The pros and cons of netcharts

Nicolas BAUDRU & Rémi MORIN

Laboratoire d'Informatique Fondamentale de Marseille
Université de Provence, 39 rue F. Joliot-Curie, F-13453 Marseille cedex 13, France

Abstract. Netcharts have been introduced recently by Mukund et al. in [17]. This new appealing approach to the specification of collections of message sequence charts (MSCs) benefits from a graphical description, a formal semantics based on Petri nets, and an appropriate expressive power. As opposed to high-level MSCs, any regular MSC language is the language of some netchart. Motivated by two open problems raised in [17], we establish in this paper that the questions

(i) whether a given high-level MSC describes some netchart language
(ii) whether a given netchart is equivalent to some high-level MSC
(iii) whether a given netchart describes a regular MSC language

are undecidable. These facts are closely related to our first positive result: We prove that netchart languages are *exactly* the MSC languages that are implementable by message passing automata up to refinement of message contents. Next we focus on FIFO netcharts: The latter are defined as the netcharts whose executions correspond to *all* firing sequences of their low-level Petri net. We show that the questions

(i) whether a netchart is a FIFO netchart
(ii) whether a FIFO netchart describes a regular MSC language
(iii) whether a regular netchart is equivalent to some high-level MSC

are decidable.

Introduction

Message Sequence Charts (MSCs) are a popular model often used for the documentation of telecommunication protocols. They profit by a standardized visual and textual presentation (ITU-T recommendation Z.120 [11]) and are related to other formalisms such as sequence diagrams of UML. An MSC gives a graphical description of communications between processes. It usually abstracts away from the values of variables and the actual contents of messages. However, this formalism can be used at a very early stage of design to detect errors in the specification [10]. In this direction, several studies have already brought up methods and complexity results for the model-checking and implementation of MSCs viewed as a specification language [1–3, 5, 8, 14, 16, 18, 19].

Collections of MSCs are often specified by means of high-level MSCs (HMSCs). The latter can be seen as directed graphs labelled by component MSCs. However such specifications may be unrealistic because this formalism allows for the description of sets of MSCs that correspond to no communicating system. Furthermore in most cases *it is undecidable whether a HMSC describes an implementable language* [1, 14, 8]. In [17], Mukund et al. introduced a new formalism for specifying collections of MSCs: *Netcharts* can be seen as HMSCs with some distributed control whereas HMSCs require implicitly some global control over processes in the system. Basically a netchart

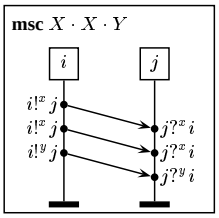

FIG. 1. FIFO MSC

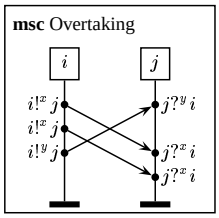

FIG. 2. Non-FIFO MSC

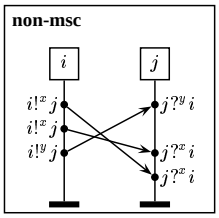

FIG. 3. Degenerate behavior

is a Petri net whose places are labelled by processes and whose transitions are labelled by MSCs. This new approach benefits from a graphical description, a formal semantics, and an appropriate expressive power: As opposed to HMSCs, *all netcharts describe implementable languages*. Our first result completes this relationship and shows that *netcharts describe precisely all implementable languages* (Th. 3.7). This key result allows us to answer negatively to some questions left open in [17].

First we present several comparisons between netcharts and HMSCs. We show that *it is undecidable whether a HMSC describes a netchart language* (Th. 4.7). Conversely, we show also that *it is undecidable whether a netchart language can be described by some HMSC* (Cor. 4.4). Yet as explained below, we can effectively check whether a *regular* netchart is equivalent to some HMSC. These two results follow from the observation that a netchart language corresponds to some HMSC if and only if it describes a finitely generated set of MSCs (Th. 4.3).

In the literature regular MSC languages have attracted a lot of interest. These languages appeared in [2, 18] as a framework where many model-checking problems become decidable. They were investigated later thoroughly and characterized in a logical way in [8, 9]. In particular [8, Th. 4.1] shows how to decide whether a regular set of MSCs is finitely generated. Noteworthy, similarly to high-level *compositional* MSCs [7], any regular MSC language is the language of some netchart [17]. Answering a second open question from [17], another negative consequence of our first result is that *regularity is undecidable for netchart languages* (Cor. 3.8). This is admittedly a major drawback of netcharts.

Motivated by some restrictions considered at some point in [17], we prove in Theorem 5.3 that *regularity is decidable for the subclass of FIFO netcharts*. The latter are defined as those netcharts whose FIFO behaviors correspond exactly to the firing sequences of the underlying low-level Petri net. Theorem 5.3 relies on a difficult and unrecognized result by Lambert [12, Th. 5.2] together with the remark that a netchart language is regular if and only if it requires bounded channel capacities. Additionally we show that *we can check effectively whether a netchart is FIFO* (Th. 5.2) by reduction to the reachability problem in Petri nets [15].

This paper investigates two semantics of netcharts. The FIFO semantics adopted in [17] appears as a restriction of a more general semantics that allows non-FIFO behaviors. In most cases, results extend from the FIFO semantics to the non-FIFO one. However *we exhibit a netchart that is not implementable under the non-FIFO semantics*. To simplify the presentation of our results the non-FIFO semantics is investigated separately in the last section.

1 Background

Message sequence charts (MSCs) are defined by several recommendations that indicate how one should represent them graphically [11]. Examples of MSCs are given in Figures 1 and 2 in which time flows top-down. In this paper we regard MSCs as particular labelled partial orders (or pomsets) following a traditional trend of modeling concurrent executions [6, 13, 20]. This approach allows for applying classical results of Mazurkiewicz trace theory to the framework of MSCs [18, 8, 9, 16, 3].

A *pomset* over an alphabet Σ is a triple $t = (E, \preccurlyeq, \xi)$ where $(E, \preccurlyeq)$ is a finite partial order and ξ is a mapping from E to Σ. A pomset can be seen as an abstraction of an execution of a concurrent system. In this view, the elements e of E are *events* and their label $\xi(e)$ describes the basic action of the system that is performed by the event $e \in E$. Furthermore, the order $\preccurlyeq$ describes the causal dependence between the events.

An *order extension* of a pomset $t = (E, \preccurlyeq, \xi)$ is a pomset $t' = (E, \preccurlyeq', \xi)$ such that $\preccurlyeq \subseteq \preccurlyeq'$. A *linear extension* of t is an order extension that is linearly ordered. It corresponds to a sequential view of the concurrent execution t. Linear extensions of a pomset t over Σ can naturally be regarded as words over Σ. By $\mathrm{LE}(t) \subseteq \Sigma^\star$, we denote the set of linear extensions of a pomset t over Σ.

1.1 FIFO and non-FIFO basic message sequence charts

We present here a formal definition of MSCs. The latter appear as particular pomsets over some alphabet $\Sigma_{\mathcal{I}}^{\Lambda}$ that we introduce first. Let $\mathcal{I}$ be a finite set of processes (also called *instances*) and Λ be a finite set of messages. For any instance $i \in \mathcal{I}$, the alphabet $\Sigma_i^{\Lambda} = \Sigma_{!,i}^{\Lambda} \cup \Sigma_{?,i}^{\Lambda}$ is the disjoint union of the set of *send actions* $\Sigma_{!,i}^{\Lambda} = \{i!^x j \mid j \in \mathcal{I} \setminus \{i\}, x \in \Lambda\}$ and the set of *receive actions* $\Sigma_{?,i}^{\Lambda} = \{i?^x j \mid j \in \mathcal{I} \setminus \{i\}, x \in \Lambda\}$. The alphabets Σ_i^{Λ} are disjoint and we put $\Sigma_{\mathcal{I}}^{\Lambda} = \bigcup_{i \in \mathcal{I}} \Sigma_i^{\Lambda}$. Given an action $a \in \Sigma_{\mathcal{I}}^{\Lambda}$, we denote by $\mathrm{Ins}(a)$ the unique instance i such that $a \in \Sigma_i^{\Lambda}$, that is the particular instance on which each occurrence of action a takes place.

For any pomset $(E, \preccurlyeq, \xi)$ over $\Sigma_{\mathcal{I}}^{\Lambda}$ we denote by $\mathrm{Ins}(e)$ the instance on which the event e occurs: $\mathrm{Ins}(e) = \mathrm{Ins}(\xi(e))$. We say that f *covers* e and we write $e \mathrel{—\!\prec} f$ if $e \prec f$ and $e \prec g \preccurlyeq f$ implies $g = f$. We say that two events e and f are two *matching events* and we write $e \rightsquigarrow f$ if e is the n-th send event $i!^x j$ and f is the n-th receive event $j?^x i$: In other words, we put $e \rightsquigarrow f$ if there are two instances i and j and some message $x \in \Lambda$ such that $\xi(e) = i!^x j$, $\xi(f) = j?^x i$ and $\mathrm{Card}\{e' \in E \mid \xi(e') = i!^x j \wedge e' \preccurlyeq e\} = \mathrm{Card}\{f' \in E \mid \xi(f') = j?^x i \wedge f' \preccurlyeq f\}$.

DEFINITION 1.1. *A* basic message sequence chart (MSC) *over the set of messages Λ is a pomset $M = (E, \preccurlyeq, \xi)$ over $\Sigma_{\mathcal{I}}^{\Lambda}$ such that*
M_1*:* $\forall e, f \in E$*:* $\mathrm{Ins}(e) = \mathrm{Ins}(f) \Rightarrow (e \preccurlyeq f \vee f \preccurlyeq e)$
M_2*:* $\forall e \in E, \exists f \in E, e \rightsquigarrow f \vee f \rightsquigarrow e$
M_3*:* $e \rightsquigarrow f \Rightarrow e \preccurlyeq f$
M_4*:* $[e \mathrel{—\!\prec} f \wedge \mathrm{Ins}(e) \neq \mathrm{Ins}(f)] \Rightarrow e \rightsquigarrow f$.

By M_1, events occurring on the same instance are linearly ordered: In particular nondeterministic choice cannot be described within a basic MSC. Condition M_2 makes

sure that each receive event matches some send event and conversely. Thus there is no duplication of messages within the channels and M_2 formalizes partly the reliability of the channels. Following the recommendation Z.120, we allow *overtaking* (Fig. 2) but forbid any reversal of the order in which two identical messages m sent from i to j are received by j (Fig. 3). Now M_3 formalizes simply that the receipt of any message will occur after the corresponding send event. Finally, by M_4, causality in M consists only in the linear dependency on each instance and the ordering of pairs of matching events. The set of all basic MSCs is denoted by $\mathrm{b}\mathbb{MSC}$. Note here that if two basic MSCs share some linear extension then they are equal. We denote by $\mathrm{Ins}(M)$ the set of active instances of an MSC M: $\mathrm{Ins}(M) = \{i \in \mathcal{I} \mid \exists e \in E, \mathrm{Ins}(e) = i\}$.

In Figure 2, the basic MSC exhibits some *overtaking* of message y above two messages x. A basic MSC satisfies *the FIFO requirement* if it shows no overtaking, that is, the messages from one instance to another are delivered in the order they are sent (Fig. 1). Non-FIFO basic MSCs allow for specifying scenarios that use several channels (or message types) between pairs of processes (Fig. 2). A more critical situation is illustrated by Figure 3. In this drawing, one message overtakes another one with the same content: In this paper, differently from [4] we forbid this kind of behaviors.

1.2 Petri nets

Let us now recall the definition of a Petri net and some usual notations. A *Petri net* is a triple $\mathcal{P} = (P, T, F)$ where P is a set of *places*, T is a set of *transitions* such that $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ is a *flow relation*. We shall use the following usual notations. For all $x \in P \cup T$, we put ${}^\bullet x = \{y \in P \cup T \mid (y, x) \in F\}$ and $x^\bullet = \{y \in P \cup T \mid (x, y) \in F\}$. Clearly, for all transitions t, ${}^\bullet t$ and $t^\bullet$ are sets of places, and conversely for all places $p \in P$, ${}^\bullet p$ and $p^\bullet$ are both sets of transitions. A *marking* $\mathfrak{m}$ of $\mathcal{P}$ is a multiset of places $\mathfrak{m} \in P^{\mathbb{N}}$. A transition t is *enabled* at $\mathfrak{m} \in P^{\mathbb{N}}$ if $\mathfrak{m}(p) \geqslant 1$ for all $p \in {}^\bullet t$. In this case, we write $\mathfrak{m} \,[t\rangle\, \mathfrak{m}'$ where the marking $\mathfrak{m}'$ is defined by $\mathfrak{m}'(p) = \mathfrak{m}(p) - 1$ if $p \in {}^\bullet t \setminus t^\bullet$, $\mathfrak{m}'(p) = \mathfrak{m}(p) + 1$ if $p \in t^\bullet \setminus {}^\bullet t$, and $\mathfrak{m}'(p) = \mathfrak{m}(p)$ otherwise.

In this paper, we consider Petri nets provided with an *initial marking* $\mathfrak{m}_{\mathrm{in}}$ and a *finite* set of final markings $\mathfrak{F}$. An *execution sequence* is a word $u = t_1 \ldots t_n \in T^\star$ such that there are markings $\mathfrak{m}_0$,..., $\mathfrak{m}_n$ satisfying $\mathfrak{m}_0 = \mathfrak{m}_{\mathrm{in}}$, $\mathfrak{m}_n \in \mathfrak{F}$ and $\mathfrak{m}_{k-1} \,[t_k\rangle\, \mathfrak{m}_k$ for all integers $k \in [1, n]$. The language $L(\mathcal{P})$ consists of all execution sequences of $\mathcal{P}$.

1.3 Netcharts

A netchart is basically a Petri net whose places are labelled by instances and whose transitions are labelled by FIFO basic MSCs. Similarly to Petri nets, netcharts admit an intuitive visual representation: Examples of netcharts are given in Fig. 4, 7, and 9.

DEFINITION 1.2. *A* netchart *over Λ consists of a Petri net $(P, T, F, \mathfrak{m}_{\mathrm{in}}, \mathfrak{F})$ and two mappings* $\mathrm{Ins} : P \to \mathcal{I}$ *and* $\mathcal{M} : T \to \mathrm{b}\mathbb{MSC}$ *such that* Ins *associates some instance* $\mathrm{Ins}(p)$ *to each place p and $\mathcal{M}$ associates a FIFO basic MSC $\mathcal{M}(t)$ over the set of messages Λ to each transition $t \in T$. Three conditions are required for such a structure to be a netchart:*

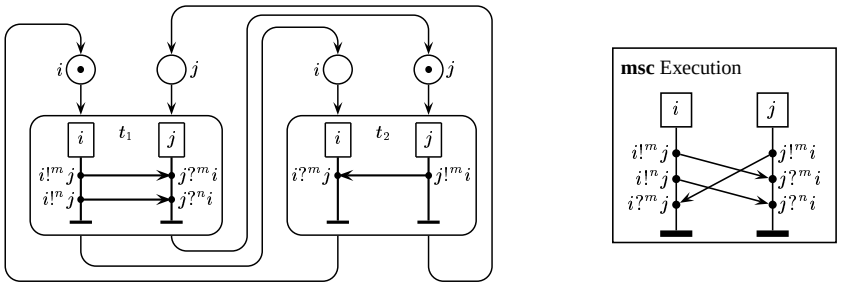

FIG. 4. A netchart $\mathcal{N}$ and a corresponding MSC

N_1: *For each instance $i \in \mathcal{I}$, the places located on instance i contain exactly one token in the initial marking, i.e.* $\sum_{\mathrm{Ins}(p)=i} \mathfrak{m}_{\mathrm{in}}(p) = 1$.

N_2: *For each transition t and each active instance $i \in \mathrm{Ins}(\mathcal{M}(t))$, there is exactly one place $p \in {}^\bullet t$ such that* $\mathrm{Ins}(p) = i$ *and there is exactly one place $p \in t^\bullet$ such that* $\mathrm{Ins}(p) = i$.

N_3: *For each transition t and each place $p \in {}^\bullet t \cup t^\bullet$, the instance associated to p is active in $\mathcal{M}(t)$:* $\mathrm{Ins}(p) \in \mathrm{Ins}(\mathcal{M}(t))$.

Observe here that the last requirement N_3 implies that ${}^\bullet t \cup t^\bullet$ is empty as soon as $\mathcal{M}(t)$ is the empty MSC. However Axiom N_3 plays actually no rôle in the semantics of netcharts and it could be removed for simplification's sake.

2 Semantics of netcharts

In this section we fix a netchart $\mathcal{N} = ((P, T, F, \mathfrak{m}_{\mathrm{in}}, \mathfrak{F}), \mathrm{Ins}, \mathcal{M})$ over the set of messages Λ and define formally its behaviors. The semantics of $\mathcal{N}$ consists of basic MSCs over Λ (Fig. 4). The latter are derived from the basic MSCs that represent the execution sequences of some low-level Petri net $\mathcal{P}_{\mathcal{N}}$ (Fig. 4 and 6). Actually, the execution sequences of $\mathcal{P}_{\mathcal{N}}$ use a *refined set of messages* Λ° and MSCs of $\mathcal{N}$ are obtained by projection of messages from Λ° onto Λ.

2.1 From MSCs to Petri nets

The construction of the low-level Petri net $\mathcal{P}_{\mathcal{N}}$ starts with the translation of each transition $t \in T$ with component MSC $\mathcal{M}(t) = (E, \preccurlyeq, \xi)$ into some Petri net $\mathcal{P}_t = (P_t, T_t, F_t)$. This natural operation is depicted in Fig. 5.

This construction need to regard each component MSC $M = (E, \preccurlyeq, \xi)$ as a dag (direct acyclic graph) denoted by $(E, \prec, \xi)$. For any instance $i \in \mathcal{I}$ we let $\preccurlyeq_i$ be the restriction of $\preccurlyeq$ to events located on instance i. Then $e \mathrel{—\!\!\prec_i} f$ if e occurs immediately before f on instance i. Then the binary relation $\prec$ consists of pairs of matching events together with pairs of covering events w.r.t. $\preccurlyeq_i$.

DEFINITION 2.1. *The* MSC dag *of a basic MSC $M = (E, \preccurlyeq, \xi)$ is a labelled directed acyclic graph $(E, \prec, \xi)$ such that $e \prec f$ if $e \rightsquigarrow f$ or $e \mathrel{—\!\!\prec_i} f$ for some instance $i \in \mathcal{I}$.*

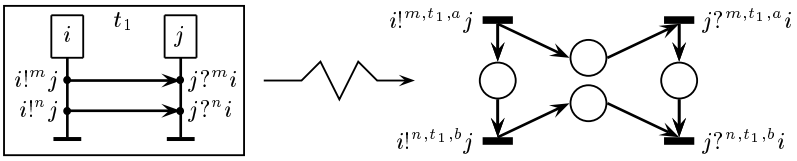

FIG. 5. From transition t_1 to Petri net $\mathcal{P}_{t_1}$

Clearly we can recover the basic MSC from its MSC dag. The reason for this is that $\prec\!\!\!\cdot \subseteq \prec$ hence $\preccurlyeq$ is simply the reflexive and transitive closure of $\prec$. That is why we will identify a basic MSC with its corresponding MSC dag.

We can now formalize how each component MSC $\mathcal{M}(t) = (E, \prec, \xi)$ is translated into some Petri net $\mathcal{P}_t = (P_t, T_t, F_t)$. First, the places P_t are identified with pairs from $\prec$. Second the transitions T_t are identified with some send or receive actions over a new set of messages from $\Lambda \times T \times P_t$. Formally, we put $P_t = \prec$ and $T_t = \{i!^{m,t,(e,f)} j, j?^{m,t,(e,f)} i \mid (e,f) \in \prec \wedge\, \xi(e) = i!^m j \wedge \xi(f) = j?^m i\}$.

Note that the translation from the basic MSC $\mathcal{M}(t)$ into the Petri net $\mathcal{P}_t$ is one-to-one: We will be able to recover the basic MSC $\mathcal{M}(t)$ from the Petri net $\mathcal{P}_t$. For this, we let ρ be the mapping from T_t to E such that $\rho(i!^{m,t,(e,f)} j) = e$ and $\rho(j?^{m,t,(e,f)} i) = f$. To complete the definition of $\mathcal{P}_t$ we choose a flow relation F_t in accordance with the causality relation $\prec$ of $\mathcal{M}(t)$: We put

$$F_t = \{(r,(e,f)) \in T_t \times P_t \mid \rho(r) = e\} \cup \{((e,f),r) \in P_t \times T_t \mid \rho(r) = f\}.$$

The transitions of the Petri net $\mathcal{P}_t = (P_t, T_t, F_t)$ will be connected to places of $\mathcal{N}$ by means of the following connection relation:

$$\begin{aligned} F'_t = {} & \{(p,r) \in P \times T_t \mid p \in {}^\bullet t \wedge {}^\bullet r = \emptyset \wedge \mathrm{Ins}(\rho(r)) = \mathrm{Ins}(p)\} \\ & \cup \{(r,p) \in T_t \times P \mid p \in t^\bullet \wedge r^\bullet = \emptyset \wedge \mathrm{Ins}(\rho(r)) = \mathrm{Ins}(p)\}. \end{aligned}$$

2.2 Low-level Petri net

Now, in order to build the low-level Petri net $\mathcal{P}_\mathcal{N}$ of the netchart $\mathcal{N}$, we replace each transition $t \in T$ of $\mathcal{N}$ by its corresponding Petri net $\mathcal{P}_t$ as shown in Fig. 6.

The low-level Petri net $\mathcal{P}_\mathcal{N} = (P_\mathcal{N}, T_\mathcal{N}, F_\mathcal{N}, \mathfrak{m}_{\mathrm{in}}, \mathfrak{F}_\mathcal{N})$ is built as follows. First, the set of places $P_\mathcal{N}$ collects the places of $\mathcal{N}$ and the places of all $\mathcal{P}_t$: $P_\mathcal{N} = \bigcup_{t \in T} P_t \cup P$. Second, the set of transitions collects all transitions of all $\mathcal{P}_t$: $T_\mathcal{N} = \bigcup_{t \in T} T_t$. Now the flow relation consists of the flow relation F_t of each $\mathcal{P}_t$ together with the connection relations F'_t: $F_\mathcal{N} = \bigcup_{t \in T} F_t \cup F'_t$. The initial marking of $\mathcal{P}$ is the one of $\mathcal{N}$: The new places $p \in P_\mathcal{N} \setminus P$ are initially empty. Similarly a marking $\mathfrak{m}$ of $\mathcal{P}_\mathcal{N}$ is final if the restriction of $\mathfrak{m}$ to the places of $\mathcal{N}$ is a final marking of $\mathcal{N}$ and if all other places are empty: $\mathfrak{F}_\mathcal{N} = \{\mathfrak{m} \in P^\mathbb{N} \mid \mathfrak{m}_{|P} \in \mathfrak{F} \wedge \mathfrak{m}_{|P_\mathcal{N} \setminus P} = 0\}$.

Any execution sequence $u \in L(\mathcal{P}_\mathcal{N})$ of the low-level Petri net leads from the initial marking to some final marking for which all places from P_t are empty. Moreover u is actually a linear extension of a unique basic MSC.

DEFINITION 2.2. *The MSC language $\mathcal{L}_{\mathrm{fifo}}(\mathcal{P}_\mathcal{N})$ consists of the FIFO basic MSCs M such that at least one linear extension of M is an execution sequence of $\mathcal{P}_\mathcal{N}$.*

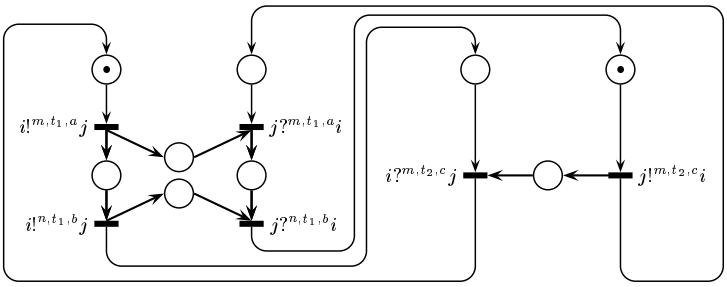

FIG. 6. The low-level Petri net $\mathcal{P}_{\mathcal{N}}$ associated to the netchart $\mathcal{N}$ of Fig. 4

Interestingly, similarly to a property observed with message passing automata (Def. 3.2 below), it can be easily shown that a basic MSC M belongs to $\mathcal{L}_{\text{fifo}}(\mathcal{P}_{\mathcal{N}})$ if and only if *all* linear extensions of M are execution sequences of $\mathcal{P}_{\mathcal{N}}$. Noteworthy it can happen that an execution sequence of the low-level Petri net $\mathcal{P}_{\mathcal{N}}$ corresponds to a *non-FIFO* MSC (see e.g. [17, Fig. 5] or Fig. 7). Following [17], we focus on FIFO behaviors and neglect this kind of execution sequences here. We will investigate a non-FIFO semantics of netcharts in the last section only.

2.3 Set of MSCs associated to some netchart

Recall now that MSCs from $\mathcal{L}_{\text{fifo}}(\mathcal{P}_{\mathcal{N}})$ use a refined set of messages Λ° that consists of triples (m,t,a) where $m \in \Lambda$, $t \in T$, and $a \in P_t$. We let $\pi^\circ : \Lambda^\circ \to \Lambda$ denote the labelling that associates the message $m \in \Lambda$ to each triple $(m,t,a) \in \Lambda^\circ$. This labelling extends to a function that maps actions of $\Sigma_{\mathcal{I}}^{\Lambda^\circ}$ onto actions of $\Sigma_{\mathcal{I}}^{\Lambda}$ in a natural way. Furthermore this mapping extends in the obvious way from the FIFO basic MSCs over Λ° to the FIFO basic MSCs over Λ. The semantics of the netchart $\mathcal{N}$ is defined from the semantics of its low-level Petri net $\mathcal{P}_{\mathcal{N}}$ by means of the labelling π°.

DEFINITION 2.3. *The MSC language $\mathcal{L}_{\text{fifo}}(\mathcal{N})$ is the set of FIFO basic MSCs obtained from an MSC of its low-level Petri net by the labelling π°: $\mathcal{L}_{\text{fifo}}(\mathcal{N}) = \pi^\circ(\mathcal{L}_{\text{fifo}}(\mathcal{P}_{\mathcal{N}}))$.*

We stress here that π° maps FIFO basic MSCs onto FIFO basic MSCs. The situation with non-FIFO basic MSCs may be more complicated as we will see in the last section.

3 Netcharts vs. implementable languages

In this section, we study how netcharts relate to communicating systems. We consider the set of channels $\mathcal{K}$ that consists of all triples $(i,j,x) \in \mathcal{I} \times \mathcal{I} \times \Lambda$: A *channel state* is then formalized by a map $\chi : \mathcal{K} \to \mathbb{N}$ that describes the queues of messages within the channels at some stage of an execution. The *empty channel state* χ_0 is such that each channel maps to 0.

DEFINITION 3.1. *A* message passing automaton (MPA) $\mathcal{S}$ *over Λ consists of a family of local components $(\mathcal{A}_i)_{i\in\mathcal{I}}$ and a subset of global final states F such that each*

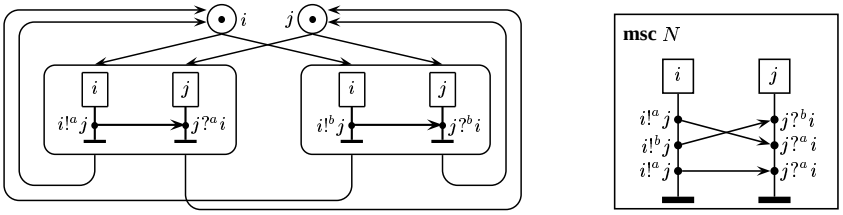

FIG. 7. Netchart $\mathcal{N}_1$ and some non-FIFO behaviour $N \notin \mathcal{L}_{\text{fifo}}(\mathcal{N})$

component $\mathcal{A}_i$ *is a transition system* $(Q_i, \imath_i, \longrightarrow_i)$ *over* Σ_i^{Λ} *where* Q_i *is a finite set of* i-local states*, with initial state* $\imath_i \in Q_i$*,* $\longrightarrow_i \subseteq (Q_i \times \Sigma_i^{\Lambda} \times Q_i)$ *is the* i-local transition relation *and* $F \subseteq \left(\prod_{i\in\mathcal{I}} Q_i\right) \times \{\chi_0\}$.

3.1 Semantics of MPA

A *global state* is a pair (s, χ) where $s \in \prod_{i\in\mathcal{I}} Q_i$ is a tuple of local states and χ is a channel state. The *initial global state* is the pair $\imath = (s, \chi)$ such that $s = (\imath_i)_{i\in\mathcal{I}}$ and $\chi = \chi_0$ is the empty channel state. The *system of global states* associated to $\mathcal{S}$ is the transition system $\mathcal{A}_{\mathcal{S}} = (Q, \imath, \longrightarrow)$ over $\Sigma_{\mathcal{I}}^{\Lambda}$ where $Q = \prod_{i\in\mathcal{I}} Q_i \times \mathbb{N}^{\mathcal{K}}$ is the set of global states and the global transition relation $\longrightarrow \subseteq Q \times \Sigma_{\mathcal{I}}^{\Lambda} \times Q$ satisfies:

- for all $(i,j,m) \in \mathcal{K}$, $(q_k)_{k\in\mathcal{I}}, \chi \xrightarrow{i!^m j} (q'_k)_{k\in\mathcal{I}}, \chi'$ if
  1. $q_i \xrightarrow{i!^m j}_i q'_i$ and $q'_k = q_k$ for all $k \in \mathcal{I} \setminus \{i\}$,
  2. $\chi'(i,j,m) = \chi(i,j,m) + 1$ and $\chi(x) = \chi'(x)$ for all $x \in \mathcal{K} \setminus \{(i,j,m)\}$;
- for all $(i,j,m) \in \mathcal{K}$, $(q_k)_{k\in\mathcal{I}}, \chi \xrightarrow{j?^m i} (q'_k)_{k\in\mathcal{I}}, \chi'$ if
  1. $q_j \xrightarrow{j?^m i}_j q'_j$ and $q'_k = q_k$ for all $k \in \mathcal{I} \setminus \{j\}$,
  2. $\chi(i,j,m) = 1 + \chi'(i,j,m)$ and $\chi(x) = \chi'(x)$ for all $x \in \mathcal{K} \setminus \{(i,j,m)\}$.

As usual with transition systems, for any $u = a_1 ... a_n \in \Sigma_{\mathcal{I}}^{\Lambda\star}$, we write $q \xrightarrow{u} q'$ if there are some global states $q_0, ..., q_n \in Q$ such that $q_0 = q$, $q_n = q'$ and for all $r \in [1, n]$, $q_{r-1} \xrightarrow{a_r} q_r$. An *execution sequence* of $\mathcal{S}$ is a word $u \in \Sigma_{\mathcal{I}}^{\Lambda\star}$ such that $\imath \xrightarrow{u} q$ for some global final state $q \in F$.

Consider now an MPA $\mathcal{S}$ with components $(\mathcal{A}_i)_{i\in\mathcal{I}}$ and global final states F. Any execution sequence $u \in \Sigma_{\mathcal{I}}^{\star}$ is a linear extension of a (unique) basic MSC.

DEFINITION 3.2. *The* language $\mathcal{L}_{\text{fifo}}(\mathcal{S})$ *consists of the FIFO basic MSCs* M *such that at least one linear extension of* M *is an execution sequence of* $\mathcal{S}$*.*

Noteworthy, it can be easily shown that a basic MSC M belongs to $\mathcal{L}_{\text{fifo}}(\mathcal{S})$ iff *all* linear extensions of M are execution sequences of $\mathcal{S}$. We say that a language $\mathcal{L} \subseteq \text{bMSC}$ is *realizable* if there exists some MPA $\mathcal{S}$ such that $\mathcal{L} = \mathcal{L}_{\text{fifo}}(\mathcal{S})$.

EXAMPLE 3.3. Consider the netchart $\mathcal{N}_1$ depicted in Figure 7 for which the initial marking is the single final marking. Its language $\mathcal{L}_{\text{fifo}}(\mathcal{N}_1)$ is the set of all basic MSCs that consist only of messages a and b exchanged from i to j in a FIFO manner. Clearly, the language $\mathcal{L}_{\text{fifo}}(\mathcal{N}_1)$ is realizable.

3.2 Implementation of MSC languages

As observed in [1], there are finite sets of FIFO basic MSCs that are not realizable. For this reason, it is natural to relax the notion of realization. In [9], Henriksen et al. suggested to allow some refinements of message contents as follows.

DEFINITION 3.4. *Let $\mathcal{L} \subseteq \mathrm{bMSC}$ be an MSC language over the set of messages Λ. We say that $\mathcal{L}$ is* implementable *if there are some MPA $\mathcal{S}$ over some set of messages Λ' and some labelling $\lambda : \Lambda' \to \Lambda$ such that $\mathcal{L} = \lambda(\mathcal{L}_{\mathrm{fifo}}(\mathcal{S}))$.*

Note here that any implementable language consists of FIFO basic MSCs only because $\lambda(M)$ is FIFO as soon as M is FIFO.

As the next result shows, the refinement of message contents by means of labellings helps the synthesis of MPAs from sets of scenarios. As opposed to the restrictive approach studied in [1, 14] which sticks to the specified set of message contents, labellings allow for the implementation of any finite set of basic MSCs. Actually the refinement of messages allows for the implementation of any *regular* set of FIFO basic MSCs. Recall here that an MSC language $\mathcal{L} \subseteq \mathrm{bMSC}$ is called *regular* if the set of corresponding linear extensions $\mathrm{LE}(\mathcal{L}) = \{\mathrm{LE}(M) \mid M \in \mathcal{L}\}$ is a regular set of words.

THEOREM 3.5. *[9, Th. 3.4] All regular sets of FIFO basic MSCs are implementable.*

One main property of netcharts is the following.

THEOREM 3.6. *[17] For any netchart $\mathcal{N}$, $\mathcal{L}_{\mathrm{fifo}}(\mathcal{N})$ is implementable.*

Note that Theorem 3.6 fails if we forbid refinements, that is if we require that $\mathcal{L}_{\mathrm{fifo}}(\mathcal{N}) = \mathcal{L}_{\mathrm{fifo}}(\mathcal{S})$. The reason for this is again that there are finite sets of FIFO basic MSCs that are not realizable while they are netchart languages.

3.3 From message passing automata to netcharts

In [17, Th. 6], it is shown that any regular MSC language is a netchart language. However the converse fails: There are netchart languages that are not regular (see e.g. Example 3.3). Our first result characterizes the expressive power of netcharts and establishes the converse of Theorem 3.6.

THEOREM 3.7. *Any implementable language is the MSC language of some netchart whose component MSCs consist only of a pair of matching events.*

We stress that Theorem 3.7 is effective: For any MPA $\mathcal{S}$ over the set of messages Λ' and any labelling $\lambda : \Lambda' \to \Lambda$, we can build a netchart $\mathcal{N}$ such that $\mathcal{L}_{\mathrm{fifo}}(\mathcal{N}) = \lambda(\mathcal{L}_{\mathrm{fifo}}(\mathcal{S}))$. Theorem 3.7 subsumes [17, Th. 6] because all regular MSC languages are implementable (Th. 3.5) and there are implementable languages that are not regular (Ex. 3.3). The proof of Theorem 3.7 is rather tedious. It differs from the proof of [17, Th. 6] in that we do not assume the implementable language $\lambda(\mathcal{L}_{\mathrm{fifo}}(\mathcal{S}))$ to be regular.

Theorem 3.7 shows that the expressivity of netcharts coincides with the expressivity of MPAs up to labellings. This leads us to a first answer to questions from [17].

COROLLARY 3.8. *It is undecidable whether a netchart language is regular.*

Proof. We observe first that it is undecidable whether the language of some given MPA is regular. More precisely, similarly to the proof of [19, Prop. 7], for any instance of Post's Corresponding Problem, we build some MPA $\mathcal{S}$ such that the instance has a solution iff $\mathcal{L}_{\mathrm{fifo}}(\mathcal{S})$ is not empty and in this case $\mathcal{L}_{\mathrm{fifo}}(\mathcal{S})$ is not regular. Now the proof follows from the effectiveness of Th. 3.7 with a labelling $\lambda = \mathrm{Id} : \Lambda \to \Lambda$. ∎

4 Netcharts vs. high-level message sequence charts

Let us now recall how one can build high-level MSCs from basic MSCs. First, the *asynchronous concatenation* of two basic MSCs $M_1 = (E_1, \preccurlyeq_1, \xi_1)$ and $M_2 = (E_2, \preccurlyeq_2, \xi_2)$ is the basic MSC $M_1 \cdot M_2 = (E, \preccurlyeq, \xi)$ where $E = E_1 \uplus E_2$, $\xi = \xi_1 \cup \xi_2$ and the partial order $\preccurlyeq$ is the transitive closure of $\preccurlyeq_1 \cup \preccurlyeq_2 \cup \{(e_1, e_2) \in E_1 \times E_2 \mid \mathrm{Ins}(e_1) = \mathrm{Ins}(e_2)\}$. This concatenation allows for the composition of specifications in order to describe infinite sets of basic MSCs: We obtain high-level message sequence charts as rational expressions, following thus the usual algebraic approach that we recall next.

4.1 Rational sets of MSCs

For any subsets $\mathcal{L}$ and $\mathcal{L}'$ of bMSC, the *product* of $\mathcal{L}$ by $\mathcal{L}'$ is $\mathcal{L} \cdot \mathcal{L}' = \{x \cdot x' \mid x \in \mathcal{L} \wedge x' \in \mathcal{L}'\}$. We let 1 denote the empty basic MSC and we put $\mathcal{L}^0 = \{1\}$. For any $n \in \mathbb{N}$, $\mathcal{L}^{n+1} = \mathcal{L}^n \cdot \mathcal{L}$; then the *iteration* of $\mathcal{L}$ is $\mathcal{L}^\star = \bigcup_{n \in \mathbb{N}} \mathcal{L}^n$. It is also denoted $\langle \mathcal{L} \rangle$. A language $\mathcal{L} \subseteq \mathrm{bMSC}$ is *finitely generated* if there is a finite subset Γ of bMSC such that $\mathcal{L} \subseteq \langle \Gamma \rangle$. A subset of bMSC is *rational* if it can be obtained from the finite subsets of bMSC by means of unions, products and iterations. Any rational language is finitely generated.

DEFINITION 4.1. *A* high-level message sequence chart (HMSC) *is a rational expression of basic MSCs, that is, an expression built from finite sets of basic MSCs by use of union ($+$), product ($\cdot$) and iteration ($\star$).*

We follow here the approach adopted e.g. in [1, 2, 5, 8, 14, 19] where HMSCs are however often flattened into *message sequence graphs*. The set of MSCs corresponding to some HMSC $\mathcal{H}$ is denoted by $\mathcal{L}_{\mathcal{H}}$.

EXAMPLE 4.2. Consider again the two components MSCs A and B of the netchart $\mathcal{N}_1$ depicted in Fig. 7. As already observed in Example 3.3, the language $\mathcal{L}_{\mathrm{fifo}}(\mathcal{N}_1)$ is the set of all FIFO basic MSCs that consist only of messages a and b exchanged from i to j. This language corresponds to the HMSC $(A + B)^\star$.

4.2 For netchart languages: finitely generated means rational

As already observed in [17, Fig. 6], there are netcharts whose languages are not finitely generated. Clearly these netchart languages are not rational. We show here that it is undecidable whether a given netchart language is described by some HMSC (Cor. 4.4). As a first step, the next result shows that being finitely generated is sufficient for a netchart language to be rational.

THEOREM 4.3. *For any netchart $\mathcal{N}$, $\mathcal{L}_{\mathrm{fifo}}(\mathcal{N})$ is finitely generated iff it is the language of some HMSC.*

Proof. Let Γ be a finite set of basic MSCs over Λ such that $\mathcal{L}_{\mathrm{fifo}}(\mathcal{N}) \subseteq \langle\Gamma\rangle$. From Theorem 3.6, we can build some MPA $\mathcal{S}$ over a refined set of messages Λ' such that $\mathcal{L}_{\mathrm{fifo}}(\mathcal{N}) = \lambda(\mathcal{L}_{\mathrm{fifo}}(\mathcal{S}))$ for some $\lambda : \Lambda' \to \Lambda$. Let Γ' be the subset of FIFO basic MSCs M over Λ' such that $\lambda(M) \in \Gamma$. Then $\mathcal{L}_{\mathrm{fifo}}(\mathcal{N}) = \lambda(\mathcal{L}_{\mathrm{fifo}}(\mathcal{S}) \cap \langle\Gamma'\rangle)$. Since $\mathcal{L}_{\mathrm{fifo}}(\mathcal{S}) \cap \langle\Gamma'\rangle$ is recognizable and finitely generated, it is described by some globally cooperative HMSC [16, Th. 2.3]. ∎

In [19, Prop. 7], it was shown that it is undecidable whether the language of some given MPA is finitely generated. Since the language of any MPA is also the language of some netchart that we can effectively build (Th. 3.7), we obtain easily a first corollary of Th. 4.3.

COROLLARY 4.4. *Given some netchart $\mathcal{N}$, it is undecidable whether $\mathcal{L}_{\mathrm{fifo}}(\mathcal{N})$ is described by some HMSC.*

Thus, it is undecidable whether a netchart language is rational. In the end of this section we show that the opposite question is undecidable, too (Th. 4.7).

4.3 From HMSCs to netcharts

Let us now relate the notions of regularity and channel-boundedness in the framework of netcharts. Recall first that the channel-width of some basic MSC M is the maximal number of messages that may be sent in a channel but not received along some linear extension of M. Formally, the *channel-width* of M is

$$\max_{(i,j,x)\in\mathcal{K}}\{|v|_{i!^x j} - |v|_{j?^x i} \mid u \in \mathrm{LE}(M) \wedge v \text{ is a prefix of } u\}.$$

A language of basic MSCs $\mathcal{L} \subseteq \mathrm{b}\mathbb{MSC}$ is called *channel-bounded* by an integer B if each basic MSC of $\mathcal{L}$ has a channel-width at most B. It was observed in [8] that each regular MSC language is channel-bounded. In general the converse fails. However, for netchart languages the two notions coincide as the next elementary observation shows.

LEMMA 4.5. *Let $\mathcal{N}$ be a netchart. The language $\mathcal{L}_{\mathrm{fifo}}(\mathcal{N})$ is regular iff it is channel-bounded.*

This result may be seen as a direct consequence of Theorem 3.6 although it is much easier to prove it directly. With the help of Lemma 4.5 and Th. 3.5 we can now easily characterize which channel-bounded FIFO HMSCs describe a netchart language.

THEOREM 4.6. *Let $\mathcal{H}$ be a HMSC such that $\mathcal{L}_{\mathcal{H}}$ is channel-bounded and FIFO. Then $\mathcal{L}_{\mathcal{H}}$ is regular iff $\mathcal{L}_{\mathcal{H}}$ is a netchart language.*

By means of the proof technique of [8, Th. 4.6], we can show easily that it is undecidable whether a channel-bounded FIFO HMSC describes a regular language. As a consequence, we get the following negative result.

THEOREM 4.7. *It is undecidable whether the language of some given HMSC can be described by some netchart. This holds even if we restrict to HMSCs that describe channel-bounded languages.*

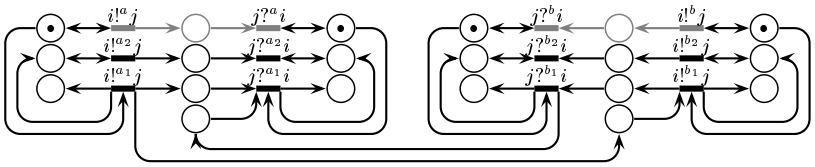

FIG. 8. Construction to decide whether some netchart is FIFO

5 Two positive results for FIFO netcharts

We have proved in Cor. 3.8 that checking regularity of $\mathcal{L}_{\mathrm{fifo}}(\mathcal{N})$ is undecidable. To cope with this negative result, we introduce a subclass of netcharts for which regularity becomes decidable. This restriction was also considered at some point in [17].

DEFINITION 5.1. *A netchart $\mathcal{N}$ is called* FIFO *if any execution sequence of its low-level Petri net is a linear extension of some FIFO basic MSC.*

Figure 7 shows a non-FIFO netchart whereas Figure 4 shows a FIFO netchart. Interestingly, this subclass of netcharts is decidable and regularity is decidable in this subclass.

THEOREM 5.2. *It is decidable whether a netchart is a FIFO netchart.*

Proof. We consider two distinct messages a and b from Λ°. These two messages are involved in four transitions $i!^a j$, $j?^a i$, $i!^b j$ and $j?^b i$ in the low-level net $\mathcal{P}_\mathcal{N}$. In order to check whether b can overtake a in some execution sequence of $\mathcal{P}_\mathcal{N}$, we build a new Petri net from $\mathcal{P}_\mathcal{N}$ by adding some places and some transitions. More precisely, around the four transitions related to a and b and the two corresponding places depicted in gray in Fig. 8, we add 8 new transitions $i!^{a_k} j$, $j?^{a_k} i$, $i!^{b_k} j$ and $j?^{b_k} i$ ($k \in \{1, 2\}$) and 18 new places drawn in black in Fig. 8. Observe that the new transition $i!^{a_1} j$ can be executed at most once; moreover in this case a token is put in the new place at its left. A similar observation holds for $j?^{a_1} i$, $i!^{b_1} j$, and $j?^{b_1} i$. Observe also that $i!^{b_1} j$ can be executed only after $i!^{a_1} j$ whereas $j?^{a_1} i$ can be executed only after $j?^{b_1} i$. Now each arc from a place p to the transition $i!^a j$ is copied into an arc from p to $i!^{a_1} j$ and another arc from p to $i!^{a_2} j$. We proceed similarly with places in $i!^a j^\bullet$ and with the transition $i!^b j$. Now we claim that some MSC of $\mathcal{P}_\mathcal{N}$ shows some overtaking of b over a iff the new Petri net admits an execution sequence that involves the transitions $i!^{a_1} j$ and $i!^{b_1} j$. We can check the existence of such an execution sequence by reachability analysis [15]. ■

THEOREM 5.3. *Regularity of $\mathcal{L}_{\mathrm{fifo}}(\mathcal{N})$ is decidable for FIFO netcharts.*

Proof. By Lemma 4.5, we have to check whether $\mathcal{L}_{\mathrm{fifo}}(\mathcal{N})$ is channel-bounded. Since $\mathcal{N}$ has finitely many final states, we may assume that $\mathcal{N}$ has a unique final marking. Since $\mathcal{L}_{\mathrm{fifo}}(\mathcal{N}) = \pi^\circ(\mathcal{L}_{\mathrm{fifo}}(\mathcal{P}_\mathcal{N}))$, $\mathcal{L}_{\mathrm{fifo}}(\mathcal{N})$ is channel-bounded iff $\mathcal{L}_{\mathrm{fifo}}(\mathcal{P}_\mathcal{N})$ is channel-bounded. Moreover $\mathcal{L}_{\mathrm{fifo}}(\mathcal{P}_\mathcal{N})$ is channel-bounded iff it is regular. Since $\mathcal{N}$ is FIFO, this holds iff the set of all execution sequences of $\mathcal{P}_\mathcal{N}$ is regular. This question is decidable as shown by Lambert [12, Th. 5.2]. An alternative to this proof is to apply a recent and independent work by Wimmel [21] which is also based on [12]. ■

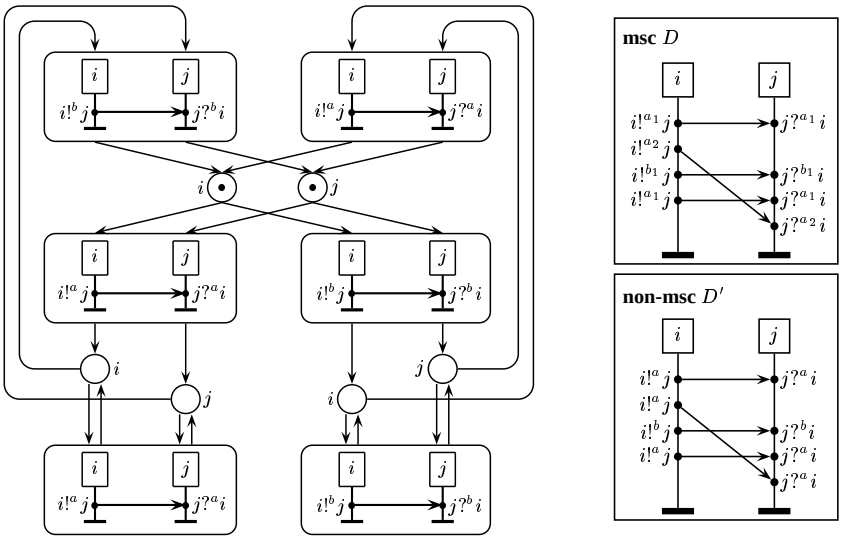

FIG. 9. Netchart $\mathcal{N}_2$ and a degenerate behavior D' ($D' \notin \mathcal{L}(\mathcal{N}_2)$)

6 Getting rid of the FIFO restriction

In this section we introduce an extended semantics for netcharts which includes non-FIFO MSCs. We show that most results in the FIFO semantics remain valid with this new approach. However we exhibit a netchart that is not implementable (Ex. 6.5).

6.1 Non-FIFO behaviors of netcharts

Let $\mathcal{N}$ be a netchart and $\mathcal{P}_\mathcal{N}$ be its low-level Petri net. The non-FIFO language $\mathcal{L}(\mathcal{P}_\mathcal{N})$ of $\mathcal{P}_\mathcal{N}$ consists of the (possibly non-FIFO) basic MSCs M such that each linear extension from $\mathrm{LE}(M)$ is an execution sequence of $\mathcal{P}_\mathcal{N}$. In particular, $\mathcal{L}_{\mathrm{fifo}}(\mathcal{P}_\mathcal{N})$ consists of all FIFO basic MSCs of $\mathcal{L}(\mathcal{P}_\mathcal{N})$. When dealing with non-FIFO basic MSCs and labellings, one has to take care of degenerating MSCs.

DEFINITION 6.1. *Let Λ_1 and Λ_2 be two sets of messages and $\lambda : \Lambda_1 \to \Lambda_2$ be a mapping from Λ_1 to Λ_2. A basic MSC $M = (E, \preccurlyeq, \xi)$ over Λ_1 is called* degenerating with λ *if the dag $\lambda(M) = (E, \prec, \lambda \circ \xi)$ is not the MSC dag of some basic MSC.*

EXAMPLE 6.2. Consider the drawings of Fig. 9. The directed acyclic graph D' is obtained from the MSC dag D with the labelling π° such that $a_1, a_2 \mapsto a$ and $b_1 \mapsto b$. Since D' is not an MSC dag, the basic MSC D is degenerating with π°.

Since we do not want to deal with degenerate behaviors in this paper, we have to select from the basic MSCs of the low-level Petri net only those basic MSCs that are not degenerating with the labelling π°.

DEFINITION 6.3. *The non-FIFO semantics $\mathcal{L}(\mathcal{N})$ of a netchart $\mathcal{N}$ consists of the basic MSCs obtained from the basic MSCs of $\mathcal{L}(\mathcal{P}_\mathcal{N})$ that are not degenerating with π°:*

$$\mathcal{L}(\mathcal{N}) = \{\pi^\circ(M) \mid M \in \mathcal{L}(\mathcal{P}_\mathcal{N}) \wedge M \text{ is not degenerating with } \pi^\circ\}.$$

EXAMPLE 6.4. Consider the netchart $\mathcal{N}_2$ of Fig. 9 for which a marking $\mathfrak{m}$ is final if $\sum_{\mathrm{Ins}(p)=i} \mathfrak{m}(p) = 1$ for each instance $i \in \mathcal{I}$. As explained in Example 6.2 the basic MSC $D \in \mathcal{L}(\mathcal{P}_{\mathcal{N}_2})$ is degenerating with π°.

6.2 Non-FIFO semantics of MPAs

A rather natural non-FIFO semantics for MPAs and a corresponding notion of implementation may be defined as follows. First, the non-FIFO semantics $\mathcal{L}(\mathcal{S})$ of an MPA $\mathcal{S}$ consists of the (possibly non-FIFO) basic MSCs M such that each linear extension of M is an execution sequence of $\mathcal{S}$. Now, an MSC language $\mathcal{L}$ is *implementable under the non-FIFO semantics of MPAs* if there are some MPA $\mathcal{S}$ over some set of messages Λ' and some labelling $\lambda : \Lambda' \to \Lambda$ such that no MSC from $\mathcal{L}(\mathcal{S})$ is degenerating with λ and $\mathcal{L} = \lambda(\mathcal{L}(\mathcal{S}))$. Differently from the FIFO semantics, *there are netcharts that are not implementable under the non-FIFO semantics*.

EXAMPLE 6.5. Continuing Example 6.4, the low-level Petri net of the netchart $\mathcal{N}_2$ depicted in Fig. 9 admits some non-FIFO executions. However all these basic MSCs are degenerating with π°: Therefore the non-FIFO semantics of $\mathcal{N}_2$ consists actually of FIFO basic MSCs only. More precisely, $\mathcal{L}(\mathcal{N}_2) = \mathcal{L}_{\mathrm{fifo}}(\mathcal{N}_2)$ is described by the HMSC $(A + B)^\star$ of Example 4.2. It is easy to show that *this MSC language is not implementable under the non-FIFO semantics of MPAs*.

6.3 Extending some results from the FIFO to the non-FIFO semantics

Theorems 3.7, 4.3, 4.6 and 4.7 can be established with the non-FIFO semantics by adapting the proofs slightly. Yet Corollaries 3.8 and 4.4 need to be more careful.

THEOREM 6.6. *It is undecidable whether some netchart language $\mathcal{L}(\mathcal{N})$ is regular (resp. can be described by some HMSC).*

Proof. The proof is based on the following key technical result: For any MPA $\mathcal{S}$ over Λ° and any mapping $\lambda : \Lambda^\circ \to \Lambda$ we can effectively build a netchart $\mathcal{N}$ such that $\mathcal{L}(\mathcal{N}) = \lambda(\mathcal{L}_\lambda)$ where $\mathcal{L}_\lambda$ be the set of basic MSCs $M \in \mathcal{L}(\mathcal{S})$ that are not degenerating with λ. Now we apply again [19, Prop. 7]. Let $\mathcal{S}$ be some MPA over Λ. We consider $\Lambda_\# = \{\#\}$ and $\lambda : \Lambda \to \{\#\}$. By the above construction, we can build some netchart $\mathcal{N}$ such that $\mathcal{L}(\mathcal{N}) = \lambda(\mathcal{L}_{\mathrm{fifo}}(\mathcal{S}))$ because $\mathcal{L}_\lambda = \mathcal{L}_{\mathrm{fifo}}(\mathcal{S})$. Then $\mathcal{L}(\mathcal{N})$ is finitely generated (resp. regular) iff $\mathcal{L}_{\mathrm{fifo}}(\mathcal{S})$ is also finitely generated (resp. regular). ■

Discussion. These undecidability results rely essentially on the possible presence of degenerating MSCs in the low-level Petri net. Similarly to results obtained for FIFO netcharts (Th. 5.2 and 5.3), we can check effectively whether a netchart admits some degenerating MSCs in its low-level Petri net. Moreover, in case no such MSC appears, then $\mathcal{L}(\mathcal{N})$ is easily implementable under the non-FIFO semantics of MPAs and we can effectively check whether it is regular. Thus, it is quite useful to avoid degenerate behaviors. For this reason, we suggest that component MSCs should use disjoint set of messages (that is, messages should be private to transitions) because this simple requirement ensures that no degenerating MSC appears in the low-level Petri net.

Acknowledgements Thanks to the anonymous referees for suggestions to improve the presentation of the paper. We thank also H. Wimmel for communicating us paper [21].

References

1. Alur R., Etessami K. and Yannakakis M.: *Realizability and verification of MSC graphs*. ICALP, LNCS **2076** (2001) 797–808
2. Alur R. and Yannakakis M.: *Model Checking of Message Sequence Charts*. CONCUR, LNCS **1664** (1999) 114–129
3. Baudru N. and Morin R.: *Safe Implementability of Regular Message Sequence Charts Specifications*. Proc. of the ACIS 4th Int. Conf. SNDP (2003) 210–217
4. Bollig B., Leucker M. and Noll Th.: *Generalised Regular MSC Languages*. FoSSaCS, LNCS **2303** (2002) 52–66
5. Caillaud B., Darondeau Ph., Hélouët L. and Lesventes G.: *HMSCs as partial specifications... with PNs as completions*. LNCS **2067** (2001) 87–103
6. Diekert V. and Rozenberg G.: *The Book of Traces*. (World Scientific, 1995)
7. Gunter E.L., Muscholl A. and Peled D.: *Compositional Message Sequence Charts*. TACAS, LNCS **2031** (2001) 496–511
8. Henriksen J.G., Mukund M., Narayan Kumar K. and Thiagarajan P.S.: *On message sequence graphs and finitely generated regular MSC language*. ICALP, LNCS **1853** (2000) 675–686
9. Henriksen J.G., Mukund M., Narayan Kumar K. and Thiagarajan P.S.: *Regular collections of message sequence charts*. MFCS, LNCS **1893** (2000) 405–414
10. Holzmann G.J.: *Early Fault Detection*. TACAS, LNCS **1055** (1996) 1–13
11. ITU-TS: *Recommendation Z.120: Message Sequence Charts*. (Geneva, 1996)
12. Lambert J.L.: *A structure to decide reachability in Petri nets*. Theoretical Comp. Science **99** (1992) 79–104
13. Lamport L.: *Time, Clocks and the Ordering of Events in a Distributed System*. Communications of the ACM **21,7** (1978) 558–565
14. Lohrey M.: *Realizability of High-level Message Sequence Charts: closing the gaps*. Theoretical Comp. Science **309** (2003) 529–554
15. Mayr E.W.: *An algorithm for the general Petri net reachability problem*. SIAM Journal of Computing **13:3** (1984) 441–460
16. Morin R.: *Recognizable Sets of Message Sequence Charts*. STACS 2002, LNCS **2285** (2002) 523–534
17. Mukund M., Narayan Kumar K. and Thiagarajan P.S: *Netcharts: Bridging the Gap between HMSCs and Executable Specifications*. CONCUR 2003, LNCS **2761** (2003) 296–310
18. Muscholl A. and Peled D.: *Message sequence graphs and decision problems on Mazurkiewicz traces*. MFCS, LNCS **1672** (1999) 81–91
19. Muscholl A. and Peled D.: *From Finite State Communication Protocols to High-Level Message Sequence Charts*. ICALP, LNCS **2076** (2001) 720–731
20. Pratt V.: *Modelling concurrency with partial orders*. International Journal of Parallel Programming **15** (1986) 33–71
21. Wimmel H.: *Infinity of Intermediate States is Decidable for Petri Nets*. Applications and Theory of Petri Nets, LNCS (2004) –To appear