RWTH Aachen

Department of Computer Science
Technical Report

Generalised Regular MSC Languages

Benedikt Bollig, Martin Leucker, and Thomas Noll

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2002-03

RWTH Aachen · Department of Computer Science · January 2002

Generalised Regular MSC Languages

Benedikt Bollig[1], Martin Leucker[2], and Thomas Noll[1]

[1] Lehrstuhl für Informatik II, Aachen University of Technology (RWTH), Germany
`{bollig,noll}@informatik.rwth-aachen.de`
[2] Dept. of Computer and Information Science***, University of Pennsylvania, USA
`leucker@cis.upenn.edu`

Abstract. In this paper, we establish the concept of regularity for languages consisting of Message Sequence Charts (MSCs). To this aim, we formalise their behaviour by string languages and give a natural definition of regularity in terms of an appropriate Nerode right congruence. Moreover, we present a class of accepting automata and, using this characterisation, establish several decidability and closure properties of MSC languages. We also provide a logical characterisation by a monadic second-order logic interpreted over MSCs. In contrast to existing work on regular MSC languages, our approach is neither restricted to a certain class of MSCs nor tailored to a fixed communication medium (such as a FIFO channel). It explicitly allows MSCs with message overtaking and is thus applicable to a broad range of channel types like mixtures of stacks and FIFOs.

1 Introduction

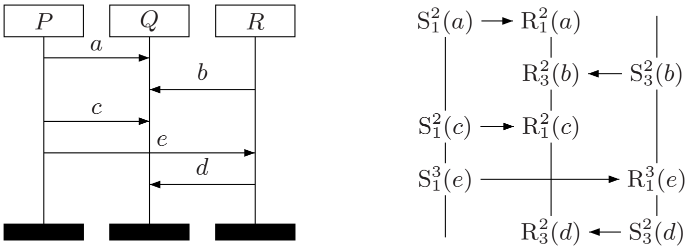
Components of distributed systems usually communicate with each other via message passing: A sender process sends a message over a channel, from which it is taken by the receiver process. A prominent formalism to model this kind of systems is that of *Message Sequence Charts* (MSCs) [9, 10]. They are standardised, can be denoted both textually and graphically, and are often employed in industry. Furthermore, they are quite similar to the notion of sequence charts of the Unified Modelling Language (UML) [2].

An MSC 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. A labelled arrow from one line to another corresponds to the communication event of sending the labelling value from the first process to the second. As the vertical lines are interpreted as time axes, there is the general rule that arrows must not go "upwards", because this would describe a situation that a message is received before it has been sent. Figure 1(a) gives an example of an MSC. Collections of MSCs are used to capture the scenarios that a designer might want the system to follow or to avoid.

When one considers the dynamic behaviour of an MSC, i.e., the sequences of actions that may be observed when the system is executed, one distinguishes between the so-called visual-order semantics and the causal-order semantics. The *visual order* assumes that the events are ordered as shown in the MSC. That is,

---

[$\star\star\star$] Most of the work was completed during the author's employment at Lehrstuhl für Informatik II, Aachen University of Technology, Germany.



(a) (b)

Fig. 1. An MSC and its formalisation

the events on a single process line are linearly ordered, and sending events precede their corresponding receiving events. For example, Process Q in Figure 1(a) has to read the a symbol before it can read b. In the *causal order-based semantics*, a concrete communication medium between the processes is taken into account, e.g., a first-in-first-out (FIFO) channel. Furthermore, receiving events on the same process line are not ordered unless they are "causality dependent". For instance, reading event b may occur before reading a: As Process P might have sent a after R has sent b and assuming a single FIFO incoming channel for Q, Q will potentially receive b before a. Note that, under the same communication assumption, reading c must occur before reading d. To simplify our presentation, we adopt the visual-order point of view in the following. But we would like to stress that—with minor modifications—our very general approach also works wrt. the causal order.

Given the system specification in the form of a collection of MSCs, one is interested in doing formal analysis to discover errors at the early stages of system design. Of course, the first question arising is which kinds of collections of MSCs are amenable to formal methods. In a pioneering work by Henriksen et al. [8], a definition of *regularity* of MSC languages is proposed. A characterisation in terms of message-passing automata and in terms of monadic second-order logic is also given. The paper explains in a convincing way the benefits of these alternative descriptions, arguing that this is the "right" notion of regularity for MSCs. For example, a characterisation in terms of finite devices (automata) gives evidence for a collection of MSCs to be *realisable*.

However, this approach has a serious limitation. So-called "MSCs with message overtaking" cannot be considered. But these are explicitly defined in the official standard [9] and must be taken into account. The limitation stems from the fact that, for establishing a link between MSCs and classical language theory, the graphical representation of an MSC has somehow to be mapped to the domain of strings. The straightforward approach, enumerating the possible linearisations of the events that occur in an MSC, only works for simple types of MSCs where the correspondence between a sending event and its receiving counterpart can be derived from the order in which they occur in the string. Note that also [1] has to restrict the admissible class of MSCs in order to be able to relate MSCs and string languages.

Our solution to this problem is to associate with every communication event in the string representation of an MSC a natural number that explicitly establishes this correspondence. As it will become clear in the next section, this allows us to drop any restriction on the set of MSCs under consideration. The price to pay is that, for *arbitrary* collections of MSCs, we have to work with strings, automata, etc. over *infinite alphabets.* For practical applications though, the "simple" collections of MSCs are of interest. Therefore, within the domain of (MSC word) languages, we will spot the *regular* ones. These are defined in terms of a Nerode right congruence, which allows a straightforward generalisation to languages over infinite alphabets.

To support formal analysis, we introduce a new kind of automaton (MFA) accepting linearisations of MSCs. More precisely, our notion of MFAs guarantees that every accepted word is indeed a linearisation of an MSC. Moreover, we establish several closure properties and decidability results. In particular, we show that language inclusion is decidable, a crucial property for *model-checking* applications. Our concept of automata is similar to the one introduced by Kaminski and Francez [11]. Note, however, that in their setting the problem of language inclusion is undecidable [15]. Furthermore, our framework is well suited for extensions. In [7], compositional message sequence graphs (CMSGs) are introduced to describe larger classes of MSCs. Our automata model MFA is well prepared to accept languages of CMSGs, which can be characterised by MSC languages with a regular set of *representative linearisations*, a concept defined and studied by Madhusudan and Meenakshi [13]. However, due to lack of space, this topic will be discussed elsewhere.

Subsequently, we follow the line of [8] and develop an alternative automata-theoretic characterisation based on message-passing automata as well as a description in terms of monadic second-order logic. Note that, although the results are similar, the proofs are of a different nature because it is generally impossible to lift proofs directly from the setting of languages over finite alphabets to the infinite case.

The main contribution of the paper is to develop a theory of regular collections of MSCs in terms of Nerode right congruences, finite automata, message-passing automata, and models of MSO formulas for the full class of MSCs. Thus, we provide the formal basis for subsequent verification questions. Note that our approach has already turned out to be useful in the setting of LTL model checking for MSCs [3].

This paper is a revised version of the Technical Report [4].

2 Message Sequence Charts and Their Linearisations

In this section, we present our formal model for MSCs and establish a string representation, which describes their behaviour in a linear way.

2.1 Message Sequence Charts

For $N \geq 2$, let $\mathcal{P}_N := \{1, \ldots, N\}$ be a set of *processes* and Λ a finite *message alphabet*. Let further $\Sigma_{\mathcal{S}} := \{\mathrm{S}_p^q(\lambda) \mid p, q \in \mathcal{P}_N,\ p \neq q,\ \lambda \in \Lambda\}$ and $\Sigma_{\mathcal{R}} := \{\mathrm{R}_p^q(\lambda) \mid p, q \in \mathcal{P}_N,\ p \neq q,\ \lambda \in \Lambda\}$ denote the sets of *send* and *receive actions*, respectively, and $\Sigma := \Sigma_{\mathcal{S}} \cup \Sigma_{\mathcal{R}}$ their union. An action $\mathrm{S}_p^q(\lambda)$ stands for sending a message λ from Process p to Process q, and $\mathrm{R}_p^q(\lambda)$ represents the corresponding receive action, which is then executed by Process q. In this sense, $\mathit{Corr} := \{(\mathrm{S}_p^q(\lambda), \mathrm{R}_p^q(\lambda)) \mid p, q \in \mathcal{P}_N,\ p \neq q,\ \lambda \in \Lambda\}$ relates those actions that belong together. From now on, all premises and definitions are made wrt. a fixed set $\mathcal{P}_N$ of processes and a fixed message alphabet Λ.

An MSC is a tuple of the form $M = (\{E_p\}_{p \in \mathcal{P}_N}, \{\preceq_p\}_{p \in \mathcal{P}_N}, f, L)$ where $\{E_p\}_{p \in \mathcal{P}_N}$ is a family of pairwise disjoint finite sets of *events*, each of which is totally ordered by a relation $\preceq_p \subseteq E_p \times E_p$. (For simplicity, we consider $\preceq_p$ as a relation over $E := \bigcup_{p \in \mathcal{P}_N} E_p$, the set of all events.) Let $P : E \cup \Sigma \to \mathcal{P}_N$ yield the process an event or an action belongs to, i.e., $P(e) = p$ for any $e \in E_p$, $P(\mathrm{S}_p^q(\lambda)) = p$, and $P(\mathrm{R}_p^q(\lambda)) = q$. M is required to induce a partition $E = S \cup R$ of the events into send (S) and receive events (R) such that $f : S \to R$ is a bijective mapping satisfying the following:

- The *visual order* $\preceq \subseteq E \times E$ of M, i.e., the reflexive and transitive closure of $\bigcup_{p \in \mathcal{P}_N} \preceq_p \cup \{(e, f(e)) \mid e \in S\}$, is a partial order; in particular, it is antisymmetric.
- $L : E \to \Sigma$ provides information about the messages being interchanged by communicating events whereby, for all $e \in S$, there is some $\lambda \in \Lambda$ such that

$$L(e) = \mathrm{S}_{P(e)}^{P(f(e))}(\lambda) \text{ and } L(f(e)) = \mathrm{R}_{P(e)}^{P(f(e))}(\lambda).$$

For example, Figure 1(b) presents a formal version of the MSC shown in Figure 1(a).

A partial execution (configuration) of an MSC can be described by a downwards closed subset of events, containing those events that occurred so far. Formally, given an MSC $M = (\{E_p\}_{p \in \mathcal{P}_N}, \{\preceq_p\}_{p \in \mathcal{P}_N}, f, L)$, a *configuration* of M is a subset E' of E satisfying $E' = {\downarrow}E' := \{e \in E \mid \exists e' \in E' : e \preceq e'\}$. Let $\mathit{Conf}(M)$ denote the set of configurations of M. The execution of M can be described by a transition relation $\longrightarrow_M \subseteq \mathit{Conf}(M) \times \Sigma \times \mathit{Conf}(M)$ where $c \xrightarrow{\sigma}_M c'$ iff there exists $e \in E - c$ such that $L(e) = \sigma$ and $c' = c \cup \{e\}$.

2.2 MSC Words

A suitable notion of regularity for a class of objects should have similarities with existing notions for regular sets of objects. We will therefore reduce regularity of collections of MSCs to regularity of word languages. Thus, we have to identify an MSC with a set of words, which will be called *linearisations* or *MSC words.* A linearisation represents a possible execution sequence of the events occurring in an MSC. To justify this view, it is necessary to guarantee that—up to isomorphism—from a set of linearisations a corresponding MSC can be unambiguously inferred and vice versa. We are then able to define an equivalence on MSC words whose equivalence classes on their own determine exactly one MSC and, as a whole, stand for the set of all MSCs.

So one of the main problems is how to define an MSC word. For example, $w = \mathrm{S}_1^2(a)\mathrm{S}_1^2(a)\mathrm{R}_1^2(a)\mathrm{R}_1^2(a) \in \Sigma^*$ might define the MSC M_1 given in Figure 2. But as w is also a correct linearisation of the MSC aside, we could likewise imagine that w represents M_2, relating the first and the fourth position of w. We therefore cannot unambiguously correlate a word in Σ^* with an MSC. Faced with causal-order semantics, the problem of relating events will be even more involved. In particular, if we make use of nondeterministic channels (which might allow MSCs to behave both in a FIFO manner and as a stack, for example), we need some information about which positions belong together. For this purpose, each position of a word $w \in \Sigma^*$ is equipped with a natural number indicating the matching positions (namely those showing the same number). The words $\alpha_1, \alpha_2 \in (\Sigma \times \mathbb{N})^*$ from Figure 3 are such MSC words. Notice that α_1 will determine the MSC M_1, whereas M_2 will emerge from α_2. To avoid these difficulties, [8] and [1] do not allow an MSC like M_2. However, M_2 is a perfect "MSC with message overtaking", which is explicitly allowed in the MSC standard [9, 10].

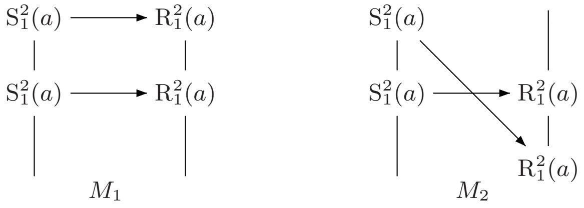


Fig. 2. MSCs generated by α_1 and α_2

We want to develop the respective theory step by step and first call a word $\alpha \in (\Sigma \times \mathbb{N})^*$

- *proper* iff for all $(\sigma, \tau) \in Corr$, $\pi \in \mathbb{N}$, and prefixes α' of α, $|\alpha'|_{(\tau,\pi)} \leq |\alpha'|_{(\sigma,\pi)} \leq |\alpha'|_{(\tau,\pi)} + 1$, and
- *complete* iff it is proper and for all $(\sigma, \tau) \in Corr$ and $\pi \in \mathbb{N}$, $|\alpha|_{(\sigma,\pi)} = |\alpha|_{(\tau,\pi)}$.

Thus, in a proper word every receiving event (we sometimes refer to positions of MSC words as events) must be preceded by a sending counterpart, and, for each number π and each send action, at most one "open" sending event is admitted.

Definition 1 (MSC Word). *A word* ${\sigma_1 \atop \pi_1} \ldots {\sigma_\ell \atop \pi_\ell} \in (\Sigma \times \mathbb{N})^*$ *is called an* MSC word *iff it is complete. Let* MW *denote the set of all MSC words and* PW *the set of proper words.*

To see some examples, look at the words $\alpha_1, \ldots, \alpha_4 \in (\Sigma \times \mathbb{N})^*$ given in Figure 3. As mentioned before, α_1 and α_2 are MSC words, whereas α_3 is certainly proper but not complete and α_4 is not even proper. We will refer to α_1 and α_2 as exemplary MSC words throughout the rest of the paper.

$$\alpha_1 = \begin{matrix} \mathrm{S}_1^2(a) & \mathrm{S}_1^2(a) & \mathrm{R}_1^2(a) & \mathrm{R}_1^2(a) \\ 1 & 3 & 1 & 3 \end{matrix} \qquad \alpha_2 = \begin{matrix} \mathrm{S}_1^2(a) & \mathrm{S}_1^2(a) & \mathrm{R}_1^2(a) & \mathrm{R}_1^2(a) \\ 1 & 2 & 2 & 1 \end{matrix}$$

$$\alpha_3 = \begin{matrix} \mathrm{S}_1^2(a) & \mathrm{S}_2^1(b) & \mathrm{R}_2^1(b) & \mathrm{S}_2^1(b) & \mathrm{R}_1^2(a) \\ 2 & 1 & 1 & 1 & 2 \end{matrix} \qquad \alpha_4 = \begin{matrix} \mathrm{S}_1^2(a) & \mathrm{S}_2^1(b) & \mathrm{R}_2^1(b) & \mathrm{S}_2^1(b) & \mathrm{S}_1^2(a) \\ 2 & 1 & 1 & 1 & 2 \end{matrix}$$

Fig. 3. Exemplary words

Given a proper word $\alpha = \genfrac{}{}{0pt}{}{\sigma_1 \ldots \sigma_\ell}{\pi_1 \ldots \pi_\ell} \in \mathrm{PW}$, we determine which positions are *matching*. For $i, j \in \{1, \ldots, \ell\}$, we write $i \searrow_\alpha j$ iff the following conditions hold:

- $i < j$,
- $(\sigma_i, \sigma_j) \in \mathit{Corr}$, and
- $j = \min\{k \mid k > i \text{ and } \pi_k = \pi_i \text{ and } (\sigma_i, \sigma_k) \in \mathit{Corr}\}$.

Referring to the previous example, the matching positions of α_1 and α_2 can be illustrated as in Figure 4, i.e., $1 \searrow_{\alpha_1} 3$ and $2 \searrow_{\alpha_1} 4$ as well as $1 \searrow_{\alpha_2} 4$ and $2 \searrow_{\alpha_2} 3$.

Fig. 4. Matching positions

2.3 From MSC Words to MSCs

Let us show that MSC words indeed represent MSCs. Falling back on the matching relation, a word $\alpha = \genfrac{}{}{0pt}{}{\sigma_1 \ldots \sigma_\ell}{\pi_1 \ldots \pi_\ell} \in \mathrm{MW}$ generates an MSC $M(\alpha) := (\{E_p\}_{p \in \mathcal{P}_N}, \{\preceq_p\}_{p \in \mathcal{P}_N}, f, L)$ where

- $E_p = \{n \in \{1, \ldots, \ell\} \mid P(\sigma_n) = p\}$,
  $S = \{n \in \{1, \ldots, \ell\} \mid \sigma_n \in \Sigma_{\mathcal{S}}\}$,
  $R = \{n \in \{1, \ldots, \ell\} \mid \sigma_n \in \Sigma_{\mathcal{R}}\}$,
- $n \preceq_p m$ iff $n, m \in E_p$ and $n \leq m$,
- $f(n) = m$ iff $n \searrow_\alpha m$, and
- $L(n) = \sigma_n$.

For example, α_1 generates the MSC M_1 illustrated in Figure 2, whereas α_2 generates M_2.

Moreover, there is no problem in extending the above definition to proper words, which then determine prefixes of MSCs.

Note that two different proper words can stand—up to isomorphism—for one and the same MSC or configuration of an MSC, respectively: Since the naturals are only used for identifying matching positions, we have some freedom in choosing the actual value. Furthermore, we are free to choose the linearisation of independent events. Therefore, we define two equivalence relations $\approx \subseteq \mathrm{PW} \times \mathrm{PW}$ and $\sim \subseteq \mathrm{MW} \times \mathrm{MW}$. The first identifies words with equivalent projections onto the second component; the latter, as introduced further below, allows to permute the positions of an MSC word.

Thus, for $\alpha = \genfrac{}{}{0pt}{}{\sigma_1}{\pi_1} \dots \genfrac{}{}{0pt}{}{\sigma_\ell}{\pi_\ell} \in \mathrm{PW}$ and $\beta = \genfrac{}{}{0pt}{}{\tau_1}{\rho_1} \dots \genfrac{}{}{0pt}{}{\tau_m}{\rho_m} \in \mathrm{PW}$, let $\alpha \approx \beta$ iff $\sigma_1 \dots \sigma_\ell = \tau_1 \dots \tau_m$ and for all $i, j \in \{1, \dots, \ell\}$, $i \searrow_\alpha j$ iff $i \searrow_\beta j$.

Remark 1. $\approx$ is an equivalence relation.

For instance, let α_1^n emerge from α_1 by replacing 3 in the natural-number component with some $n \in \mathbb{N}$. Then, $\alpha_1^n \in \mathrm{MW}$ iff $n \neq 1$, and $\alpha_1^n \in \mathrm{MW}$ implies $\alpha_1 \approx \alpha_1^n$. But notice that $\alpha_1 \not\approx \alpha_2$ because the second condition in the definition of $\approx$ is violated.

For a proper word $\alpha = \genfrac{}{}{0pt}{}{\sigma_1}{\pi_1} \dots \genfrac{}{}{0pt}{}{\sigma_\ell}{\pi_\ell} \in \mathrm{PW}$, let $\mathit{open}(\alpha) \subseteq \Sigma_{\mathcal{S}} \times \mathbb{N}$ denote the set of those send events that are not followed by a matching receive event, i.e., $\mathit{open}(\alpha) := \{(\sigma_i, \pi_i) \mid \sigma_i \in \Sigma_{\mathcal{S}}$ and there is no $j > i$ such that $i \searrow_\alpha j\}$. We call the elements of $\mathit{open}(\alpha)$ *open events*. A word $\alpha \in \mathrm{PW}$ is called in *normal form* iff for all prefixes $\genfrac{}{}{0pt}{}{\sigma_1}{\pi_1} \dots \genfrac{}{}{0pt}{}{\sigma_k}{\pi_k}$ of α, $\sigma_k \in \Sigma_{\mathcal{S}}$ implies $\pi_k = \min\{\pi \in \mathbb{N} \mid (\sigma_k, \pi) \notin \mathit{open}(\genfrac{}{}{0pt}{}{\sigma_1}{\pi_1} \dots \genfrac{}{}{0pt}{}{\sigma_{k-1}}{\pi_{k-1}})\}$. Thus, for every sending event, the lowest available number is chosen. Note that every equivalence class in $\mathrm{PW}/_{\approx}$ contains exactly one word in normal form. For $\alpha \in \mathrm{PW}$, let furthermore $\mathit{nf}(\alpha) = \beta$ iff $\alpha \approx \beta$ and β is in normal form. For instance, $\mathit{nf}(\alpha_1) = \alpha_1^2$, whereas α_2 is already in normal form so that $\mathit{nf}(\alpha_2) = \alpha_2$. nf is applied to sets of words in the expected manner.

In the following, we will not distinguish $\approx$-equivalent words.

Definition 2 (MSC Word Language). *A set* $\mathcal{L} \subseteq \mathrm{MW}$ *is called an* MSC word language *iff* $\mathcal{L} = \mathcal{L}^{\approx}$ *where* $\mathcal{L}^{\approx}$ *denotes the* $\approx$*-closure of* $\mathcal{L}$*.*

Note that, for any MSC word language $\mathcal{L}$, it holds $\mathcal{L} = \mathit{nf}(\mathcal{L})^{\approx}$.

Characterising regular languages within the scope of MSCs, a certain restriction of words and MSCs will prove to be important. Given a natural number B, $\alpha \in \mathrm{MW}$ is called *B-bounded* iff for all prefixes α' of α and actions $\sigma \in \Sigma_{\mathcal{S}}$, $|\mathit{open}(\alpha') \cap \{(\sigma, \pi) \mid \pi \in \mathbb{N}\}| \leq B$. This means that the number of open events is bounded by B for every send action. Examples for 2-bounded MSC words are α_1 and α_2. Note that we could likewise call α B-bounded iff for all prefixes α' of α, $|\mathit{open}(\alpha')| \leq B$, i.e., the total number of open send events is bounded by B, or also iff for all prefixes α' of α and $p \in \mathcal{P}_N$, $|\mathit{open}(\alpha') \cap \{(\sigma, \pi) \mid \pi \in \mathbb{N}, P(\sigma) = p\}| \leq B$, which means that the number of open events per process is bounded by B.

The definitions differ in the concrete bound, and the appropriate definition taken may vary depending on an underlying channel type. However, all presented results hold for every of these definitions.

2.4 Linearisations of MSCs

To finally relate MSCs to the rich theories of languages and automata over words, the concept of linearisations of an MSC is essential. We call an MSC word $\alpha = {\sigma_1 \atop \pi_1} \ldots {\sigma_\ell \atop \pi_\ell} \in (\Sigma \times \mathbb{N})^*$ a *linearisation* of an MSC $M = (\{E_i\}_{i \in \mathcal{P}_N}, \{\preceq_i\}_{i \in \mathcal{P}_N}, f, L)$ with a set of events $E = \{e_1, \ldots, e_\ell\}$ iff there are $c_1, \ldots, c_\ell \in \mathit{Conf}(M)$ with $\emptyset \xrightarrow{\sigma_1}_M c_1 \xrightarrow{\sigma_2}_M \cdots \xrightarrow{\sigma_\ell}_M c_\ell$ and there is a bijective mapping $\chi : E \to \{1, \ldots, \ell\}$ such that

- for all $e \in E$, $L(e) = \sigma_{\chi(e)}$, and
- for all $e \in S$, $e' \in R$, $f(e) = e'$ implies $\chi(e) \searrow_\alpha \chi(e')$.

$\mathit{Lin}(M)$ denotes the set of linearisations of M. For a set $\mathcal{M}$ of MSCs, we canonically define $\mathit{Lin}(\mathcal{M}) := \bigcup \{\mathit{Lin}(M) \mid M \in \mathcal{M}\}$. For instance, the exemplary word α_1 is a linearisation of the MSC M_1 shown in Figure 2, and α_2 is a linearisation of M_2. When, above, we spoke of isomorphism of two MSCs, we actually meant "inducing the same set of linearisations" instead.

An MSC is called B-bounded iff all of its linearisations are B-bounded. A collection of MSCs (a collection of MSC words, respectively) is B-bounded iff all members are B-bounded. Furthermore, we speak of *boundedness* in general iff we deal with B-boundedness for an arbitrary B.

We now turn towards $\sim \subseteq \mathrm{MW} \times \mathrm{MW}$, the second natural equivalence relation to study on linearisations of MSCs because it takes permutations of positions into account. For example, in Figure 1, it makes no real difference whether $\mathrm{S}_3^2(b)$ occurs before $\mathrm{R}_1^2(a)$ or after it. Given Σ, we define the *dependence relation* $D(\Sigma) \subseteq (\Sigma \times \mathbb{N})^2$ and write $(\sigma, \pi) D(\Sigma) (\sigma', \pi')$ iff

- $P(\sigma) = P(\sigma')$ or
- $(\sigma, \sigma') \in \mathit{Corr}$ and $\pi = \pi'$ or
- $(\sigma', \sigma) \in \mathit{Corr}$ and $\pi = \pi'$.

It turns out that the pair $(\Sigma \times \{1, \ldots, B\}, D(\Sigma) \cap (\Sigma \times \{1, \ldots, B\})^2)$ is a Mazurkiewicz trace alphabet [6] for every natural B—a fact which was already used in [12] providing a *direct* link between Mazurkiewicz traces and MSCs.

We then define the relation $\sim$ to be the least equivalence relation satisfying the following: If $\alpha = \beta_1 (\sigma, \pi)(\sigma', \pi') \beta_2$ and $\alpha' = \beta_1 (\sigma', \pi')(\sigma, \pi) \beta_2$ for suitable β_1, β_2 and not $(\sigma, \pi) D(\Sigma) (\sigma', \pi')$, then $\alpha \sim \alpha'$.

This section concludes with the following important properties of sets of linearisations that are induced by MSCs. In particular, they establish the expected connections between linearisations and the equivalence relations $\approx$ and $\sim$.

Theorem 1. *For an MSC M and $\alpha \in \mathit{Lin}(M)$, $\mathit{Lin}(M) = \mathit{Lin}(M(\alpha))$.*

Theorem 2. *For* $\alpha \in \mathrm{MW}$, $Lin(M(\alpha)) = [\alpha]_{(\approx\cup\sim)^*}$.

Theorem 1 and Theorem 2 can be shown by employing standard techniques taken, for example, from the theory of Mazurkiewicz traces. The proofs are left to the reader.

3 Regular MSC Word Languages and Their Automata

We already mentioned that the regularity of collections of MSCs will be defined in terms of regular MSC word languages. But as MSC words are defined over the infinite alphabet $\Sigma \times \mathbb{N}$, we have to modify the usual notion of regularity. In [11], a definition of regular word languages over infinite alphabets is proposed by providing an extended automata model that employs a finite transition relation but generates a behaviour catering for the fact that we deal with an infinite alphabet. However, important questions for these automata are undecidable. Thus, we follow a different approach. We first constitute an algebraic characterisation of regularity by means of a slightly adapted version of the Nerode right congruence, which allows a straightforward extension to infinite alphabets. Then, we establish its equivalence to an automata model that has similarities with the one described in [11] but is better suited for MSCs and provides desired properties.

3.1 Regular MSC Word Languages

Given an MSC word language $\mathcal{L}$, recall the definition of the Nerode right congruence $\equiv_{\mathcal{L}} \subseteq \mathrm{PW} \times \mathrm{PW}$:

$$\alpha \equiv_{\mathcal{L}} \beta \text{ iff } \forall \gamma \in (\Sigma \times \mathbb{N})^*.\alpha\gamma \in \mathcal{L} \text{ iff } \beta\gamma \in \mathcal{L}$$

As we want to identify $\approx$-equivalent words, we define $\approxeq_{\mathcal{L}} \subseteq \mathrm{PW} \times \mathrm{PW}$ as an extension of the Nerode right congruence by $\alpha \approxeq_{\mathcal{L}} \beta$ iff $nf(\alpha) \equiv_{\mathcal{L}} nf(\beta)$. Figure 5 illustrates the definition of $\approxeq_{\mathcal{L}}$.

$$\begin{array}{ccc} \alpha & \approxeq_{\mathcal{L}} & \beta \\ \approx & & \approx \\ nf(\alpha) & \equiv_{\mathcal{L}} & nf(\beta) \end{array}$$

Fig. 5. Extending the Nerode right congruence

Definition 3 (Regular MSC Word Language). *An MSC word language* $\mathcal{L}$ *is called* regular *iff* $\approxeq_{\mathcal{L}}$ *has finite index.*

The next characterisation of regular MSC word languages prepares for proving their correspondence with a certain class of finite automata, which we introduce further below.

Theorem 3. *Let $\mathcal{L}$ be an MSC word language. $\mathcal{L}$ is regular iff $\mathit{nf}(\mathcal{L})$ is a regular word language over $\Sigma \times Q$ for a finite subset Q of $\mathbb{N}$.*

Proof. ($\Longrightarrow$) Let $Q = \bigcup\{\{\pi_1, \ldots, \pi_\ell\} \mid \genfrac{}{}{0pt}{}{\sigma_1}{\pi_1} \cdots \genfrac{}{}{0pt}{}{\sigma_\ell}{\pi_\ell} \in \mathit{nf}(\mathcal{L})\}$. We show both that Q is finite and that $\equiv_{\mathit{nf}(\mathcal{L})}$ has finite index. Then $\mathit{nf}(\mathcal{L})$ is a regular word language.

We begin proving the finiteness of Q. Suppose Q is infinite. Then there is a family $\{\alpha_n\}_{n \in \mathbb{N}} \subseteq \mathit{nf}(\mathcal{L})$ such that (without loss of generality) n is the maximal number occurring in α_n and $\alpha_n = \genfrac{}{}{0pt}{}{\sigma_{n_1}}{\pi_{n_1}} \cdots \genfrac{}{}{0pt}{}{\sigma_{n_{k_n}}}{n} \cdots \genfrac{}{}{0pt}{}{\sigma_{n_{\ell_n}}}{\pi_{n_{\ell_n}}}$ where $\sigma_{n_{k_n}}$ is the last send action annotated with n. Let $\alpha'_n = \genfrac{}{}{0pt}{}{\sigma_{n_1}}{\pi_{n_1}} \cdots \genfrac{}{}{0pt}{}{\sigma_{n_{k_n}}}{n}$ (and hence $\mathit{nf}(\alpha'_n) = \alpha'_n$). As $\approxeq_{\mathcal{L}}$ has finite index, there are $i, j \in \mathbb{N}$, $i < j$, with $\alpha'_i \approxeq_{\mathcal{L}} \alpha'_j$. By definition, $\mathit{nf}(\alpha'_i) \equiv_{\mathcal{L}} \mathit{nf}(\alpha'_j)$ and consequently $\alpha'_i \equiv_{\mathcal{L}} \alpha'_j$. But for $\gamma = \genfrac{}{}{0pt}{}{\sigma_{i_{k_i+1}}}{\pi_{i_{k_i+1}}} \cdots \genfrac{}{}{0pt}{}{\sigma_{i_{\ell_i}}}{\pi_{i_{\ell_i}}}$, $\alpha'_i\gamma \in \mathcal{L}$ and $\alpha'_j\gamma \notin \mathcal{L}$, because j does not occur in γ which is necessary to make $\alpha_j\gamma$ an MSC word. So assuming that Q is infinite leads to a contradiction.

Let us then show that $\equiv_{\mathit{nf}(\mathcal{L})}$ has finite index, again by contradiction. Suppose that $\equiv_{\mathit{nf}(\mathcal{L})}$ has infinite index. Then there is a family $\{\alpha_n\}_{n \in \mathbb{N}} \subseteq \mathrm{PW}$ such that for all $n, m \in \mathbb{N}$ with $n \neq m$, $\alpha_n \not\equiv_{\mathit{nf}(\mathcal{L})} \alpha_m$, and, without loss of generality, α_n is in normal form. As $\approxeq_{\mathcal{L}}$ has finite index, there are $i, j \in \mathbb{N}$, $i \neq j$, satisfying $\alpha_i \approxeq_{\mathcal{L}} \alpha_j$ and hence $\mathit{nf}(\alpha_i) \equiv_{\mathcal{L}} \mathit{nf}(\alpha_j)$. Let $\gamma \in (\Sigma \times \mathbb{N})^*$ such that $\mathit{nf}(\alpha_i)\gamma \in \mathit{nf}(\mathcal{L})$. Then $\mathit{nf}(\alpha_j)\gamma$ must also be an MSC word, furthermore in normal form and therefore contained in $\mathit{nf}(\mathcal{L})$. (γ has to close the same send events in $\mathit{nf}(\alpha_i)$ as in $\mathit{nf}(\alpha_j)$. The "rest" of γ forms a proper and complete word in normal form so that $\mathit{nf}(\alpha_j)\gamma$ is in normal form.) We conclude $\mathit{nf}(\alpha_i) \equiv_{\mathit{nf}(\mathcal{L})} \mathit{nf}(\alpha_j)$ and hence $\alpha_i \equiv_{\mathit{nf}(\mathcal{L})} \alpha_j$ leading to a contradiction.

($\Longleftarrow$) Let us first slightly extend the definition of $\approx$ by writing $\alpha \approx \beta$ for suffixes α and β of MSC words iff there is a word $\alpha' \in \mathrm{PW}$ such that $\alpha'\alpha \approx \alpha'\beta$.

Let $\mathit{nf}(\mathcal{L}) \subseteq (\Sigma \times Q)^*$, $Q \subseteq \mathbb{N}$ finite, be a regular word language. Suppose $\approxeq_{\mathcal{L}}$ has infinite index. There exists a family $\{\alpha_n\}_{n \in \mathbb{N}} \subseteq \mathrm{PW}$ with $\alpha_n \not\approxeq_{\mathcal{L}} \alpha_m$ and hence $\mathit{nf}(\alpha_n) \not\equiv_{\mathcal{L}} \mathit{nf}(\alpha_m)$ for all $n, m \in \mathbb{N}$ with $n \neq m$. But as $\equiv_{\mathit{nf}(\mathcal{L})}$ has finite index, we can find $i, j \in \mathbb{N}$, $i \neq j$, satisfying $\mathit{nf}(\alpha_i) \equiv_{\mathit{nf}(\mathcal{L})} \mathit{nf}(\alpha_j)$. Let $\gamma \in (\Sigma \times \mathbb{N})^*$ such that $\mathit{nf}(\alpha_i)\gamma \in \mathcal{L}$. Then there exists a word γ', $\gamma' \approx \gamma$, with $\mathit{nf}(\alpha_i)\gamma' \in \mathit{nf}(\mathcal{L})$ and (due to $\mathit{nf}(\alpha_i) \equiv_{\mathit{nf}(\mathcal{L})} \mathit{nf}(\alpha_j)$) $\mathit{nf}(\alpha_j)\gamma' \in \mathit{nf}(\mathcal{L})$. As $\mathcal{L}$ is the $\approx$-closure of $\mathit{nf}(\mathcal{L})$, we finally get $\mathit{nf}(\alpha_j)\gamma \in \mathcal{L}$ and altogether $\mathit{nf}(\alpha_i) \equiv_{\mathcal{L}} \mathit{nf}(\alpha_j)$ resulting in a contradiction. □

Corollary 1. *Regular MSC word languages are bounded.*

The next theorem will be useful when, in Section 4, we consider $\sim$-closed MSC word languages.

Theorem 4. *Let $\mathcal{L}$ be a $\sim$-closed regular MSC word language. Then $\mathit{nf}(\mathcal{L})^\sim$ is a regular word language over a finite alphabet.*

Proof. To prove the theorem, we need to introduce some Mazurkiewicz trace theory. Given a (finite) alphabet A and a relation $D \subseteq A \times A$, we call the pair (A, D) a *dependence alphabet* iff D is reflexive and symmetric. (A, D) induces an equivalence relation $\sim_{(A,D)} \subseteq A^* \times A^*$ as follows: $w \sim_{(A,D)} w'$ iff there exists a sequence $w_1, \ldots, w_n \in A^*$ such that $w = w_1$, $w' = w_n$, and for all $i \in \{1, \ldots, n-1\}$, there are letters $z, z' \in A$ and words $w_{i_1}, w_{i_2} \in A^*$ with

- $w_i = w_{i_1} z z' w_{i_2}$,
- $w_{i+1} = w_{i_1} z' z w_{i_2}$, and
- $(z, z') \notin D$.

In other words, w and w' are equivalent iff w' can be obtained from w by finitely often permuting independent letters where two letters are called independent iff they are not dependent. A language $X \subseteq A^*$ is called a *regular Mazurkiewicz trace language* over (A, D) iff it is a regular word language and $\sim_{(A,D)}$-closed. We assume A to be totally ordered and $<$ to denote the corresponding lexicographic ordering on A^*. A word $w \in A^*$ is said to be in *lexicographic normal form* iff it is minimal in $[w]_{\sim_{(A,D)}}$ wrt. $<$. Furthermore, for $X \subseteq A^*$, $\mathit{Min}(X) := \{w \in X \mid w$ is in lexicographic normal form$\}$ denotes the set of minimal elements in X. Due to Theorem 4.6 and Corollary 4.6 in [6], for a regular word language $X \subseteq A^*$, $\mathit{Min}(X^{\sim_{(A,D)}}) \subseteq X$ implies that $X^{\sim_{(A,D)}}$ is a regular Mazurkiewicz trace language. Thus, for a regular set X containing at least the representatives in lexicographic normal form, the closure of X wrt. $\sim_{(A,D)}$ is a regular (Mazurkiewicz trace) language.

We now turn towards the proof. So let $\mathcal{L}$ be a $\sim$-closed regular MSC word language. As we already know from Theorem 3, there is a $B \in \mathbb{N}$ such that $\mathit{nf}(\mathcal{L})$ is a regular word language over $\Sigma \times \{1, \ldots, B\}$. We define $\widehat{\Sigma}$ to be the pair $(\Sigma \times \{1, \ldots, B\}, D(\Sigma) \cap (\Sigma \times \{1, \ldots, B\})^2)$ and easily verify that $\widehat{\Sigma}$ is a dependence alphabet satisfying $\sim_{\widehat{\Sigma}} = \sim \cap (\Sigma \times \{1, \ldots, B\})^2$. Let $\Sigma \times \{1, \ldots, B\}$ be totally ordered such that for all $(\sigma, \pi) \in \Sigma_{\mathcal{S}} \times \{1, \ldots, B\}$ and $(\sigma', \pi') \in \Sigma_{\mathcal{R}} \times \{1, \ldots, B\}$, (σ, π) is smaller than (σ', π'). We show that $\mathit{Min}(\mathit{nf}(\mathcal{L})^{\sim}) \subseteq \mathit{nf}(\mathcal{L})$ and, according to the previous remarks, conclude that $\mathit{nf}(\mathcal{L})^{\sim}$ is a regular (Mazurkiewicz trace) language. Suppose there is a non-minimal word $\alpha \in \mathit{nf}(\mathcal{L})$. We show that it is possible to transform α into a (lexicographically) minimal MSC word α' in normal form (wrt. $\approx$) such that $\alpha' \sim \alpha$. As $\mathcal{L}$ is $\sim$-closed, we conclude that $\alpha' \in \mathit{nf}(\mathcal{L})$ so that $\mathit{nf}(\mathcal{L})$ indeed contains the minimal representatives. Consider α: It may be of the form $\beta_1(\sigma, \pi)(\sigma', \pi')\beta_2$ for suitable β_1 and β_2 where, wrt. the underlying total ordering, (σ', π') is smaller than (σ, π). Consider the case that $\sigma, \sigma' \in \Sigma_{\mathcal{S}}$ and $((\sigma, \pi), (\sigma', \pi')) \notin D(\Sigma)$. Obviously, $\beta_1(\sigma', \pi')(\sigma, \pi)\beta_2$ is in normal form (wrt. $\approx$) as well but respecting the lexicographic ordering at the considered positions. The cases $\sigma, \sigma' \in \Sigma_{\mathcal{R}}$ and $(\sigma, \sigma') \in \Sigma_{\mathcal{R}} \times \Sigma_{\mathcal{S}}$ behave in the same manner, respectively. By successively applying such permutations, we can obtain the minimal word α' in $[\alpha]_{\sim}$. $\square$

3.2 MSC Finite-Memory Automata

We now present an automata model that characterises the class of regular MSC word languages. Our definition is inspired by [11] but modified to suit the requirements for MSCs and to allow stronger decidability results. Our model can be described as a finite automaton that makes use of a finite window whose positions occur in the labellings of the transitions—as well as elements of Σ—and indicate where to store a symbol of the infinite alphabet $\Sigma \times \mathbb{N}$ (concerning send actions) and where to take it from (concerning receive actions), respectively. Note that normal forms of regular MSC word languages could also be accepted by "standard" finite automata and we can use this fact to establish certain closure properties. However, not every finite automaton accepts normal forms of MSC words so that we do not get a precise automata-theoretic characterisation of regular MSC word languages which is the basis for a powerful algorithmic support of the theory on MSCs.

Definition 4 (MSC Finite-Memory Automaton). *An* MSC finite-memory automaton *(MFA) is a quintuple of the form* $\mathcal{A} = (S, r, \Delta, q_0, F)$ *where*

- S *is a nonempty finite set of* states*,*
- $r \geq 1$ *is a natural number called* window length*,*
- $\Delta \subseteq S \times (\Sigma \times \{1, \ldots, r\}) \times S$ *is the* transition relation*,*
- $q_0 \in S$ *is the* initial state*, and*
- $F \subseteq S$ *is the set of* final states.

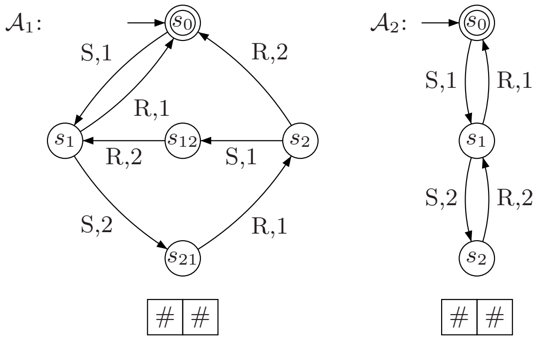


Fig. 6. Two MFAs

Figure 6 shows two MFAs, each with a window of length two. Let thereby S stand for $\mathrm{S}_1^2(a)$ and R for $\mathrm{R}_1^2(a)$.

Let $\mathcal{A}$ be an MFA as above. A configuration of $\mathcal{A}$ lists the current state and the current window entries, which are either numbered send events or empty (denoted by #). Thus, let $\mathit{Conf}_{\mathcal{A}} := S \times ((\Sigma_{\mathcal{S}} \times \mathbb{N}) \cup \{\#\})^r$ denote the (infinite) set of *configurations* of $\mathcal{A}$. We define a transition relation $\Longrightarrow_{\mathcal{A}} \subseteq \mathit{Conf}_{\mathcal{A}} \times (\Sigma \times \mathbb{N}) \times \mathit{Conf}_{\mathcal{A}}$ as follows:

- For $\sigma \in \Sigma_{\mathcal{S}}$, $(s, \mathbf{w}) \overset{(\sigma,\pi)}{\Longrightarrow}_{\mathcal{A}} (t, \mathbf{v})$ iff (σ, π) does not occur in $\mathbf{w}$ and there is a transition $(s, (\sigma, k), t) \in \Delta$ such that
  - $\mathbf{w}[k] = \#$,

  - $\mathbf{v}[k] = (\sigma, \pi)$, and
  - for each $l \neq k$, $\mathbf{w}[l] = \mathbf{v}[l]$.
- For $\sigma \in \Sigma_{\mathcal{R}}$, $(s, \mathbf{w}) \overset{(\sigma,\pi)}{\Longrightarrow}_{\mathcal{A}} (t, \mathbf{v})$ iff there is a transition $(s, (\sigma, k), t) \in \Delta$ such that
  - $\mathbf{w}[k] = (\tau, \pi)$ where $(\tau, \sigma) \in \mathit{Corr}$,
  - $\mathbf{v}[k] = \#$, and
  - for each $l \neq k$, $\mathbf{w}[l] = \mathbf{v}[l]$.

Thus, the meaning of a transition $(s, (\mathrm{S}_p^q(\lambda), k), t)$ is the following: If $\mathcal{A}$ is in state s, it is able to read an input symbol $(\mathrm{S}_p^q(\lambda), \pi)$, $\pi \in \mathbb{N}$, iff the kth position of its window is currently free and, furthermore, $(\mathrm{S}_p^q(\lambda), \pi)$ does not occur elsewhere in the window, i.e., there is no further open $(\mathrm{S}_p^q(\lambda), \pi)$-labelled send event. Taking the transition, the automaton stores $(\mathrm{S}_p^q(\lambda), \pi)$ in the kth position and enters state t. If, in contrast, the automaton reads an input symbol $(\mathrm{R}_p^q(\lambda), \pi)$, there has to be a transition $(s, (\mathrm{R}_p^q(\lambda), k), t)$ such that the kth position of the window currently shows the corresponding send symbol $(\mathrm{S}_p^q(\lambda), \pi)$. Replacing this symbol with $\#$, the automaton enters state t.

A *run* of $\mathcal{A}$ on a word $\frac{\sigma_1}{\pi_1} \cdots \frac{\sigma_\ell}{\pi_\ell} \in (\Sigma \times \mathbb{N})^*$ is a corresponding sequence $(s_0, \mathbf{w}_0)(s_1, \mathbf{w}_1) \ldots (s_\ell, \mathbf{w}_\ell)$ of configurations such that

- $s_0 = q_0$,
- $\mathbf{w}_0 = \#^r$, and
- for each $i \in \{1, \ldots, \ell\}$, $(s_{i-1}, \mathbf{w}_{i-1}) \overset{(\sigma_i,\pi_i)}{\Longrightarrow}_{\mathcal{A}} (s_i, \mathbf{w}_i)$.

The run is *accepting* iff $s_\ell \in F$ and $\mathbf{w}_\ell = \#^r$. $\mathcal{L}(\mathcal{A}) := \{\alpha \mid$ there is an accepting run of $\mathcal{A}$ on $\alpha\}$ forms the language defined by $\mathcal{A}$. We conclude that matching events in an accepted word use one and the same position of the window for their "agreement".

Due to the conditions we laid down for making transitions and accepting words, the automaton will accept MSC words only. A receive symbol has to be preceded by a corresponding send symbol, which, on its part, has to wait for the corresponding receive symbol before repeating the identical send symbol. Thus, we make sure that an accepted word is proper. Furthermore, recall that a run is accepting as soon as it ends in a final configuration featuring an empty window. In this way, completeness of accepted words is ensured. Moreover, the recognised language is $\approx$-closed because matching symbols can be read with—up to the MSC-word condition—arbitrary natural numbers. Notice that a regular MSC word language is not necessarily $\sim$-closed, a key feature allowing [13] to model CMSGs in terms of MSC word languages. We sum up these considerations as follows:

Proposition 1. *Given an MFA $\mathcal{A}$, $\mathcal{L}(\mathcal{A})$ is an MSC word language.*

For example, let us consider the MFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ illustrated by Figure 6 and behaving in a FIFO manner and as a stack, respectively. For the sake of clarity, let S stand for $\mathrm{S}_1^2(a)$ and R for $\mathrm{R}_1^2(a)$. Note that our MFAs permit only Process 1 to send and only Process 2 to receive a message a.

Recall our exemplary words α_1 and α_2. In fact, $\alpha_1 \in \mathcal{L}(\mathcal{A}_1)$ and $\alpha_2 \in \mathcal{L}(\mathcal{A}_2)$, but $\alpha_1 \notin \mathcal{L}(\mathcal{A}_2)$ and $\alpha_2 \notin \mathcal{L}(\mathcal{A}_1)$. An accepting run of $\mathcal{A}_1$ on α_1 first writes $(\mathrm{S}_1^2(a), 1)$ into the first position of the window and then $(\mathrm{S}_1^2(a), 3)$ into the second, whereupon the window is cleared in the same order, reading first $(\mathrm{R}_1^2(a), 1)$ and then $(\mathrm{R}_1^2(a), 3)$.

We can show that our notion of automata covers exactly the class of regular MSC word languages.

Theorem 5. *An MSC word language $\mathcal{L}$ is regular iff there is an MFA $\mathcal{A}$ such that $\mathcal{L} = \mathcal{L}(\mathcal{A})$.*

Proof. Exploiting Theorem 3, we specify respective automata.

($\Longrightarrow$) Let $\mathcal{A} = (S, \longrightarrow, q_0, F)$ be a finite automaton with $\longrightarrow \subseteq S \times (\Sigma \times Q) \times S$ for a finite set $Q \subseteq \mathbb{N}$ such that $\mathcal{L}(\mathcal{A}) = nf(\mathcal{L})$. We can consider $\mathcal{A}$ as an MFA $\mathcal{A}' = (S', r, \Delta, q_0', F')$ satisfying $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})^{\approx} = \mathcal{L}$ as follows:

- $S' = S$,
- $r = |Q|$,
- $(s, (\sigma, k), t) \in \Delta$ iff $s \xrightarrow{(\sigma,k)} t$,
- $q_0' = q_0$, and
- $F' = F$.

($\Longleftarrow$) Given an MFA, the corresponding set of MSC words in normal form makes use of finitely many configurations only, which become states in the finite automaton to be constructed. Thus, for an MFA $\mathcal{A} = (S, r, \Delta, q_0, F)$, let $Q = \{1, \ldots, r\}$ and $\mathcal{A}' = (S', \longrightarrow, q_0', F')$ be the corresponding finite automaton satisfying $\mathcal{L}(\mathcal{A}') = nf(\mathcal{L}(\mathcal{A}))$, defined as follows:

- $S' = S \times ((\Sigma_{\mathcal{S}} \times Q) \cup \{\#\})^r$,
- $\longrightarrow \subseteq S' \times (\Sigma \times Q) \times S'$ where $(s, \mathbf{w}) \xrightarrow{(\sigma,\pi)} (t, \mathbf{v})$ iff both $(s, \mathbf{w}) \overset{(\sigma,\pi)}{\Longrightarrow}_{\mathcal{A}} (t, \mathbf{v})$ and $\sigma \in \Sigma_{\mathcal{S}}$ implies $\pi = \min\{\pi' \in \mathbb{N} \mid (\sigma, \pi')$ does not occur in $\mathbf{w}\}$,
- $q_0' = (q_0, \#^r)$, and
- $F' = F \times \{\#^r\}$.

In both cases, it is straightforward to show that the constructed automaton has the desired property. □

Given an MSC word language in terms of an MFA, the first natural question is whether it defines the trivial language.

Theorem 6. *It is decidable whether a regular MSC word language given by an MFA is empty.*

Proof. Theorem 6 is a direct consequence of Theorem 3 and the constructive proof of Theorem 5. Given an MFA $\mathcal{A}$, build the corresponding finite automaton $\mathcal{A}'$ satisfying $\mathcal{L}(\mathcal{A}') = nf(\mathcal{L}(\mathcal{A}))$, which is then checked for emptiness by means of standard techniques. And clearly, $\mathcal{L}(\mathcal{A}) = \emptyset$ iff $\mathcal{L}(\mathcal{A}') = \emptyset$. □

By means of Theorem 3 and due to the fact that the concatenation of two MSC words in normal form is again in normal form, we obtain some closure properties of regular MSC word languages.

Theorem 7. *The class of regular MSC word languages is closed under union, intersection, concatenation, and Kleene star.*

To support the algorithmic handling of MFAs, we furthermore illustrate corresponding automata-theoretic constructions, which also establish the above closure properties.

Let $\mathcal{A}_1 = (S_1, r_1, \Delta_1, q_{01}, F_1)$ and $\mathcal{A}_2 = (S_2, r_2, \Delta_2, q_{02}, F_2)$ be MFAs, $S_1 \cap S_2 = \emptyset$. The MFA $\mathcal{A}^{\cup} = (S, r, \Delta, q_0, F)$, $q_0 \notin S_1 \cup S_2$, satisfying $\mathcal{L}(\mathcal{A}^{\cup}) = \mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$, is given by

- $S = S_1 \cup S_2 \cup \{q_0\}$,
- $r = r_1 + r_2$,
- $(s, (\sigma, k), t) \in \Delta$ iff
  - $s = q_0$ and $(q_{01}, (\sigma, k), t) \in \Delta_1$ or
  - $s = q_0$ and $(q_{02}, (\sigma, k - r_1), t) \in \Delta_2$ or
  - $(s, (\sigma, k), t) \in \Delta_1$ or
  - $(s, (\sigma, k - r_1), t) \in \Delta_2$, and
- $F = F_1 \cup F_2 \cup \{q_0 \mid q_{01} \in F_1 \text{ or } q_{02} \in F_2\}$.

The main idea is to assign to each of the automata an exclusive part of the window. A run first decides to enter either $\mathcal{A}_1$ or $\mathcal{A}_2$ and is henceforth limited to the respective part of the window.

The other cases require the automata constructions used in the proof of Theorem 5. Finite automata accepting normal forms can be intersected and concatenated using standard techniques. As mentioned above, the resulting automata can be understood as MFAs.

We easily see that the class of regular MSC word languages is not closed under complement, because the complement of a regular MSC word language is always unbounded. This also implies that the standard way to show decidability of language inclusion does not work. However, in contrast to the general case of regular languages over infinite alphabets where language inclusion is undecidable (see [15]), we can directly show that the inclusion problem is decidable. This is of great importance for the development of model checking algorithms.

Theorem 8. *Given MFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, it is decidable whether $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$.*

Proof. According to the construction in the proof of Theorem 5, we are able to ascribe decidability to the case of regular languages over finite alphabets. In fact, given finite automata $\mathcal{A}'_1$ and $\mathcal{A}'_2$ with $\mathcal{L}(\mathcal{A}'_1) = nf(\mathcal{L}(\mathcal{A}_1))$ and $\mathcal{L}(\mathcal{A}'_2) = nf(\mathcal{L}(\mathcal{A}_2))$, it holds $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ iff $\mathcal{L}(\mathcal{A}'_1) \subseteq \mathcal{L}(\mathcal{A}'_2)$. ☐

4 Regular MSC Languages and Their Automata

4.1 Regular MSC Languages

We now extend our theory of regular MSC word languages to collections of MSCs. Regularity of such a collection is ascribed to regularity of the set of corresponding linearisations.

Definition 5 (Regular MSC Language). *A collection $\mathcal{M}$ of MSCs is called a* regular MSC language *iff $Lin(\mathcal{M})$ is a regular MSC word language.*

According to this definition, the set of linearisations of a regular MSC language is necessarily $\sim$-closed by Theorem 2. Hence, regular MSC languages cannot be characterised by MFAs as presented in the previous section because these accepts also non-$\sim$-closed languages. We therefore develop a generalisation of message-passing automata [8] that accept exactly regular MSC word languages corresponding to regular MSC languages. It should be noted that a regular MSC language is bounded.

One might ask at this stage for the reason considering regular MSC word languages as well as regular MSC languages because the latter seem to be the first choice studying linearisations of MSCs. This is true when we abstract from a communication medium between the processes of an MSC. Consider for example the MSC presented in Figure 1(b). In the visual-order approach, there is no difference whether $\mathrm{S}_3^2(b)$ occurs before $\mathrm{R}_1^2(a)$ or vice versa. However, turning towards more complex semantics of MSCs, this might not be true any longer. Suppose the two processes communicate via a one element buffer. Then the only linear execution we will see is that $\mathrm{R}_1^2(a)$ occurs before $\mathrm{S}_3^2(b)$. Thus, the set of linearisations of an MSC is no longer necessarily $\sim$-closed. It is indeed possible to model communication mediums by means of certain MFAs, which enrich a specification in form of MSCs [3]. However, let us come back to visual-order semantics and to $\sim$-closed languages.

4.2 Generalised Message-Passing Automata

The following automata model employs different automata components, each of which executes the actions of one single process. They communicate with each other over a window roughly as featured by an MFA. The length of this window is still bounded by a natural number r. The crucial point is that the window entries are no longer single send events (each paired with a natural number) but sequences of send events (each paired with a natural number and an additional message). To preserve $\sim$-closedness of the recognised languages, the components rather have to restrict themselves, whereas the window is a communication medium only. For example, we could imagine an automata component that has to keep a send action waiting until it executes a certain receive action, which, in turn, has to be preceded by a corresponding send action executed by another component. In fact, for regular languages, our view generalises the model proposed in [8], which has its origins in [16].

Definition 6. *A* generalised message-passing automaton *(GMPA) is a family* $\mathcal{A} = (\{\mathcal{A}_p\}_{p\in\mathcal{P}_N}, r, \overline{q}_{in}, F, \mathit{Mess})$ *of so-called* local automata *together with a natural number* $r \geq 1$, *a* global initial state $\overline{q}_{in}$, *a set of* global final states F, *and a nonempty finite set of* messages Mess. *A local automaton is of the form* $\mathcal{A}_p = (S_p, \Delta_p)$ *where*

- S_p *is a nonempty finite set of* local states *and*
- $\Delta_p \subseteq S_p \times (\Sigma_p \times \{1,\ldots,r\} \times \mathit{Mess}) \times S_p$ *is a set of* local transitions *(*Σ_p *contains the actions belonging to Process* p*).*

$\overline{q}_{in}$ *is an element and* F *a subset of* $S_{\mathcal{A}} := \times_{p\in\mathcal{P}_N} S_p$, *the set of* global states *of* $\mathcal{A}$.

For an GMPA $\mathcal{A}$, the (infinite) set of its *configurations* is defined by $\mathit{Conf}_{\mathcal{A}} := S_{\mathcal{A}} \times \{\chi \mid \chi : \Sigma_{\mathcal{S}} \times \{1,\ldots,r\} \to (\mathbb{N} \times \mathit{Mess})^*\}$. Let $\overline{s}[p]$ be the pth component of a global state $\overline{s} \in S_{\mathcal{A}}$. Furthermore, for $\mathtt{W} : \Sigma_{\mathcal{S}} \times \{1,\ldots,r\} \to (\mathbb{N} \times \mathit{Mess})^*$, let $\mathtt{W}[(\sigma,k) \;/\; w]$ denote the function that coincides with $\mathtt{W}$ with the exception that, for (σ,k), it yields w. We define a transition relation $\Longrightarrow_{\mathcal{A}} \subseteq \mathit{Conf}_{\mathcal{A}} \times (\Sigma \times \mathbb{N}) \times \mathit{Conf}_{\mathcal{A}}$ as follows:

- For $\sigma \in \Sigma_{\mathcal{S}}$ with $P(\sigma) = p$, $(\overline{s},\mathtt{W}) \overset{(\sigma,\pi)}{\Longrightarrow}_{\mathcal{A}} (\overline{t},\mathtt{V})$ iff for all $k' \in \{1,\ldots,r\}$ and $m' \in \mathit{Mess}$, (π,m') does not occur in $\mathtt{W}(\sigma,k')$, and there is a transition $(\overline{s}[p],(\sigma,k,m),\overline{t}[p]) \in \Delta_p$ such that
  - $\mathtt{V} = \mathtt{W}[(\sigma,k) \;/\; \mathtt{W}(\sigma,k)\cdot(\pi,m)]$ and
  - for all $l \in \mathcal{P}_N - \{p\}$, $\overline{s}[l] = \overline{t}[l]$.
- For $\sigma \in \Sigma_{\mathcal{R}}$ with $P(\sigma) = p$ and $(\tau,\sigma) \in \mathit{Corr}$, $(\overline{s},\mathtt{W}) \overset{(\sigma,\pi)}{\Longrightarrow}_{\mathcal{A}} (\overline{t},\mathtt{V})$ iff there are a transition $(\overline{s}[p],(\sigma,k,m),\overline{t}[p]) \in \Delta_p$ and a word $w \in (\mathbb{N} \times \mathit{Mess})^*$ such that
  - $\mathtt{W}(\tau,k) = (\pi,m)\cdot w$,
  - $\mathtt{V} = \mathtt{W}[(\tau,k) \;/\; w]$, and
  - for all $l \in \mathcal{P}_N - \{p\}$, $\overline{s}[l] = \overline{t}[l]$.

A *run* of $\mathcal{A}$ on a word $\frac{\sigma_1}{\pi_1}\cdots\frac{\sigma_\ell}{\pi_\ell} \in (\Sigma \times \mathbb{N})^*$ is defined in analogy to the MFA case. That is, we are dealing with a sequence $(\overline{s}_0,\mathtt{W}_0)(\overline{s}_1,\mathtt{W}_1)\ldots(\overline{s}_\ell,\mathtt{W}_\ell)$ of configurations such that

- $\overline{s}_0 = \overline{q}_{in}$,
- $\mathtt{W}_0(\sigma,k) = \varepsilon$ for all $(\sigma,k) \in \Sigma_{\mathcal{S}} \times \{1,\ldots,r\}$, and
- for each $i \in \{1,\ldots,\ell\}$, $(\overline{s}_{i-1},\mathtt{W}_{i-1}) \overset{(\sigma_i,\pi_i)}{\Longrightarrow}_{\mathcal{A}} (\overline{s}_i,\mathtt{W}_i)$.

The run is *accepting* iff $\overline{s}_\ell \in F$ and $\mathtt{W}_\ell(\sigma, k) = \varepsilon$ for all $(\sigma, k) \in \Sigma_\mathcal{S} \times \{1, \ldots, r\}$. Finally, $\mathcal{L}(\mathcal{A}) := \{\alpha \mid \text{there is an accepting run of } \mathcal{A} \text{ on } \alpha\}$ denotes the language defined by $\mathcal{A}$.

Let $\mathit{Reach}(\mathcal{A})$ denote the set of configurations reachable within a run of $\mathcal{A}$. For $B \in \mathbb{N}$, we call $\mathcal{A}$ *B-bounded* iff for all $(\overline{s}, \mathtt{W}) \in \mathit{Reach}(\mathcal{A})$ and $\sigma \in \Sigma_\mathcal{S}$, $\sum_{k \in \{1,\ldots,r\}} |\mathtt{W}(\sigma, k)| \leq B$. We call it *bounded* iff it is B-bounded for some B. Figure 8 illustrates a possible run of the automaton $\mathcal{A}$ from Figure 7.

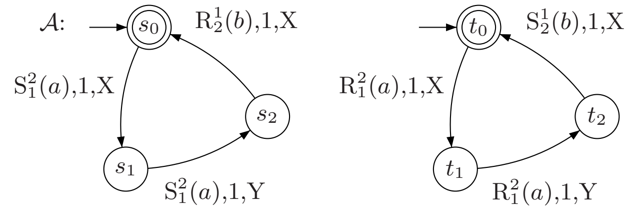


Fig. 7. A 2-bounded GMPA

Let us formulate the fundamental result of this section.

Theorem 9. *Let $\mathcal{L} \subseteq \mathrm{MW}$ be an MSC word language. The following statements are equivalent:*

1. *There is a regular MSC language $\mathcal{M}$ with $\mathit{Lin}(\mathcal{M}) = \mathcal{L}$.*
2. *$\mathcal{L}$ is a $\sim$-closed regular MSC word language.*
3. *There is a bounded GMPA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}$.*

Proof. The equivalence of 1. and 2. immediately follows from the definitions. Given a bounded GMPA, it is an easy task to define an equivalent MFA which shows that 3. implies 2. The other direction, however, is more involved and requires some results on regular Mazurkiewicz trace languages and related automata due to Zielonka [16].

A nonempty family $\widetilde{A} = \{A_i\}_{i \in \{1,\ldots,n\}}$ of (not necessarily disjoint) finite alphabets is called a *distributed alphabet*. From now on, for $z \in A := \bigcup_{i \in \{1,\ldots,n\}} A_i$, $\mathit{loc}(z) := \{i \mid z \in A_i\}$ will denote the set of components in which z is involved. An *asynchronous automaton* over the distributed alphabet $\widetilde{A}$ is a tuple $\mathcal{Z} = (\{Q_i\}_{i \in \{1,\ldots,n\}}, \{\longrightarrow_z\}_{z \in A}, \overline{q}_0, F)$ where

- Q_i is a nonempty finite set of *local states*,
- $\longrightarrow_z \subseteq (\bigtimes_{i \in \mathit{loc}(z)} Q_i)^2$ is the *transition relation* of z,
- $\overline{q}_0 \in \bigtimes_{i \in \{1\ldots,n\}} Q_i$ is the (global) *initial state*, and
- $F \subseteq \bigtimes_{i \in \{1\ldots,n\}} Q_i$ is the set of (global) *final states*.

Let $Q_\mathcal{Z} := \bigtimes_{i \in \{1\ldots,n\}} Q_i$ denote the set of *global states* of $\mathcal{Z}$, and, for $C \subseteq \{1, \ldots, n\}$, let $\overline{q}_{|C}$ be the projection of $\overline{q} \in Q_\mathcal{Z}$ onto the components from C. The *global transition relation* $\Longrightarrow_\mathcal{Z} \subseteq Q_\mathcal{Z} \times A \times Q_\mathcal{Z}$ of $\mathcal{Z}$ is defined as follows:

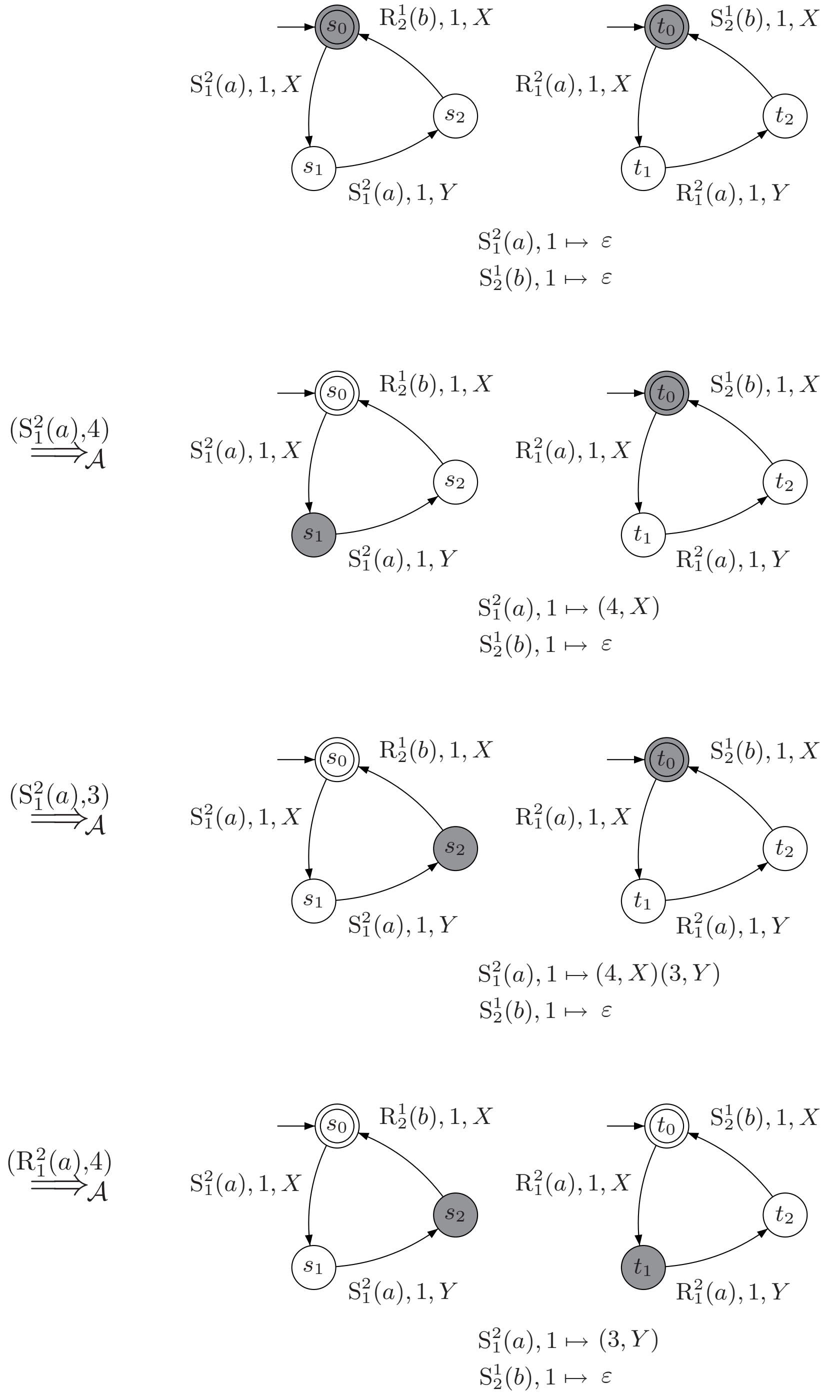


Fig. 8. A run of $\mathcal{A}$

$\overline{q} \overset{z}{\Longrightarrow}_{\mathcal{Z}} \overline{q}'$ iff $\overline{q}_{|loc(z)} \longrightarrow_z \overline{q}'_{|loc(z)}$ and $\overline{q}_{|\{1,\ldots,n\}-loc(z)} = \overline{q}'_{|\{1,\ldots,n\}-loc(z)}$. The language of $\mathcal{Z}$, $\mathcal{L}(\mathcal{Z}) \subseteq A^*$, is defined in the obvious manner.

$\tilde{A}$ induces the dependence alphabet (A, D), $D = \{(z, z') \mid \exists i \in \{1, \ldots, n\} : \{z, z'\} \subseteq A_i\}$. According to [16], $X \subseteq A^*$ is a regular Mazurkiewicz trace language over (A, D) iff there is an (even deterministic) asynchronous automaton $\mathcal{Z}$ over $\tilde{A}$ recognising X.

Given a B-bounded $\sim$-closed regular MSC word language $\mathcal{L}$, we build a B-bounded GMPA $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = \mathcal{L}$. The outline of this construction is as follows: We first observe that $\mathit{nf}(\mathcal{L})^{\sim}$ can be considered to be a regular Mazurkiewicz trace language over $\Sigma \times \{1, \ldots, B\}$ with an appropriate dependence alphabet. Then we can find an asynchronous automaton recognising $\mathit{nf}(\mathcal{L})^{\sim}$. The underlying distributed alphabet will comprise, apart from alphabets for each process, some additional components, which guarantee that the induced dependence relation complies with $D(\Sigma)$ (see also the proof of Theorem 4). These additional components have to be factored into the process components and the transition relation, making the transformation of the asynchronous automaton into a GMPA complicated. Concretely, the transitions synchronously taken by several local automata have to be simulated by message passing. For example, consider Process P_1 sending a message to Process P_2 by executing (σ, k). Actually, an equally labelled transition would have to be taken on the part of an additional component, in which (σ, k) is involved. But as in the GMPA such a component is not at the disposal of P_1, P_1 guesses a corresponding move and writes it, along with the original message, into the message pool. The receiving process can take this message if the guessed move corresponds to the actual state of the additional component, which P_2 carries along. Our construction is similar to the one in [8] but uses the time-stamping protocol for *non*-FIFO computations described in [14] to ensure boundedness of the constructed GMPA. However, the main difference is the choice of the underlying distributed alphabet. Recalling the ideas of [12], it reflects precisely the dependence relation $D(\Sigma)$, while [8] deals with a context-sensitive dependence relation, which cannot be directly represented as a (static) distributed alphabet.

Let $Co := \{(\sigma, \tau, \pi) \mid (\sigma, \tau) \in \mathit{Corr}, \pi \in \{1, \ldots, B\}\}$ and, for $(\sigma, \tau, \pi) \in Co$, $\Sigma_{(\sigma,\tau,\pi)} := \{(\sigma, \pi), (\tau, \pi)\}$. We define the distributed alphabet $\tilde{\Sigma}$ to be $\{\Sigma_x\}_{x \in \mathcal{P}_N \cup Co}$. Then $\tilde{\Sigma}$ induces the dependence alphabet $\widehat{\Sigma} := (\Sigma \times \{1, \ldots, B\}, D(\Sigma) \cap (\Sigma \times \{1, \ldots, B\})^2)$. In accordance with Theorems 1, 2, and 4, $\mathit{nf}(\mathcal{L})^{\sim}$ forms a regular Mazurkiewicz trace language over the alphabet $\widehat{\Sigma}$. Thus, there is an asynchronous automaton $\mathcal{Z} = (\{Q_x\}_{x \in \mathcal{P}_N \cup Co}, \{\longrightarrow_{(\sigma,\pi)}\}_{(\sigma,\pi) \in \Sigma \times \{1,\ldots,B\}}, \overline{q}_0, F)$ over $\tilde{\Sigma}$ such that $\mathcal{L}(\mathcal{Z}) = \mathit{nf}(\mathcal{L})^{\sim}$. Let TS denote the set of time-stamps generated by the protocol of [14]. For $\mathit{Mess} = (\bigcup_{x \in Co}(Q_x \times Q_x)) \times TS$, we define $\mathcal{A}$ to be the GMPA $(\{\mathcal{A}_p\}_{p \in \mathcal{P}_N}, B, \overline{q}_{in}, F', \mathit{Mess})$, $\mathcal{A}_p = (S_p, \Delta_p)$, with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{Z})^{\approx} = \mathcal{L}$. $\mathcal{A}_p$ is given by $S_p = \{(s, \overline{s}, \vartheta) \mid s \in Q_p,\ \overline{s} \in \bigtimes_{x \in Co, \Sigma_x \cap \Sigma_p \subseteq \Sigma_{\mathcal{R}}} Q_x,\ \vartheta \in TS\}$ and the following:

- For $\sigma = \mathrm{S}_p^q(\lambda)$ and $\tau = \mathrm{R}_p^q(\lambda)$, $((s_p, \overline{s}_p, \vartheta_p), (\sigma, k, m), (s'_p, \overline{s}'_p, \vartheta'_p)) \in \Delta_p$ iff $\overline{s}_p = \overline{s}'_p$ and there are $s, s' \in Q_{(\sigma,\tau,k)}$ with
  - $m = ((s, s'), \vartheta'_p)$,
  - $(s_p, s) \longrightarrow_{(\sigma,k)} (s'_p, s')$, and
  - ϑ'_p is the time-stamp generated from ϑ_p by the protocol of [14].
- For $\sigma = \mathrm{S}_p^q(\lambda)$ and $\tau = \mathrm{R}_p^q(\lambda)$, $((s_q, \overline{s}_q, \vartheta_q), (\tau, k, m), (s'_q, \overline{s}'_q, \vartheta'_q)) \in \Delta_q$ iff there are $s, s', s'' \in Q_{(\sigma,\tau,k)}$ and $\vartheta \in TS$ with
  - $m = ((s, s'), \vartheta)$,
  - $s = \overline{s}_q[(\sigma, \tau, k)]$,
  - $(s_q, s') \longrightarrow_{(\tau,k)} (s'_q, s'')$,
  - $\overline{s}'_q[x] = \begin{cases} s'' & \text{iff } x = (\sigma, \tau, k) \\ \overline{s}_q[x] & \text{otherwise} \end{cases}$, and
  - ϑ'_p is the time-stamp generated from ϑ_p and ϑ by the protocol of [14].

$\overline{q}_{in}$ and F' are defined as expected.

Thanks to the use of the time-stamping protocol, $\mathcal{A}$ is B-bounded. Furthermore, it is straightforward to show that $\mathcal{L}(\mathcal{Z})^{\approx} \subseteq \mathcal{L}(\mathcal{A})$. On the other hand, each accepting run of $\mathcal{A}$ on a word σ can be reordered resulting in an accepting run of $\mathcal{Z}$ on a word σ' with $\sigma(\approx \cup \sim)^*\sigma'$. For example, $\mathcal{A}$ may simulate a run of $\mathcal{Z}$ on $\underset{1}{\mathrm{S}_1^2(a)}\ \underset{1}{\mathrm{R}_1^2(a)}\ \underset{1}{\mathrm{S}_1^2(a)}\ \underset{1}{\mathrm{R}_1^2(a)}$ by a run on α_1 from Figure 3. Accordingly, $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{Z})^{\approx}$ which concludes our proof. □

Thus, bounded GMPAs characterise exactly those regular MSC word languages that are $\sim$-closed and therewith exactly the regular MSC languages. For example, the 2-bounded GMPA $\mathcal{A}$ given in Figure 7 recognises the $(\approx \cup \sim)^*$-closure of

$$\left\{ \left(\underset{1}{\mathrm{S}_1^2(a)}\ \underset{2}{\mathrm{S}_1^2(a)}\ \underset{1}{\mathrm{R}_1^2(a)}\ \underset{2}{\mathrm{R}_1^2(a)}\ \underset{1}{\mathrm{S}_2^1(b)}\ \underset{1}{\mathrm{R}_2^1(b)} \right)^n \;\middle|\; n \geq 0 \right\}.$$

5 A Logical Characterisation

We formulate a monadic second-order logic that characterises exactly the class of regular MSC languages. Given a supply $\mathrm{Var} = \{x, y, \ldots\}$ of individual variables and a supply $\mathrm{VAR} = \{X, Y, \ldots\}$ of set variables, the syntax of $\mathrm{MSO}(\mathcal{P}_N, \Lambda)$ is defined by

$$\varphi ::= L_\sigma(x) \mid x \in X \mid x \preceq y \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \exists x\varphi \mid \exists X\varphi \in \mathrm{MSO}(\mathcal{P}_N, \Lambda)$$

where $\sigma \in \Sigma$, $x, y \in \mathrm{Var}$, and $X \in \mathrm{VAR}$. Moreover, we allow the usual abbreviations. Let M be an MSC with set of events E and labelling function L. Given an interpretation function $\mathcal{I}$, which assigns to an individual variable x an event $\mathcal{I}(x) \in E$ and to a set variable X a set of events $\mathcal{I}(X) \subseteq E$, the satisfaction relation $M \models_{\mathcal{I}} \varphi$ for an MSC M and a formula $\varphi \in \mathrm{MSO}(\mathcal{P}_N, \Lambda)$ is inductively defined as follows:

- $M \models_{\mathcal{I}} L_\sigma(x)$ iff $L(\mathcal{I}(x)) = \sigma$,

- $M \models_{\mathcal{I}} x \in X$ iff $\mathcal{I}(x) \in \mathcal{I}(X)$,
- $M \models_{\mathcal{I}} x \preceq y$ iff $\mathcal{I}(x) \preceq \mathcal{I}(y)$,
- $M \models_{\mathcal{I}} \neg\varphi$ iff $M \not\models_{\mathcal{I}} \varphi$,
- $M \models_{\mathcal{I}} \varphi \vee \psi$ iff $M \models_{\mathcal{I}} \varphi$ or $M \models_{\mathcal{I}} \psi$,
- $M \models_{\mathcal{I}} \exists x\varphi$ iff $\exists e \in E :\ M \models_{\mathcal{I}[x/e]} \varphi$, and
- $M \models_{\mathcal{I}} \exists X\varphi$ iff $\exists E' \subseteq E :\ M \models_{\mathcal{I}[X/E']} \varphi$.

From now on, we only consider formulas without free variables and accordingly write $M \models \varphi$ instead of $M \models_{\mathcal{I}} \varphi$. For $\varphi \in \mathrm{MSO}(\mathcal{P}_N, \Lambda)$ and $B \in \mathbb{N}$, let $\mathcal{M}_\varphi^B := \{M \mid M \text{ is } B\text{-bounded and } M \models \varphi\}$.

We present the fundamental result of this section.

Theorem 10. *Given a collection $\mathcal{M}$ of MSCs, $\mathcal{M}$ is a regular MSC language iff there exist a formula $\varphi \in \mathrm{MSO}(\mathcal{P}_N, \Lambda)$ and $B \in \mathbb{N}$ such that*

$$Lin(\mathcal{M}) = Lin(\mathcal{M}_\varphi^B).$$

Proof. The proof follows the outline of [8] although the concrete steps are different.

($\Longrightarrow$) Let $\mathcal{M}$ be a regular MSC language. We can find $B \in \mathbb{N}$ such that $Lin(\mathcal{M})$ is a $\sim$-closed B-bounded regular MSC word language. Due to Büchi's Theorem [5] and Theorem 4, there is a formula $\varphi \in \mathrm{MSO}(\Sigma \times \{1, \ldots, B\})$ over words such that $\mathcal{L}_\varphi = nf(Lin(\mathcal{M}))^\sim$ where $\mathcal{L}_\varphi = \{\alpha \mid \alpha \models \varphi\}$. Accordingly, our aim is to define a formula $\overline{\varphi} \in \mathrm{MSO}(\mathcal{P}_N, \Lambda)$ with $Lin(\mathcal{M}_{\overline{\varphi}}^B) = \mathcal{L}_\varphi^{\approx}$. Let $\overline{\varphi} = \|\varphi\|$ where

- $\|L_{(\sigma,\pi)}(x)\| := L_\sigma(x)$,
- $\|x \in X\| := x \in X$,
- $\|\neg\psi\| := \neg\|\psi\|$,
- $\|\psi_1 \vee \psi_2\| := \|\psi_1\| \vee \|\psi_2\|$,
- $\|\exists x\psi\| := \exists x\|\psi\|$,
- $\|\exists X\psi\| := \exists X\|\psi\|$, and
- $\|x \leq y\| := (x \preceq y \wedge y \preceq x) \vee Lex(x, y)$.

The last item weakens the relation $\leq$ since, informally spoken, $x \leq y$ on the part of words does not imply $x \preceq y$ on the part of MSCs. By means of the predicate $Lex(x, y)$, as it is defined in [8], we fall back on a canonical linearisation of an MSC along which the formula to be constructed is interpreted.

($\Longleftarrow$) We transform φ into a sentence $\overline{\varphi} \in \mathrm{MSO}(\Sigma \times \{1, \ldots, B\})$ over words such that $\mathcal{L}_{\overline{\varphi}} = nf(Lin(\mathcal{M}_\varphi^B))$. $\overline{\varphi}$ is of the form $Comp \wedge Normal \wedge \|\varphi\|$. *Comp* ensures that only MSC words are defined, *Normal* guarantees that we finally deal with MSC words in normal form, and $\|\varphi\|$ helps to restrict the set of such MSC words to the ones contained in $Lin(\mathcal{M}_\varphi^B)$. Let us begin: Due to the underlying finite alphabet, $\|\varphi\|$ specifies B-bounded words. So let

$$Comp = \bigwedge_{\substack{(\sigma,\tau) \in Corr \\ \pi \in \{1,\ldots,B\}}} \Phi_1(\sigma, \tau, \pi) \wedge \Phi_2(\sigma, \tau, \pi) \wedge \Phi_3(\sigma, \tau, \pi) \qquad \text{where}$$

- $\Phi_1(\sigma,\tau,\pi) =$
  $\forall x \forall y [x < y \wedge L_{(\sigma,\pi)}(x) \wedge L_{(\sigma,\pi)}(y) \longrightarrow \exists z (x < z < y \wedge L_{(\tau,\pi)}(z))]$
  ensures that (σ,π) and (τ,π) only occur in turns,
- $\Phi_2(\sigma,\tau,\pi) = \forall x [L_{(\tau,\pi)}(x) \longrightarrow \exists y (y < x \wedge L_{(\sigma,\pi)}(y))]$
  guarantees that (τ,π) is preceded by its send event (σ,π), and furthermore
- $\Phi_3(\sigma,\tau,\pi) = \forall x [L_{(\sigma,\pi)}(x) \longrightarrow \exists y (x < y \wedge L_{(\tau,\pi)}(y))]$
  makes sure that a send event (σ,π) is finally followed by its receive event (τ,π).

Notice that, in the context of *Comp*, Φ_1 and Φ_2 lead to properness, whereas completeness is only guaranteed with the help of Φ_3. Let furthermore

$$\mathit{Normal} = \bigwedge_{\substack{(\sigma,\tau) \in \mathit{Corr} \\ \pi \in \{2,\ldots,B\}}} \forall x [L_{(\sigma,\pi)}(x) \longrightarrow \exists y (y < x \wedge L_{(\sigma,\pi-1)}(y) \wedge \forall z (y < z < x \longrightarrow \neg L_{(\tau,\pi-1)}(z)))].$$

Finally, we inductively obtain $\|\varphi\|$ as above except that

- $\|L_\sigma(x)\| := \bigvee_{\pi \in \{1,\ldots,B\}} L_{(\sigma,\pi)}(x)$ and
- $\|x \preceq y\| := \trianglelefteq(x,y)$

where $\trianglelefteq$ is defined by

$$\trianglelefteq(x,y) := \exists X [\, x \in X \wedge y \in X \wedge \forall z (z \in X \wedge z \neq y \longrightarrow \exists z' (\, z' \in X \wedge z < z' \wedge \mathit{Proc}(z) = \mathit{Proc}(z') \vee \searrow(z,z')))]$$

where *Proc* is used and defined in the obvious manner and the matching predicate is given as follows:

$$\searrow(x,y) := \bigvee_{\substack{(\sigma,\tau) \in \mathit{Corr} \\ \pi \in \{1,\ldots,B\}}} [x < y \wedge L_{(\sigma,\pi)}(x) \wedge L_{(\tau,\pi)}(y) \wedge \nexists z (x < z < y \wedge L_{(\tau,\pi)}(z))]$$

In fact, one can easily see that the formula $\overline{\varphi}$ has the desired properties. From Büchi's Theorem [5], we conclude that this direction of Theorem 10 holds. □

References

1. R. Alur and M. Yannakakis. Model checking of message sequence charts. In *Proceedings of the 10th International Conference on Concurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 114–129. Springer, 1999.
2. João Araújo. Formalizing sequence diagrams. In *Proceedings of the OOPSLA'98 Workshop on Formalizing UML. Why? How?*, volume 33, 10 of *ACM SIGPLAN Notices*, New York, 1998. ACM Press.

3. Benedikt Bollig and Martin Leucker. Modelling, Specifying, and Verifying Message Passing Systems. In Claudio Bettini and Angelo Montanari, editors, *Proceedings of the Symposium on Temporal Representation and Reasoning (TIME'01)*, pages 240–248. IEEE Computer Society Press, June 2001.
4. Benedikt Bollig, Martin Leucker, and Thomas Noll. Regular MSC Languages. Technical Report AIB-05-2001, RWTH Aachen, April 2001.
5. J. Büchi. Weak second order logic and finite automata. *Z. Math. Logik, Grundlag. Math.*, 5:66–62, 1960.
6. Volker Diekert and Yves Métivier. Partial commutation and traces. In G. Rozenberg and A. Salomaa, editors, *Handbook on Formal Languages*, volume III. Springer, Berlin-Heidelberg-New York, 1997.
7. Elsa Gunter, Anca Muscholl, and Doron Peled. Compositional message sequence charts. In Tiziana Margaria and Wang Yi, editors, *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'01)*, volume 2031 of *Lecture Notes in Computer Science*, pages 496–511. Springer, April 2001.
8. J. G. Henriksen, M. Mukund, K. Narayan Kumar, and P. S. Thiagarajan. Regular collections of message sequence charts. In *Proceedings of 25th International Symposium on Mathemtical Foundations of Computer Science (MFCS'2000)*, volume 1893 of *Lecture Notes in Computer Science*, pages 405–414. Springer, 2000.
9. ITU-TS. ITU-TS Recommendation Z.120anb: Formal Semantics of Message Sequence Charts. Technical report, ITU-TS, Geneva, 1998.
10. ITU-TS. ITU-TS Recommendation Z.120: Message Sequence Chart 1999 (MSC99). Technical report, ITU-TS, Geneva, 1999.
11. Michael Kaminski and Nissim Francez. Finite-memory automata. *Theoretical Computer Science*, 134(2):329–363, November 1994.
12. Dietrich Kuske. Another step towards a theory of regular MSC languages. In *Proceedings of the 19th International Symposium on Theoretical Aspects of Computer Science (STACS'02), 2002*, Lecture Notes in Computer Science. Springer, 2002.
13. P. Madhusudan and B. Meenakshi. Beyond message sequence graphs. In *Proceedings of the 21st Conference on Foundations of Software Technology and Theoretical Computer Science*, Lecture Notes in Computer Science. Springer, 2001.
14. M. Mukund, K. Narayan Kumar, and M. Sohoni. Keeping track of the latest gossip in message-passing systems. Technical Report TCS-95-3, School of Mathematics, SPIC Science Foundation, Madras, India, 1995.
15. Frank Neven, Thomas Schwentick, and Victor Vianu. Towards regular languages over infinite alphabets. In *Proceedings of 26th International Symposium on Mathematical Foundations of Computer Science (MFCS'01)*, Lecture Notes in Computer Science. Springer, 2001.
16. Wiesław Zielonka. Notes on finite asynchronous automata. *R.A.I.R.O. — Informatique Théorique et Applications*, 21:99–135, 1987.

Aachener Informatik-Berichte

This is a list of recent technical reports. To obtain copies of technical reports please consult http://aib.informatik.rwth-aachen.de/ or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

95-11 *	M. Staudt / K. von Thadden: Subsumption Checking in Knowledge Bases
95-12 *	G.V. Zemanek / H.W. Nissen / H. Hubert / M. Jarke: Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology
95-13 *	M. Staudt / M. Jarke: Incremental Maintenance of Externally Materialized Views
95-14 *	P. Peters / P. Szczurko / M. Jeusfeld: Business Process Oriented Information Management: Conceptual Models at Work
95-15 *	S. Rams / M. Jarke: Proceedings of the Fifth Annual Workshop on Information Technologies & Systems
95-16 *	W. Hans / St. Winkler / F. Sáenz: Distributed Execution in Functional Logic Programming
96-1 *	Jahresbericht 1995
96-2	M. Hanus / Chr. Prehofer: Higher-Order Narrowing with Definitional Trees
96-3 *	W. Scheufele / G. Moerkotte: Optimal Ordering of Selections and Joins in Acyclic Queries with Expensive Predicates
96-4	K. Pohl: PRO-ART: Enabling Requirements Pre-Traceability
96-5	K. Pohl: Requirements Engineering: An Overview
96-6 *	M. Jarke / W. Marquardt: Design and Evaluation of Computer–Aided Process Modelling Tools
96-7	O. Chitil: The ς-Semantics: A Comprehensive Semantics for Functional Programs
96-8 *	S. Sripada: On Entropy and the Limitations of the Second Law of Thermodynamics
96-9	M. Hanus (Ed.): Proceedings of the Poster Session of ALP'96 — Fifth International Conference on Algebraic and Logic Programming
96-10	R. Conradi / B. Westfechtel: Version Models for Software Configuration Management
96-11 *	C. Weise / D. Lenzkes: A Fast Decision Algorithm for Timed Refinement
96-12 *	R. Dömges / K. Pohl / M. Jarke / B. Lohmann / W. Marquardt: PRO-ART/CE* — An Environment for Managing the Evolution of Chemical Process Simulation Models
96-13 *	K. Pohl / R. Klamma / K. Weidenhaupt / R. Dömges / P. Haumer / M. Jarke: A Framework for Process-Integrated Tools

96-14 * R. Gallersdörfer / K. Klabunde / A. Stolz / M. Eßmajor: INDIA — Intelligent Networks as a Data Intensive Application, Final Project Report, June 1996
96-15 * H. Schimpe / M. Staudt: VAREX: An Environment for Validating and Refining Rule Bases
96-16 * M. Jarke / M. Gebhardt, S. Jacobs, H. Nissen: Conflict Analysis Across Heterogeneous Viewpoints: Formalization and Visualization
96-17 M. Jeusfeld / T. X. Bui: Decision Support Components on the Internet
96-18 M. Jeusfeld / M. Papazoglou: Information Brokering: Design, Search and Transformation
96-19 * P. Peters / M. Jarke: Simulating the impact of information flows in networked organizations
96-20 M. Jarke / P. Peters / M. Jeusfeld: Model-driven planning and design of cooperative information systems
96-21 * G. de Michelis / E. Dubois / M. Jarke / F. Matthes / J. Mylopoulos / K. Pohl / J. Schmidt / C. Woo / E. Yu: Cooperative information systems: a manifesto
96-22 * S. Jacobs / M. Gebhardt, S. Kethers, W. Rzasa: Filling HTML forms simultaneously: CoWeb architecture and functionality
96-23 * M. Gebhardt / S. Jacobs: Conflict Management in Design
97-01 Jahresbericht 1996
97-02 J. Faassen: Using full parallel Boltzmann Machines for Optimization
97-03 A. Winter / A. Schürr: Modules and Updatable Graph Views for PROgrammed Graph REwriting Systems
97-04 M. Mohnen / S. Tobies: Implementing Context Patterns in the Glasgow Haskell Compiler
97-05 * S. Gruner: Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge
97-06 M. Nicola / M. Jarke: Design and Evaluation of Wireless Health Care Information Systems in Developing Countries
97-07 P. Hofstedt: Taskparallele Skelette für irregulär strukturierte Probleme in deklarativen Sprachen
97-08 D. Blostein / A. Schürr: Computing with Graphs and Graph Rewriting
97-09 C.-A. Krapp / B. Westfechtel: Feedback Handling in Dynamic Task Nets
97-10 M. Nicola / M. Jarke: Integrating Replication and Communication in Performance Models of Distributed Databases
97-13 M. Mohnen: Optimising the Memory Management of Higher-Order Functional Programs
97-14 R. Baumann: Client/Server Distribution in a Structure-Oriented Database Management System
97-15 G. H. Botorog: High-Level Parallel Programming and the Efficient Implementation of Numerical Algorithms
98-01 * Jahresbericht 1997

98-02 S. Gruner/ M. Nagel / A. Schürr: Fine-grained and Structure-oriented Integration Tools are Needed for Product Development Processes

98-03 S. Gruner: Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr

98-04 * O. Kubitz: Mobile Robots in Dynamic Environments

98-05 M. Leucker / St. Tobies: Truth — A Verification Platform for Distributed Systems

98-07 M. Arnold / M. Erdmann / M. Glinz / P. Haumer / R. Knoll / B. Paech / K. Pohl / J. Ryser / R. Studer / K. Weidenhaupt: Survey on the Scenario Use in Twelve Selected Industrial Projects

98-08 * H. Aust: Sprachverstehen und Dialogmodellierung in natürlichsprachlichen Informationssystemen

98-09 * Th. Lehmann: Geometrische Ausrichtung medizinischer Bilder am Beispiel intraoraler Radiographien

98-10 * M. Nicola / M. Jarke: Performance Modeling of Distributed and Replicated Databases

98-11 * A. Schleicher / B. Westfechtel / D. Jäger: Modeling Dynamic Software Processes in UML

98-12 * W. Appelt / M. Jarke: Interoperable Tools for Cooperation Support using the World Wide Web

98-13 K. Indermark: Semantik rekursiver Funktionsdefinitionen mit Striktheitsinformation

99-01 * Jahresbericht 1998

99-02 * F. Huch: Verifcation of Erlang Programs using Abstract Interpretation and Model Checking — Extended Version

99-03 * R. Gallersdörfer / M. Jarke / M. Nicola: The ADR Replication Manager

99-04 M. Alpuente / M. Hanus / S. Lucas / G. Vidal: Specialization of Functional Logic Programs Based on Needed Narrowing

99-07 Th. Wilke: CTL+ is exponentially more succinct than CTL

99-08 O. Matz: Dot-Depth and Monadic Quantifier Alternation over Pictures

2000-01 * Jahresbericht 1999

2000-02 Jens Vöge / Marcin Jurdziński: A Discrete Strategy Improvement Algorithm for Solving Parity Games

2000-04 Andreas Becks, Stefan Sklorz, Matthias Jarke: Exploring the Semantic Structure of Technical Document Collections: A Cooperative Systems Approach

2000-05 * Mareike Schoop: Cooperative Document Management

2000-06 * Mareike Schoop, Christoph Quix (Ed.): Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling

2000-07 * Markus Mohnen / Pieter Koopman (Eds.): Proceedings of the 12th International Workshop of Functional Languages

2000-08	Thomas Arts / Thomas Noll: Verifying Generic Erlang Client-Server Implementations
2001-01 *	Jahresbericht 2000
2001-02	Benedikt Bollig / Martin Leucker: Deciding LTL over Mazurkiewicz Traces
2001-03	Thierry Cachat: The power of one-letter rational languages
2001-04	Benedikt Bollig / Martin Leucker / Michael Weber: Local Parallel Model Checking for the Alternation free μ-calculus
2001-05	Benedikt Bollig / Martin Leucker / Thomas Noll: Regular MSC languages
2001-06	Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic
2001-07	Martin Grohe / Stefan Wöhrle: An Existential Locality Theorem
2001-08	Mareike Schoop / James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling
2001-09	Thomas Arts / Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs
2001-10	Achim Blumensath: Axiomatising Tree-interpretable Structures
2001-11	Klaus Indermark and Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung
2002-01 *	Jahresbericht 2001
2002-02	Jürgen Giesl / Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems

---

* These reports are only available as a printed version.
Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.