ the class of* weak EMSO-definable product MSC languages *and denote it by $\mathcal{EP}^0$. An MSC language L is an* EMSO-definable product MSC language *($L \in \mathcal{EP}$) if it is the finite union of sets from $\mathcal{EP}^0$.*

Theorem 1. *The classes of languages proposed so far draw the picture shown in Figure 2. The hierarchy is strict.*

Proof. $\mathcal{R} \subseteq \mathcal{E}$ has been shown, for example, in [BL04]. The other inclusions are straightforward. It remains to show strictness and incomparability. Consider the MSCs M_2 and M_3 from Figure 1. For a (crossed) arrow from a class of MSC languages $\mathfrak{C}_1$ to a class $\mathfrak{C}_2$ in Figure 4, the tabular aside specifies an MSC language L with $L \in \mathfrak{C}_1$ and $L \notin \mathfrak{C}_2$.

3.4 Product MSC Languages vs. Product Trace Languages

Product languages have been introduced first in the framework of (Mazurkiewicz) traces. So let us compare traces and MSCs in the scope of regular MSC languages and justify that, in this respect, we have chosen the same terminology for traces and MSCs. In particular, we raise the hope that results and logics regarding

product trace languages are amenable to MSCs, such as the local temporal logic PTL, which is tailored to systems that support product behavior [Thi95].

Like MSCs, traces preserve some partial-order properties of a distributed system. Given a set $[K] := \{1, \ldots, K\}$ of *agents*, $K \geq 1$, they are based on a distributed alphabet $(\Sigma_1, \ldots, \Sigma_K)$, a tuple of (not necessarily disjoint) alphabets. Elements from Σ_i are understood to be actions that are performed by agent i. Let in the following $\tilde{\Sigma} = (\Sigma_1, \ldots, \Sigma_K)$ be a distributed alphabet, let Σ stand for the union of alphabets Σ_i, and let, for $a \in \Sigma$, $loc(a) := \{i \in [K] \mid a \in \Sigma_i\}$ denote the set of agents that are involved in the action a. A distributed alphabet $\tilde{\Sigma}$ determines a *dependence relation* $\mathcal{D}_{\tilde{\Sigma}} = (\Sigma, D)$ where $D = \{(a, b) \in \Sigma \times \Sigma \mid loc(a) \cap loc(b) \neq \emptyset\}$ is a reflexive and symmetric binary relation on Σ. Thus, actions a and b are understood to be *dependent* if they can both be performed by one and the same sequential agent.

A *(Mazurkiewicz) trace* over $\tilde{\Sigma}$ is a Σ-labeled partial order $(E, \leq, \ell)$ such that, for any $e, e' \in E$, $e \lessdot e'$ implies $(\ell(e), \ell(e')) \in D$ and $(\ell(e), \ell(e')) \in D$ implies $e \leq e'$ or $e' \leq e$. The set of traces over $\tilde{\Sigma}$ is denoted by $\mathbb{TR}(\tilde{\Sigma})$. As in the MSC case, the behavior of a trace $T \in \mathbb{TR}(\tilde{\Sigma})$ can be split into components $T \restriction i := (E_i, \leq \cap (E_i \times E_i), \ell_{|E_i})$ (where $E_i := \ell^{-1}(\Sigma_i)$), each of which can be seen as a word over Σ_i and represents the behavior of one single agent. Also, given a collection of words $w_i \in \Sigma_i^*$, there is at most one trace T such that, for any $i \in [K]$, $w_i = T \restriction i$. A set $L \subseteq \mathbb{TR}(\tilde{\Sigma})$ is called *regular* if $Lin(L)$ is a regular word language over Σ. The class of regular trace languages over $\tilde{\Sigma}$ is denoted by $\mathcal{R}_{\mathbb{TR}(\tilde{\Sigma})}$. Let $L \subseteq \mathbb{TR}(\tilde{\Sigma})$ and $T \in \mathbb{TR}(\tilde{\Sigma})$. Similarly to MSCs, we write $L \vdash_{\tilde{\Sigma}} T$ if, for any $i \in [K]$, there is $T' \in L$ such that $T' \restriction i = T \restriction i$. A set $L \subseteq \mathbb{TR}(\tilde{\Sigma})$ is called a *weak product trace language* (over $\tilde{\Sigma}$) ($L \in \mathcal{P}^0_{\mathbb{TR}(\tilde{\Sigma})}$) if, for any $T \in \mathbb{TR}(\tilde{\Sigma})$, $L \vdash_{\tilde{\Sigma}} T$ implies $T \in L$. A set $L \subseteq \mathbb{TR}(\tilde{\Sigma})$ is called a *product trace language* ($L \in \mathcal{P}_{\mathbb{TR}(\tilde{\Sigma})}$) if it is the finite union of weak product trace languages [Thi95]. The classes $\mathcal{RP}^0_{\mathbb{TR}(\tilde{\Sigma})}$ and $\mathcal{RP}_{\mathbb{TR}(\tilde{\Sigma})}$ are defined as expected.

We now recall in how far bounded MSC languages can be seen as trace languages over an appropriate alphabet [Kus03]. Let B be a positive natural. We define $\mathcal{D}_B$ to be the dependence alphabet $(\mathcal{C} \times \{1, \ldots, B\}, D_B)$ where $(\sigma_1, n_1) D_B (\sigma_2, n_2)$ if $P(\sigma_1) = P(\sigma_2)$ or $((\sigma_1, \sigma_2) \in Com \cup Com^{-1}$ and $n_1 = n_2)$. Setting Co to be $\{(\sigma, \tau, n) \mid (\sigma, \tau) \in Com,\ n \in \{1, \ldots, B\}\}$, let $\tilde{\Sigma}_B$ be the distributed alphabet $(\overline{\mathcal{C}}_\gamma)_{\gamma \in Proc \cup Co}$ where, for $p \in Proc$, $\overline{\mathcal{C}}_p := \mathcal{C}_p \times \{1, \ldots, B\}$ and, for $(\sigma, \tau, n) \in Co$, $\overline{\mathcal{C}}_{(\sigma,\tau,n)} := \{(\sigma, n), (\tau, n)\}$. Note that, given $B \geq 1$, $\mathcal{D}_{\tilde{\Sigma}_B} = \mathcal{D}_B$. To an MSC $M = (E, \{\leq_p\}_{p \in Proc}, <_c, \ell) \in \mathbb{MSC}_B$, we assign the Mazurkiewicz trace $Tr_B(M) := (E, \leq, \ell')$ where for each $e \in E$, we define $\ell'(e)$ to be the new labeling $(\ell(e), |\{e' \in E_{P(\ell(e))} \mid e' \leq e\}| \mod B)$. According to [Kus03], $Tr_B(M)$ is a trace over $\tilde{\Sigma}_B$ for any $M \in \mathbb{MSC}_B$. Note that the mapping $Tr_B : \mathbb{MSC}_B \to \mathbb{TR}(\tilde{\Sigma}_B)$ is injective. It is canonically extended towards MSC languages. Thus, involving some relabeling, an MSC language $L \subseteq \mathbb{MSC}_B$ can be converted into some trace language $Tr_B(L) \subseteq \mathbb{TR}(\tilde{\Sigma}_B)$.

To make clear in the following when we address a class of MSC languages rather than trace languages, we write, for example, $\mathcal{R}_{\mathbb{MSC}}$ instead of simply $\mathcal{R}$.

Lemma 1 ([Kus02]). *For any $B \geq 1$ and any $L \subseteq \mathbb{MSC}_B$, $L \in \mathcal{R}_{\mathbb{MSC}}$ iff $Tr_B(L) \in \mathcal{R}_{\mathbb{TR}(\tilde{\Sigma}_B)}$.*

We now show that the above correspondence carries over to product behavior.

Lemma 2. *For any $B \geq 1$ and $L \subseteq \mathbb{MSC}_B$, $L \in \mathcal{RP}^0_{\mathbb{MSC}}$ iff $Tr_B(L) \in \mathcal{RP}^0_{\mathbb{TR}(\tilde{\Sigma}_B)}$.*

Proof. According to Lemma 1, Tr_B and its inverse both preserve regularity.

"only if": Suppose $L \subseteq \mathbb{MSC}_B$ to be a weak regular product MSC language. Recall that $Tr_B(L)$ is a regular trace language over $\tilde{\Sigma}_B = (\overline{\mathcal{C}}_\gamma)_{\gamma \in Proc \cup Co}$. Moreover, let $T \in \mathbb{TR}(\tilde{\Sigma}_B)$ such that, for any $\gamma \in Proc \cup Co$, there is a trace $T_\gamma \in Tr_B(L)$ satisfying $T_\gamma \restriction \gamma = T \restriction \gamma$. Then, $T \in Tr_B(\mathbb{MSC}_B)$ and, in particular, we have $T_p \restriction p = T \restriction p$ and, thus, $Tr_B^{-1}(T_p) \restriction p = Tr_B^{-1}(T) \restriction p$ for any $p \in Proc$, which implies $Tr_B^{-1}(T) \in L$ and $T \in Tr_B(L)$.

"if": Suppose $L \subseteq \mathbb{MSC}_B$ to generate a weak regular trace language over $\tilde{\Sigma}_B$, i.e., $Tr_B(L) \in \mathcal{RP}^0(\tilde{\Sigma}_B)$, and let $M \in \mathbb{MSC}_B$ such that, for any $p \in Proc$, there is $M_p \in L$ with $M_p \restriction p = M \restriction p$. Trivially, we have that, for any $p \in Proc$, $Tr_B(M_p) \restriction p = Tr_B(M) \restriction p$. Moreover, for any $\gamma = (\mathtt{send}(p,q), \mathtt{rec}(q,p), n) \in Co$, $Tr_B(M_p) \restriction \gamma = Tr_B(M) \restriction \gamma$ (note that also $Tr_B(M_q) \restriction \gamma = Tr_B(M) \restriction \gamma$). This is because, in the trace of a B-bounded MSC, the n-th receipt of a message through (p,q) is ordered before sending from p to q for the $(n+B)$-th time. Altogether, we have $Tr_B(M) \in Tr_B(L)$ and, consequently, $M \in L$. □

Corollary 2. *For any $B \geq 1$ and any $L \subseteq \mathbb{MSC}_B$, $L \in \mathcal{RP}_{\mathbb{MSC}}$ iff $Tr_B(L) \in \mathcal{RP}_{\mathbb{TR}(\tilde{\Sigma}_B)}$.*

4 Message-Passing Automata

We now introduce and study *message-passing automata* (MPAs), our model of computation, which is is close to a real-life implementation of a message-passing system. MPAs can be considered to be the most common computation model for MSCs. An MPA is a collection of state machines that share one global initial state and several global final states. The machines are connected pairwise with a priori unbounded reliable FIFO buffers. The transitions of each component are labeled with send or receive actions. Hereby, a send action $p!q$ puts a message at the end of the channel from p to q. A receive action can be taken provided the requested message is found in the channel. To extend the expressive power, MPAs can send certain *synchronization messages*. Let us be more precise:

Definition 5 (Message-Passing Automaton). *A* message-passing automaton *(MPA) is a structure $\mathcal{A} = ((\mathcal{A}_p)_{p \in Proc}, \mathcal{D}, \overline{s}^{in}, F)$ such that*

- *$\mathcal{D}$ is a nonempty finite set of* synchronization messages*,*
- *for each $p \in Proc$, $\mathcal{A}_p$ is a pair (S_p, Δ_p) where S_p is a nonempty set of* (p-)local states *and $\Delta_p \subseteq S_p \times \mathcal{C}_p \times \mathcal{D} \times S_p$ is the set of* (p-)local transitions*,*

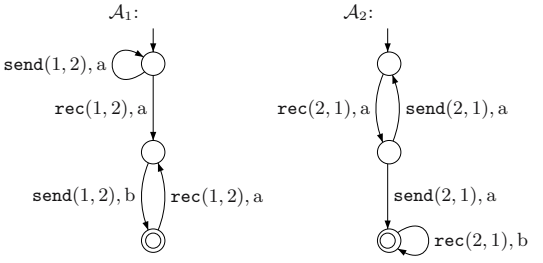


Fig. 5. A message-passing automaton

- $\overline{s}^{in} \in \prod_{p \in Proc} S_p$ *is the* global initial state*, and*
- $F \subseteq \prod_{p \in Proc} S_p$ *is a finite set of* global final states.

An MPA $\mathcal{A} = ((\mathcal{A}_p)_{p \in Proc}, \mathcal{D}, \overline{s}^{in}, F)$, $\mathcal{A}_p = (S_p, \Delta_p)$, is called

- an N-MPA, $N \geq 1$, if $|\mathcal{D}| = N$,
- *finite* if, for each $p \in Proc$, S_p is finite, and
- *locally-accepting* if there are sets $F_p \subseteq S_p$ such that $F = \prod_{p \in Proc} F_p$.

The class of MPAs is denoted by MPA, the class of finite MPAs by MPA^f. Furthermore, for a set $\mathfrak{C}$ of MPAs, we denote by N-$\mathfrak{C}$ the class of N-MPAs $\mathcal{A}$ and by $\mathfrak{C}_\ell$ the class of locally-accepting MPAs $\mathcal{A}$ with $\mathcal{A} \in \mathfrak{C}$, respectively. A locally-accepting finite 2-MPA with set of synchronization messages $\{\mathrm{a}, \mathrm{b}\}$ is illustrated in Figure 5.

In defining the behavior of an MPA, we adopt the view taken, for example, in [HMKT00, Mor02, GMSZ02], who suppose an MPA to run on linearizations of MSCs rather than on MSCs to reflect an operational behavior. Usually, such a view relies on the *global transition relation* of $\mathcal{A}$, which, in turn, defers to the notion of a *configuration*. Let us be more precise and consider an MPA $\mathcal{A} = ((\mathcal{A}_p)_{p \in Proc}, \mathcal{D}, \overline{s}^{in}, F)$, $\mathcal{A}_p = (S_p, \Delta_p)$. The set of *configurations* of $\mathcal{A}$, denoted by $Conf_{\mathcal{A}}$, is the cartesian product $S_{\mathcal{A}} \times \mathcal{C}_{\mathcal{A}}$ where $\mathcal{C}_{\mathcal{A}} := \{\chi \mid \chi : Ch \to \mathcal{D}^*\}$ is the set of possible *channel contents* of $\mathcal{A}$. Now, the *global transition relation* of $\mathcal{A}$, $\Longrightarrow_{\mathcal{A}} \subseteq Conf_{\mathcal{A}} \times \mathcal{C} \times \mathcal{D} \times Conf_{\mathcal{A}}$, is defined as follows:

- $((\overline{s}, \chi), \mathtt{send}(p, q), m, (\overline{s}', \chi')) \in \Longrightarrow_{\mathcal{A}}$ if $(\overline{s}[p], \mathtt{send}(p, q), m, \overline{s}'[p]) \in \Delta_p$, $\chi' = \chi[(p, q)/m \cdot \chi((p, q))]$ (i.e., χ' maps (p, q) to $m \cdot \chi((p, q))$ and, otherwise, coincides with χ), and for all $r \in Proc \setminus \{p\}$, $\overline{s}[r] = \overline{s}'[r]$.
- $((\overline{s}, \chi), \mathtt{rec}(p, q), m, (\overline{s}', \chi')) \in \Longrightarrow_{\mathcal{A}}$ if there is a word $w \in \mathcal{D}^*$ such that $(\overline{s}[p], \mathtt{rec}(p, q), m, \overline{s}'[p]) \in \Delta_p$, $\chi((q, p)) = w \cdot m$, $\chi' = \chi[(q, p)/w]$, and for all $r \in Proc \setminus \{p\}$, $\overline{s}[r] = \overline{s}'[r]$.

Let $\chi_\varepsilon : Ch \to \mathcal{D}^*$ map each channel onto the empty word. When we set $(\overline{s}^{in}, \chi_\varepsilon)$ to be the *initial configuration* and $F \times \{\chi_\varepsilon\}$ to be the set of *final configurations*, $\mathcal{A}$ defines in the canonical way an MSC word language $L_w(\mathcal{A}) \subseteq \mathcal{C}^*$. The corresponding MSC language will be denoted by $L(\mathcal{A})$ and is called the *language*

of $\mathcal{A}$. Given a class $\mathfrak{C}$ of MPAs, let furthermore $\mathcal{L}(\mathfrak{C}) := \{L \subseteq \mathbb{MSC} \mid \text{there is } \mathcal{A} \in \mathfrak{C} \text{ such that } L = L(\mathcal{A})\}$ denote the *class of languages* of $\mathfrak{C}$.

For configurations $(\overline{s}, \chi), (\overline{s}', \chi') \in \mathit{Conf}_{\mathcal{A}}$, we write $(\overline{s}, \chi) \Longrightarrow_{\mathcal{A}} (\overline{s}', \chi')$ if $((\overline{s}, \chi), \sigma, m, (\overline{s}', \chi')) \in \Longrightarrow_{\mathcal{A}}$ for some $\sigma \in \mathcal{C}$ and $m \in \mathcal{D}$. We call a configuration $(\overline{s}, \chi) \in \mathit{Conf}_{\mathcal{A}}$ *reachable* from another configuration $(\overline{s}', \chi') \in \mathit{Conf}_{\mathcal{A}}$ if $(\overline{s}', \chi') \Longrightarrow^*_{\mathcal{A}} (\overline{s}, \chi)$. Moreover, we say $(\overline{s}, \chi) \in \mathit{Conf}_{\mathcal{A}}$ is *productive* if there is a final configuration that is reachable from $(\overline{s}, \chi)$.

For $B \geq 1$, an MPA $\mathcal{A}$ is called *B-bounded* if, for any $(p, q) \in \mathit{Ch}$ and any configuration $(\overline{s}, \chi)$ that is productive and reachable from the initial configuration, it holds $|\chi((p, q))| \leq B$. According to [HMKT00], who use a slightly different notion of bounded MPAs, we call an MPA $\mathcal{A}$ *strongly-B-bounded* for some $B \geq 1$ if, for any $(p, q) \in \mathit{Ch}$ and any configuration $(\overline{s}, \chi)$ that is reachable from the initial configuration, $|\chi((p, q))| \leq B$. Furthermore, $\mathcal{A}$ is called *(strongly) bounded* if it is B-bounded (strongly-B-bounded, respectively) for some $B \geq 1$. Given a class $\mathfrak{C}$ of MPAs, let b$\mathfrak{C}$ (sb$\mathfrak{C}$) denote the set of (strongly, respectively) bounded MPAs $\mathcal{A}$ with $\mathcal{A} \in \mathfrak{C}$.

Sometimes, it is more convenient to consider MPAs with a set of global initial states instead of one global initial state. So let an *extended MPA* be an MPA $\mathcal{A} = ((\mathcal{A}_p)_{p \in \mathit{Proc}}, \mathcal{D}, S^{in}, F)$ where, though, $S^{in} \subseteq \prod_{p \in \mathit{Proc}} S_p$ is a finite *set of global initial states*. The language of $\mathcal{A}$ is defined analogously to the MPA case.

Lemma 3. *Let $N \geq 1$ and L be an MSC language. Then L is the language of a (bounded/finite/bounded and finite) N-MPA iff it is the language of an extended locally-accepting (bounded/finite/bounded and finite, respectively) N-MPA.*

Proof. "only if": Let $\mathcal{A} = ((\mathcal{A}_p)_{p \in \mathit{Proc}}, \mathcal{D}, \overline{s}^{in}, F)$, $\mathcal{A}_p = (S_p, \Delta_p)$, be an MPA. For each state $\overline{s} \in F$, introduce a global initial state running a distinct copy $\mathcal{A}(\overline{s})$ of $\mathcal{A}$ with local state spaces $S_p^{\overline{s}}$ (in the following, a copy of a local state $s \in S_p$ in $\mathcal{A}(\overline{s})$ is denoted by $s^{\overline{s}}$). The set of global final states is henceforth the cartesian product $\prod_{p \in \mathit{Proc}} \bigcup_{\overline{s} \in F} \{\overline{s}[p]^{\overline{s}}\}$. The resulting MPA is locally-accepting and, obviously, recognizes the same language as $\mathcal{A}$ without having affected the number of messages, boundedness, or finiteness properties.

"if": Let $\mathcal{A} = ((\mathcal{A}_p)_{p \in \mathit{Proc}}, \mathcal{D}, S^{in}, F)$, $\mathcal{A}_p = (S_p, \Delta_p)$, be an extended MPA where F is the product $\prod_{p \in \mathit{Proc}} F_p$ of sets $F_p \subseteq S_p$. The basic idea is to create a copy $S_p^{\overline{s}_0} = S_p \times \{\overline{s}_0\}$ of S_p for any global initial state $\overline{s}_0 \in S^{in}$. Starting in some new global initial state $\overline{s}^{in}$ and switching to some state $(s, \overline{s}_0)$ now settles for simulating a run of $\mathcal{A}$ from $\overline{s}_0$ by henceforth allowing to enter no other copy than $S_p^{\overline{s}_0}$. In a global final state, it is then checked whether the other processes agree in their choice of $\overline{s}_0$. More formally, we may have local transitions $((s, \overline{s}_0), \sigma, m, (s', \overline{s}_0))$ with $\overline{s}_0 \in S^{in}$ if (s, σ, m, s') is a local transition of $\mathcal{A}$. Moreover, we add kind of initial transitions $(\overline{s}^{in}[p], \sigma, m, (s, \overline{s}_0))$ if $(\overline{s}_0[p], \sigma, m, s)$ is some p-local transition of $\mathcal{A}$ with $\overline{s}_0 \in S^{in}$. It remains to reformulate the acceptance condition: $\overline{s}$ is a global final state if there is $\overline{s}_0 \in S^{in}$ such that, for any $p \in \mathit{Proc}$, either $\overline{s}[p] = \overline{s}^{in}[p]$ and $\overline{s}_0[p] \in F_p$ or $\overline{s}[p] \in F_p \times \{\overline{s}_0\}$. □

Lemma 4 ([AEY00]). $\mathcal{P}^0 = \mathcal{L}(1\text{-MPA}_\ell)$

Corollary 3. $\mathcal{P} = \mathcal{L}(\text{1-MPA})$

Proof. "$\supseteq$": According to Lemma 3, a 1-MPA can be transformed into an equivalent extended locally-accepting 1-MPA $((\mathcal{A}_p)_{p\in Proc}, \mathcal{D}, S^{in}, F)$, which then recognizes $\bigcup_{\overline{s}\in S^{in}} L(((\mathcal{A}_p)_{p\in Proc}, \mathcal{D}, \overline{s}, F))$. The assertion follows from Lemma 4 and Definition 2.

"$\subseteq$": Similarly, any MSC language $L \in \mathcal{P}$ is the union of finitely many languages $L_1, \ldots, L_k \in \mathcal{P}^0$, which, according to Lemma 4 are recognized by locally-accepting 1-MPAs $\mathcal{A}^1, \ldots, \mathcal{A}^k$ (each employing, say, a as synchronization message) with global initial states $\overline{s}_1, \ldots, \overline{s}_k$ and sets of global final states $F^1, \ldots, F^k$, respectively, where, for each $i \in \{1, \ldots, k\}$, $F^i = \prod_{p\in Proc} F^i_p$ for some $F^i_p \subseteq S^i_p$ (let hereby S^i_p be the set of p-local states of $\mathcal{A}^i$). Without loss of generality, $\mathcal{A}^1, \ldots, \mathcal{A}^k$ have mutually distinct local state spaces. The extended locally-accepting 1-MPA recognizing L processwise merges the state spaces and transitions of $\mathcal{A}^1, \ldots, \mathcal{A}^k$, employs $\{\overline{s}_1, \ldots, \overline{s}_k\}$ being the set of global initial states, and, similarly to the proof of Lemma 3, sets the set of global final states to be $\prod_{p\in Proc} \bigcup_{i\in\{1,\ldots,k\}} F^i_p$. The assertion then follows from Lemma 3. □

Lemma 5. $\mathcal{RP}^0 = \mathcal{L}(\text{1-bMPA}^f_\ell)$

Proof. "$\supseteq$": This direction directly follows from Lemma 4 and Lemma 7 below.

"$\subseteq$": Let $L \in \mathcal{RP}^0$ and, for $p \in Proc$, $\mathcal{A}_p = (S_p, \Delta_p, \overline{s}^{in}_p, F_p)$ be a finite automaton over $\mathcal{C}_p$ satisfying $L(\mathcal{A}_p) = L \restriction p := \{M \restriction p \mid M \in L\}$. Consider the MPA $\mathcal{A} = ((\mathcal{A}'_p)_{p\in Proc}, \mathcal{D}, \overline{s}^{in}, F)$ with $\mathcal{D} = \{\mathrm{a}\}$, $\overline{s}^{in} = (\overline{s}^{in}_p)_{p\in Proc}$, $F = \prod_{p\in Proc} F_p$, and $\mathcal{A}'_p = (S_p, \Delta'_p)$ where, for any $s, s' \in S_p$ and $\sigma \in \mathcal{C}_p$, $(s, \sigma, \mathrm{a}, s') \in \Delta'_p$ if $(s, \sigma, s') \in \Delta_p$. We claim that both $\mathcal{A} \in \text{1-bMPA}^f_\ell$ and $L(\mathcal{A}) = L$. First, it is easy to see that $L \subseteq L(\mathcal{A})$. Now assume an MSC M to be contained in $L(\mathcal{A})$. For each $p \in Proc$, $M \restriction p \in L(\mathcal{A}_p) = L \restriction p$ so that there is an MSC $M' \in L$ with $M' \restriction p = M \restriction p$. From the definition of $\mathcal{P}^0$, it then immediately follows that M is contained in L, too. Clearly, $\mathcal{A}$ is finite, locally-accepting, and bounded. □

Lemma 6. $\mathcal{RP}^0 \supsetneq \mathcal{L}(\text{1-sbMPA}^f_\ell)$

Proof. It remains to show strictness. Let $L = \{M_1\}^* \cup \{M_2\}^*$ with M_1 and M_2 given by Figure 6, and suppose there is an MPA $\mathcal{A} \in \text{1-MPA}^f_\ell$ with $L(\mathcal{A}) = L$. Then, for each natural $n \geq 1$, the word

$$\mathtt{send}(1,2)^2 \left(\mathtt{send}(3,1)\ \mathtt{rec}(1,2)\ \mathtt{send}(1,2)^2\ \mathtt{rec}(2,1)\ \mathtt{send}(2,3)\ \mathtt{rec}(3,2)\right)^n$$

from $\mathcal{C}^*$ leads from the initial configuration of $\mathcal{A}$ via $\Longrightarrow_{\mathcal{A}}$ to some configuration (s, χ) with $\chi((1,2)) = n+3$. Thus, $\mathcal{A}$ cannot be strongly-bounded. Nevertheless, L is contained in $\mathcal{RP}^0$ and 2-bounded. □

Corollary 4. $\mathcal{RP} = \mathcal{L}(\text{1-bMPA}^f)$

Lemma 7 ([HMKT00]). $\mathcal{R} = \mathcal{L}(\text{bMPA}^f) = \mathcal{L}(\text{sbMPA}^f)$

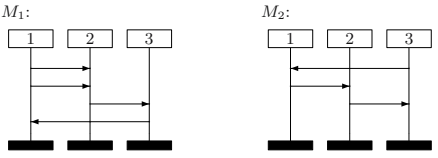


Fig. 6. Universal boundedness vs. strong boundedness

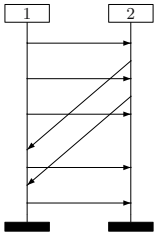


Fig. 7. $M(3,2)$

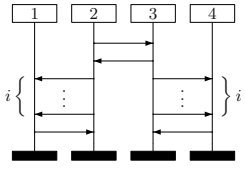


Fig. 8. MSC $M(i)$

In [BL04], it has been shown that any EMSO-definable MSC language is implementable as a finite MPA and vice versa.

Lemma 8 ([BL04]). $\mathcal{E} = \mathcal{L}(\text{MPA}^f)$

Lemma 9. *We have the following strict inclusion:*

(a) $\mathcal{L}(1\text{-MPA}^f_\ell) \subsetneq \mathcal{EP}^0$
(b) $\mathcal{L}(1\text{-MPA}^f) \subsetneq \mathcal{EP}$

Proof. Inclusion of (a) follows from Lemma 4 and Lemma 8. Inclusion of (b) then proceeds as the proof for Corollary 3. Let us turn towards strictness. For naturals $m, n \geq 1$, let the MSC $M(m,n)$ be given by its projections according to $M(m,n) \restriction 1 = \mathtt{send}(1,2)^m\,(\mathtt{rec}(1,2)\ \mathtt{send}(1,2))^n$ and $M(m,n) \restriction 2 = (\mathtt{rec}(2,1)\ \mathtt{send}(2,1))^n\,\mathtt{rec}(2,1)^m$. The MSC $M(3,2)$ is depicted in Figure 7. Now consider the EMSO-definable MSC language $L = \{M(n,n) \mid n \geq 1\}$, which is recognized by the finite locally-accepting 2-MPA from Figure 5. We easily verify that L is a weak product MSC language. However, L is not contained in $\mathcal{L}(1\text{-MPA}^f)$. Because suppose there is $\mathcal{A} = ((\mathcal{A}_p)_{p\in\{1,2\}}, \mathcal{D}, \overline{s}^{in}, F) \in 1\text{-MPA}^f$ with $L(\mathcal{A}) = L$. As $\mathcal{A}$ is finite, there is $n \geq 1$ and an accepting run of $\mathcal{A}$ on $M(n,n)$ such that $\mathcal{A}_1$, when reading the first n letters $\mathtt{send}(1,2)$ of $M(n,n) \restriction 1$, goes through a cycle, say of length i (≥ 1), and $\mathcal{A}_2$, when reading the last n letters $\mathtt{rec}(2,1)$ of $M(n,n) \restriction 2$, goes through another cycle, say of length j (≥ 1). But then there is also an accepting run of $\mathcal{A}$ on $M(n + (i \cdot j), n) \notin L$. □

Lemma 10. *For each* $N \geq 1$, $\mathcal{L}((N+1)\text{-bMPA}^f_\ell) \setminus \mathcal{L}(N\text{-MPA}) \neq \emptyset$.

For $N \geq 1$, consider the MSC language $L_{N+1} = \{M(i) \mid i \in \{1, \ldots, N^2 + 1\}\}^*$ where $M(i)$ is depicted in Figure 8. Though L_{N+1} is realizable by means of $N+1$ synchronization messages, N messages turn out to be insufficient.

Lemma 11. $\mathcal{L}(\text{1-bMPA}^f) \setminus \mathcal{L}(\text{MPA}_\ell) \neq \emptyset$

Proof. Let L^f consist of the MSCs M_1 and M_2 given by Figure 1. Then L^f is contained in $\mathcal{L}(\text{1-bMPA}^f) \setminus \mathcal{L}(\text{MPA}_\ell)$.In contrast, a bounded finite 1-MPA recognizing L^f has some global knowledge employing global final states. □

Theorem 2. *The classes of MSC languages proposed in Sections 3 and 4 draw the picture given by Figure 3.*

We did not pay special attention to the relation between (weak) EMSO-definable product languages and the classes of languages defined by (locally-accepting) finite N-MPAs for $N \geq 2$, which is indicated by the light-gray line in Figure 3. We believe that it is possible to show incomparability respectively witnessed by a language depending on N and similar to the suggested one.

References

[AEY00] R. Alur, K. Etessami, and M. Yannakakis. Inference of Message Sequence Charts. In *Proceedings of the 22nd International Conference on Software Engineering*. ACM, 2000.

[BL04] B. Bollig and M. Leucker. Message-Passing Automata are expressively equivalent to EMSO Logic. In *Proceedings of CONCUR 2004*, volume 3170 of *LNCS*. Springer, 2004.

[GMK04] B. Genest, A. Muscholl, and D. Kuske. A Kleene theorem for a class of communicating automata with effective algorithms. In *Proceedings of DLT 2004*, volume 3340 of *LNCS*. Springer, 2004.

[GMSZ02] B. Genest, A. Muscholl, H. Seidl, and M. Zeitoun. Infinite-state high-level MSCs: Model-checking and realizability. In *Proceedings of ICALP 2002*, volume 2380 of *LNCS*. Springer, 2002.

[HMK+04] J. G. Henriksen, M. Mukund, K. Narayan Kumar, M. Sohoni, and P. S. Thiagarajan. A theory of regular MSC languages. *Information and Computation*, 2004. to appear.

[HMKT00] J. G. Henriksen, M. Mukund, K. Narayan Kumar, and P. S. Thiagarajan. Regular collections of message sequence charts. In *Proceedings of MFCS 2000*, volume 1893 of *LNCS*. Springer, 2000.

[ITU99] ITU-TS Recommendation Z.120: Message Sequence Chart 1999 (MSC99), 1999.

[Kus02] D. Kuske. A further step towards a theory of regular MSC languages. In *Proceedings of STACS 2002*, volume 2285 of *LNCS*. Springer, 2002.

[Kus03] D. Kuske. Regular Sets of Infinite Message Sequence Charts. *Information and Computation*, 187:80–109, 2003.

[Mor02] R. Morin. Recognizable sets of message sequence charts. In *Proceedings of STACS 2002*, volume 2285 of *LNCS*. Springer, 2002.

[Thi95] P. S. Thiagarajan. A trace consistent subset of PTL. In *Proceedings of CONCUR 1995*, volume 962 of *LNCS*. Springer, 1995.