The Synthesis Problem of Netcharts*

Nicolas Baudru and Rémi Morin

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

Abstract. A netchart is basically a Petri net whose places are located at some process and whose transitions are labeled by message sequence charts (MSCs). Two recent papers showed independently that any globally-cooperative high-level MSC corresponds to the behaviors of some communicating finite-state machine — or equivalently a netchart. These difficult results rely either on Thomas' graph acceptors or Zielonka's construction of asynchronous automata. In this paper we give a direct and self-contained synthesis of netcharts from globally-cooperative high-level MSCs by means of a simpler unfolding procedure.

1 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 [12]) 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. Yet this formalism can be used at an early stage of design to detect errors in the specification [11]. 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, 2, 3, 5, 6, 8, 9, 10, 15, 16, 17, 18].

Collections of MSCs are often specified by means of high-level MSCs (HMSCs). The latter can be seen as directed graphs labeled by component MSCs. However such specifications may be unrealistic because this formalism allows to specify sets of MSCs that correspond to no communicating finite-state machine. Furthermore *it is undecidable whether a HMSC describes an implementable language*. 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 is a Petri net whose places are labeled by processes and whose transitions are labeled by MSCs. This new approach benefits from a graphical description, a formal semantics, and an appropriate expressive power: As opposed to HMSCs, *netcharts describe precisely all implementable languages* and it is actually easy to derive an equivalent communicating finite-state machine from a netchart. It follows that it is undecidable whether a HMSC is equivalent to some netchart.

Many model-checking problems are undecidable with general HMSCs. For this reason subclasses of HMSCs have been investigated in the literature, in particular *globally-cooperative* HMSCs [8]. Logical and algebraic characterizations of these HMSCs were

* Supported by the ANR project SOAPDC.

S. Donatelli and P.S. Thiagarajan (Eds.): ICATPN 2006, LNCS 4024, pp. 84–104, 2006.

established in [16] and various related verification techniques are now available [9]. Recently two papers showed that globally-cooperative HMSCs describe implementable languages [5, 9]. These works extend a seminal result by Henriksen et al. who showed that all regular sets of MSCs are implementable [10]. In [5] Bollig and Leucker apply the theory of graph acceptors [21] to prove that any set of MSCs definable in existential MSO logic is implementable. In [9] Genest et al. apply Zielonka's theorem [22] to prove that any existentially-bounded recognizable set of compositional MSCs is implementable. Both studies are rather difficult and quite technical. In the particular case of finitely generated and recognizable sets of MSCs [16], both results imply that any globally-cooperative HMSC describes an implementable set of MSCs, i.e. it corresponds to some netchart. *The aim of this paper is to present a direct, self-contained, and simpler implementation technique to transform a globally-cooperative HMSC into an equivalent netchart.* The translation from netcharts into communicating finite-state machines is rather simple to define but quite tedious to handle in detailed proofs. We adopt in this paper the formalism of netcharts in order to simplify the presentation of our construction. Besides netcharts were at the origine of our first intuitions.

The paper is organized as follows. In Section 1 we recall the basic definitions of MSCs, Petri nets, and netcharts. Next Section 2 presents the semantics of a netchart as the set of MSCs that correspond to the behaviors of some underlying Petri net. Section 3 introduces the notion of HMSC regarded as an automaton labeled by MSCs. We define there a simple but naive transformation of HMSCs into netcharts. In some cases this transformation leads to a netchart whose behaviors differ from those of the given HMSC. Our strategy is motivated by an example that shows that it is sufficient to *unfold* the given HMSC in order to ensure that the naive transformation into netcharts preserves the semantics. Section 4 presents in details our unfolding algorithm of globally-cooperative HMSCs together with some simple but crucial properties of the resulting structure. Finally Section 5 explains why the naive transformation preserves the behaviors when it is applied to the unfolding of any globally-cooperative HMSC.

Our unfolding algorithm proceeds inductively on the number of communication types involved in the given HMSC by defining a family of globally-cooperative HMSCs called triangles and boxes. A triangle corresponds intuitively to a partial unfolding that represents only part of the behaviors starting from a given node of the HMSC. The role of boxes is to complete triangles by connecting copies of triangles with missing edges.

Admittedly this unfolding resembles an algorithm designed recently in [4] in the framework of Mazurkiewicz traces [7] to build asynchronous automata of polynomial size in terms of the number of states from asynchronous systems. However it is often quite difficult to transfer results or techniques from Mazurkiewicz trace theory to the framework of MSCs (see e.g. [2, 9, 10]) because communication no longer means synchronisation. The unfolding procedure presented here differs from the one used in [4] in several aspects: The induction proceeds over communication types, not component basic MSCs; the termination of the construction of boxes relies essentially on the hypothesis that loops of globally-cooperative HMSCs have a connected communication graph whereas [4] unfolds asynchronous systems with possible unconnected loops and termination is there obvious; last but not least, the present unfolding algorithm is *exponential* in the number of nodes of the given HMSC.

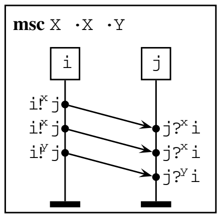

Fig. 1. FIFO MSC

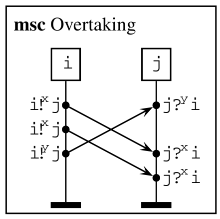

Fig. 2. Non-FIFO MSC

2 Background

Message sequence charts (MSCs) are defined by several recommendations that indicate how one should represent them graphically [12]. Examples of MSCs are given in Figures 1 and 2 in which time flows top-down. In this paper we regard MSCs as particular labeled partial orders (or pomsets) following a traditional trend of modeling concurrent executions [7, 13, 20].

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 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.

2.1 Basic Message Sequence Charts

We present here a formal definition of basic 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 $e \mathrel{-\!\!\!<} f$ 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 2.1. *A basic* message sequence chart (MSC) *over the set of messages* Λ *is a pomset* $M = (E, \preccurlyeq, \xi)$ *over* $\Sigma_{\mathcal{I}}^{\Lambda}$ *that fulfills the four following conditions:*

M_1*:* $\forall e, f \in E$*:* $\mathrm{Ins}(e) = \mathrm{Ins}(f) \Rightarrow (e \preccurlyeq f \vee f \preccurlyeq e)$
M_2*:* $\forall e, f \in E$*:* $e \rightsquigarrow f \Rightarrow e \preccurlyeq f$
M_3*:* $\forall e, f \in E$*:* $[e \prec\!\!\cdot f \wedge \mathrm{Ins}(e) \neq \mathrm{Ins}(f)] \Rightarrow e \rightsquigarrow f$
M_4*:* $\forall i, j \in \mathcal{I}, \forall x \in \Lambda, |M|_{i!^x j} = |M|_{j?^x i}$.

By M_1, events occurring on the same instance are linearly ordered: Hence non-deterministic choice cannot be described within an MSC. Property M_2 formalizes simply that the reception of any message will occur after the corresponding send event. By M_3, causality in M consists only in the linear dependency over each instance and the ordering of pairs of corresponding send and receive events. Finally, Condition M_4 requires each send event matches some receive event: The matching relation $\rightsquigarrow$ builds a one-to-one correspondence between send events and receive events. We let $\mathrm{b}\mathbb{MSC}$ denote the set of all basic MSCs. 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 a basic 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 is called *FIFO* 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 scenarios that use several channels (or message types) between pairs of processes (Fig. 2).

For convenience we shall use at some point the notion of MSC *with* ϵ*-actions*. For each instance $i \in \mathcal{I}$ we define a new symbol ϵ_i and we put $\mathrm{Ins}(\epsilon_i) = i$. Then *a basic MSC with* ϵ*-actions* is simply a pomset over the extended alphabet $\Sigma_{\mathcal{I}}^{\Lambda} \cup \{\epsilon_i \mid i \in \mathcal{I}\}$ which satisfies the conditions M_1 to M_4.

2.2 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 $\mathfrak{m} \in \mathbb{N}^P$. A transition t is *enabled* at $\mathfrak{m} \in \mathbb{N}^P$ 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* from $\mathfrak{m}$ to $\mathfrak{m}'$ is a word $u = t_1 ... t_n \in T^\star$ such that there are markings $\mathfrak{m}_0$,..., $\mathfrak{m}_n$ satisfying $\mathfrak{m}_0 = \mathfrak{m}$, $\mathfrak{m}_n = \mathfrak{m}'$, and $\mathfrak{m}_{k-1}\,[t_k\rangle\,\mathfrak{m}_k$ for all naturals $k \in [1, n]$. Then the sequence $s = \mathfrak{m}_0\,[t_1\rangle\,\mathfrak{m}_1 ... \mathfrak{m}_{n-1}\,[t_n\rangle\,\mathfrak{m}_n$ is called the *firing sequence* of u from $\mathfrak{m}$ to $\mathfrak{m}'$ and is denoted by $s = \mathfrak{m}\,[u\rangle\,\mathfrak{m}'$. If $\mathfrak{m} = \mathfrak{m}_{\mathrm{in}}$ and $\mathfrak{m}' \in \mathfrak{F}$ then the execution sequence u is called *complete*. The language $L(\mathcal{P})$ consists of all complete execution sequences of $\mathcal{P}$.

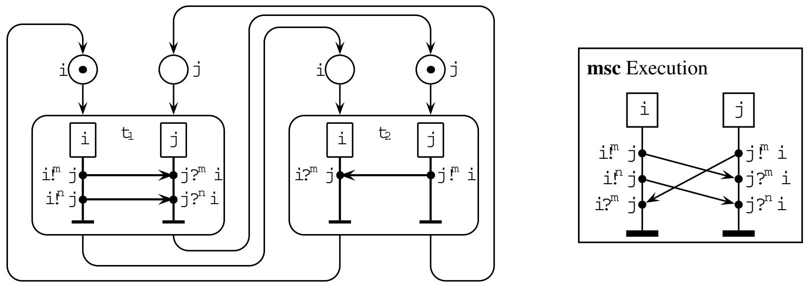

Fig. 3. A netchart $\mathcal{N}$ and a corresponding MSC

2.3 Netcharts

{A netchart is basically a Petri net whose places are labeled by instances and whose transitions are labeled by FIFO basic MSCs. Similarly to Petri nets, netcharts admit an intuitive visual representation: Examples of netcharts are given in Fig. 3, 7, 9 and 11.

DEFINITION 2.2. *A* netchart *over Λ consists of a Petri net $(P, T, F, \mathfrak{m}_{\text{in}}, \mathfrak{F})$ and two mappings* $\text{Ins} : P \to \mathcal{I}$ *and* $\mathcal{M} : T \to \text{b}\mathbb{MSC}$ *such that* Ins *associates each place $p \in P$ with some instance* $\text{Ins}(p)$ *and $\mathcal{M}$ associates each transition $t \in T$ with a FIFO basic MSC $\mathcal{M}(t)$ over the set of messages Λ. Three conditions are required for such a structure to be a netchart:*

N_1*: For each instance $i \in \mathcal{I}$, there is a single token within places located on instance i, i.e. $\sum_{\text{Ins}(p)=i} \mathfrak{m}_{\text{in}}(p) = 1$.*
N_2*: For each transition $t \in T$ and each instance $i \in \mathcal{I}$ there is at most one place $p \in t^\bullet$ such that $\text{Ins}(p) = i$.*
N_3*: For each transition $t \in T$ and each instance $i \in \mathcal{I}$ there is at most one place $p \in {}^\bullet t$ such that $\text{Ins}(p) = i$.*
A netchart is called prime *if for all $t \in T$ we have $\text{Ins}({}^\bullet t) = \text{Ins}(t^\bullet) = \text{Ins}(\mathcal{M}(t))$.*

By N_1 the initial marking of a netchart is safe; furthermore each instance is associated with a unique initial place. Intuitively this observation extends to the semantics of netcharts: In each reachable marking a token denotes the current local state of each instance. Axiom N_2 stipulates that an instance occurs at most once in the postcondition of any transition. This condition ensures that the local state of each instance corresponds to a single token. Axiom N_3 requires that at most one place located on instance i is a precondition of a given transition. The semantics detailed below will show that transitions that do not satisfy this requirement cannot take part entirely in the behaviors of the netchart: We could remove N_3 without affecting the expressive power of netcharts.

Prime netcharts are those introduced in [17]. This additional requirement ensures in particular that ${}^\bullet t \cup t^\bullet$ is empty as soon as $\mathcal{M}(t)$ is the empty MSC. In the next section we make use of basic MSCs with ϵ-actions to extend the semantics of prime netcharts studied in [17, 3] to the relaxed setting adopted here. Noteworthy any netchart can easily be transformed into an equivalent prime one: Consequently the expressive power of these extended netcharts is the same as the prime ones. This remark simplifies the presention of our result and allows us to apply to the present setting some of the results from [3].

3 Semantics of Netcharts

In this section we fix a netchart $\mathcal{N} = ((P, T, F, \mathfrak{m}_{\text{in}}, \mathfrak{F}), \text{Ins}, \mathcal{M})$ over the set of messages Λ and define formally its behaviors. The semantics of $\mathcal{N}$ consists of FIFO basic MSCs over Λ (Fig. 3). The latter are derived from the FIFO basic MSCs that correspond to the complete execution sequences of some low-level Petri net $\mathcal{P}_\mathcal{N}$ (Fig. 5). Actually, the execution sequences of $\mathcal{P}_\mathcal{N}$ use a *refined set of messages* Λ° and the behaviors of $\mathcal{N}$ are obtained by projection of messages from Λ° onto Λ.

3.1 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 FIFO basic 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. 4.

This construction needs to regard each basic MSC (with ϵ-actions) $M = (E, \preccurlyeq, \xi)$ 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 $e {-\!\!\prec_i} f$ if e occurs immediately before f on instance i. Then the binary relation $\prec$ consists of all pairs of matching events together with all pairs of covering events w.r.t. $\preccurlyeq_i$.

DEFINITION 3.1. *The* MSC dag *of a basic MSC $M = (E, \preccurlyeq, \xi)$ with possibly ϵ-actions is a labeled directed acyclic graph $(E, \prec, \xi)$ such that we have $e \prec f$ if $e \rightsquigarrow f$ or $e {-\!\!\prec_i} f$ for some instance $i \in \mathcal{I}$.*

Clearly we can recover the basic MSC from its MSC dag. The reason for this is that ${-\!\!\prec} \subseteq {\prec}$ 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 in the sequel of this paper.

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 we add to the basic MSC $\mathcal{M}(t)$ an event labeled ϵ_i on instance i if the instance i is not active in $\mathcal{M}(t)$ while there exists a place $p \in {}^\bullet t$ such that $\text{Ins}(p) = i$. Note that these new events are isolated because no other event occurs on this instance.

Now the places P_t are identified with pairs from $\prec$. In particular places do not depend on possibly added events labeled ϵ_i. On the other hand the transitions T_t are

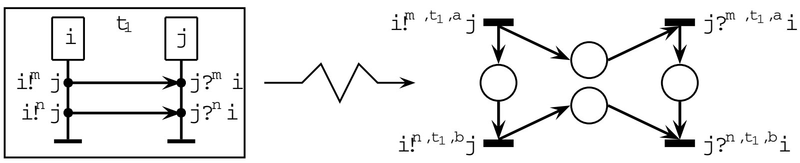

Fig. 4. From transition t_1 to Petri net $\mathcal{P}_{t_1}$

identified with some send or receive actions over the new set of messages $\Lambda^\circ = \Lambda \times T \times P_t$ or with added event labeled by ϵ_i. Formally, we put $P_t = \prec$ and

$$T_t = \{i!^{m,t,(e,f)}j, j?^{m,t,(e,f)}i \mid (e,f) \in \prec \wedge \xi(e) = i!^m j \wedge \xi(f) = j?^m i\}$$
$$\cup\{\ (\epsilon_i, t) \mid i \notin \mathrm{Ins}(\mathcal{M}(t)) \wedge \exists p \in {}^\bullet t, \mathrm{Ins}(p) = i\}.$$

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 $\rho(i!^{m,t,(e,f)}j) = e$, $\rho(j?^{m,t,(e,f)}i) = f$ and $\rho(\epsilon_i, t) = \epsilon_i$. 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

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

In the next subsection 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:

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

3.2 Low-Level Petri Net and Its FIFO Behaviors

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. 5.

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 $\mathcal{P}_t$: $T_\mathcal{N} = \bigcup_{t \in T} T_t$. For latter purposes we also define the map Comp that associates each transition a from $T_\mathcal{N}$ with the transition $t \in T$ such that $a \in T_t$. Thus $\mathrm{Comp}(i!^{m,t,p}j) = t$, $\mathrm{Comp}(i?^{m,t,p}j) = t$, and $\mathrm{Comp}(\epsilon_i, t) = t$. 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: $\mathfrak{F}_\mathcal{N} = \{\mathfrak{m} \in \mathbb{N}^P \mid \mathfrak{m}_{|P} \in \mathfrak{F} \wedge \mathfrak{m}_{|P_\mathcal{N} \setminus P} = 0\}$.

Any complete 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 $P_\mathcal{N} \setminus P$ are empty.

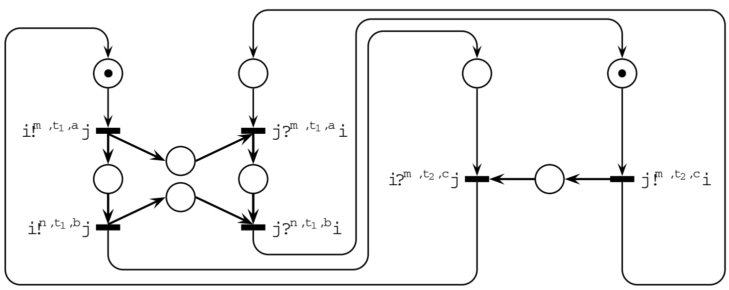

Fig. 5. The low-level Petri net $\mathcal{P}_\mathcal{N}$ associated to the netchart $\mathcal{N}$ of Fig. 3

Moreover u is actually a linear extension of a unique basic MSC over the set of extended messages Λ° that consists of triples (m, t, p).

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

Interestingly, 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 complete execution sequences of $\mathcal{P}_\mathcal{N}$. Noteworthy it can happen that a complete 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 in this paper.

3.3 Set of MSCs Associated to Some Netchart

Recall now that MSCs from $\mathcal{L}_{\text{fifo}}(\mathcal{P}_\mathcal{N})$ may contain some events labeled by ϵ_i and use a refined set of messages Λ° that consists of triples (m, t, p) where $m \in \Lambda$, $t \in T$, and $p \in P_t$. We let $\pi^\circ : \Lambda^\circ \to \Lambda$ denote the labelling that associates each triple $(m, t, p) \in \Lambda^\circ$ with the message $m \in \Lambda$. This labelling extends to a function that maps actions from $\Sigma_\mathcal{I}^{\Lambda^\circ}$ 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° onto the FIFO basic MSCs over Λ. Since we deal here with MSCs with possibly ϵ-actions, we ask in this paper that π° removes all actions ϵ_i, too. The semantics of the netchart $\mathcal{N}$ is defined now from the semantics of its low-level Petri net $\mathcal{P}_\mathcal{N}$ by means of the projection π°.

DEFINITION 3.3. *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 projection π°: $\mathcal{L}_{\text{fifo}}(\mathcal{N}) = \pi^\circ(\mathcal{L}_{\text{fifo}}(\mathcal{P}_\mathcal{N}))$.*

EXAMPLE 3.4. 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. The MSC M on the right-hand side of this figure illustrates a complete execution sequence of the low-level Petri net of $\mathcal{N}$ that does not correspond to some FIFO basic MSC.

The main property of *prime* netcharts from [17] is that their MSC language can be implemented in polynomial time as the behaviors of some communicating finite-state machine. Clearly this observation extends easily to the netcharts adopted in this paper.

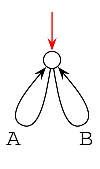

Fig. 6. $\mathcal{G}_1$

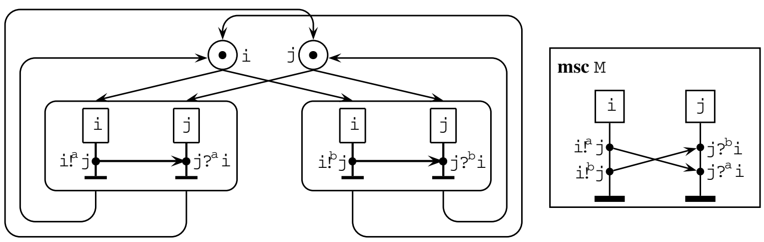

Fig. 7. Netchart $\mathcal{N}_1$ and some non-FIFO behavior $M \notin \mathcal{L}_{\text{fifo}}(\mathcal{N})$

4 Netcharts vs. High-Level Message Sequence Charts

In this section we recall the equivalent notions of high-level MSCs (HMSCs) and MSC-graphs (MSGs). We recall also some decidability results about the respective expressive power of MSGs and netcharts. By means of three examples we introduce a naive translation of MSGs into netcharts and motivate the seek for an unfolding procedure to ensure a correct implementation of globally-cooperative MSGs as netcharts.

4.1 HMSCs and MSGs

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 to compose specifications in order to describe infinite sets of basic MSCs: We obtain high-level message sequence charts (HMSCs) as rational expressions or equivalently automata labeled by basic MSCs.

DEFINITION 4.1. *An* MSC-graph (MSG) *is a structure $\mathcal{G} = (Q, \imath, \Sigma, \longrightarrow, Q_f)$ where Q is a finite set of nodes with some initial node $\imath$ and some final nodes $Q_f \subseteq Q$, Σ is a finite subset of basic MSCs, and $\longrightarrow \subseteq Q \times \Sigma \times Q$ is a set of labeled edges.*

The semantics of MSGs is quite natural. The language associated with an MSG consists of all basic MSCs that are the product of MSCs appearing along a path from the initial node to some final node. By Kleene's theorem, a set of basic MSCs corresponds to some MSG iff it is rational, i.e. it can be built from finite sets by means of union, product, and iteration.

EXAMPLE 4.2. Let A and B be the two components MSCs of the netchart $\mathcal{N}_1$ depicted in Fig. 7. The language $\mathcal{L}_{\text{fifo}}(\mathcal{N}_1)$ corresponds to the HMSC $(A + B)^\star$ and to the MSG of Fig. 6.

We showed in [3] that it is undecidable whether the language $\mathcal{L}_{\text{fifo}}(\mathcal{N})$ of a given netchart is rational, that is, can be described by some MSG [3, Cor. 4.4]. We showed also that it is undecidable whether the language of some given MSG can be described by some netchart [3, Th. 4.7].

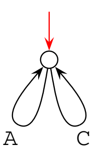

Fig. 8. $\mathcal{G}_2$

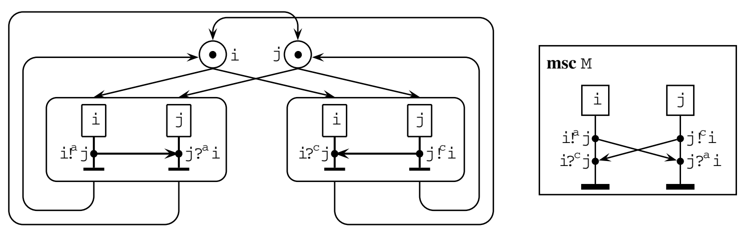

Fig. 9. Wrong implementation of $(A + C)^\star$

4.2 Globally-Cooperative MSG

Most model-checking issues related to MSGs are undecidable in general. For this reason subclasses of MSGs have been introduced in the past years. We are here interested in globally-cooperative MSGs from [8]. These MSGs correspond precisely to the $\star$-connected HMSCs from [16] and extend the class of bounded or locally-synchronized MSGs from [1, 18] by removing the requirement that the set of MSCs described by these MSGs be channel-bounded [15]. These restrictions are motivated by a similar approach in Mazurkiewicz trace theory [14, 19].

We need first to introduce the following notion. The *communication graph* $\mathrm{CG}(M)$ of a basic MSC $M = (E, \preccurlyeq, \xi)$ is the directed graph $(\mathcal{I}, \mapsto)$ such that $(i, j) \in \mapsto$ if there is an event $e \in E$ such that $\xi(e) = i!^x j$ for some $x \in \Lambda$. An instance $i \in \mathcal{I}$ is called *active* if either $i \mapsto j$ or $j \mapsto i$ for some j. In this paper a directed graph $(\mathcal{I}, \mapsto)$ is called *connected* if the symmetric closure of its restriction to active instances is connected.

DEFINITION 4.3. *An MSG is* globally-cooperative *(for short, a gc-MSG) if for all loops* $q_0 \xrightarrow{M_1} q_1 \xrightarrow{M_2} \ldots \xrightarrow{M_n} q_n = q_0$ *the product basic MSC* $M_1 \cdot M_2 \cdot \ldots \cdot M_n$ *has a connected communication graph.*

Algebraic and logical characterizations of the languages described by gc-MSGs were established in [16]. More recently two articles showed independently that these languages are implementable by communicating finite-state machines provided that one restricts to FIFO MSCs [5, 9]. On the other hand we have showed in [3, Th. 3.7] that all implementable sets of MSCs can be described by netcharts. As a consequence, *the language of any gc-MSG can be described by some netchart*. Note here that [5] relies on Thomas' graph acceptors [21] whereas [9] is based on the construction of asynchronous cellular automata [22]. Both approaches are quite involved and have high complexity costs. We give in this paper a direct, self-contained, and simpler construction that transforms any given gc-MSG into an equivalent netchart.

4.3 Naive Implementation Technique

Our method uses a translation of MSGs into netcharts illustrated by Figures 6 to 11.

DEFINITION 4.4. *Let* $\mathcal{G} = (Q, \imath, \Sigma, \longrightarrow, Q_f)$ *be an MSG. The corresponding netchart* $\widehat{\mathcal{G}}$ *is the structure* $\widehat{\mathcal{G}} = (P, T, F, \mathfrak{m}_{\mathrm{in}}, \mathfrak{F}, \mathrm{Ins}, \mathcal{M})$ *where*

- $P = Q \times \mathcal{I}$ *with* $\mathrm{Ins}(q, k) = k$,
- $T = \longrightarrow \subseteq Q \times \Sigma \times Q$ *with* $\mathcal{M}(q_1 \xrightarrow{M} q_2) = M$,
- *for all edges* $t = (q_1 \xrightarrow{M} q_2)$ *from* T *and all places* $(q, k) \in P$ *we have* $(q, k) \in {}^\bullet t \Leftrightarrow q = q_1$ *and* $(q, k) \in t^\bullet \Leftrightarrow q = q_2$,
- $\mathfrak{m}_{\mathrm{in}} = \{(\imath, k) \mid k \in \mathcal{I}\}$ *and a multiset of places* $\mathfrak{m} \in \mathbb{N}^P$ *is final if there exists a final node* $q_f \in Q_f$ *such that for each* $(q, k) \in Q$ *we have* $\mathfrak{m}(q, k) = 1$ *if* $q = q_f$ *and* $\mathfrak{m}(q, k) = 0$ *otherwise.*

EXAMPLE 4.5. Consider first again the netchart $\mathcal{N}_1$ of Fig. 7 and its two component MSCs A and B. Clearly the MSG $\mathcal{G}_1$ depicted on Fig. 6 accepts $(A + B)^\star$. It is easy to check that $\mathcal{N}_1 = \widehat{\mathcal{G}_1}$ with $\mathcal{I} = \{i, j\}$. Note here that $\widehat{\mathcal{G}_1}$ is a correct implementation of $\mathcal{G}_1$ since $\widehat{\mathcal{G}_1}$ and $\mathcal{G}_1$ both accept $(A + B)^\star$.

EXAMPLE 4.6. Consider now the netchart $\mathcal{N}_2$ of Figure 9 with its two component MSCs A and C. Then $\mathcal{N}_2$ is exactly the netchart $\widehat{\mathcal{G}_2}$ associated with the MSG $\mathcal{G}_2$ of Fig. 8. Observe here that $\mathcal{G}_2$ accepts $(A + C)^\star$ whereas $\widehat{\mathcal{G}_2}$ accepts some MSC $M \notin (A + C)^\star$ depicted on the right-hand side of Figure 9.

This example shows that the direct construction of the netchart $\widehat{\mathcal{G}}$ from some MSG $\mathcal{G}$ may fail to produce a correct implementation of $\mathcal{G}$. This is no surprise since we know that there are MSGs whose languages are not implementable and it is even undecidable to check implementability of MSGs. That is why we shall restrict to globally-cooperative MSGs in the next section.

Although $\mathcal{G}_2$ and $\widehat{\mathcal{G}_2}$ from Example 4.6 accept distinct languages we have in general the following useful inclusion relation.

PROPOSITION 4.7. *For any MSG $\mathcal{G}$ we have $L(\mathcal{G}) \subseteq L(\widehat{\mathcal{G}})$.*

For each node $q \in Q$ we let $\mathfrak{m}_q$ denote the marking of the low-level Petri net of the netchart $\widehat{\mathcal{G}}$ such that $\mathfrak{m}_q(p) = 1$ if $p = (q, k)$ for some instance k — that is, p is a place from the netchart $\widehat{\mathcal{G}}$ that corresponds to the node q — and $\mathfrak{m}_q(p) = 0$ otherwise. We say that a firing sequence $s = \mathfrak{m}\,[u\rangle\,\mathfrak{m}'$ in the low-level Petri net of $\widehat{\mathcal{G}}$ is *arched* if there are two nodes q and q' in $\mathcal{G}$ such that $\mathfrak{m} = \mathfrak{m}_q$ and $\mathfrak{m}' = \mathfrak{m}_{q'}$. Noteworthy each complete execution sequence that leads the low-level Petri net of the netchart $\widehat{\mathcal{G}}$ from its initial marking to some final marking corresponds to an arched firing sequence. The next observation will be used to prove our main technical lemma.

REMARK 4.8. Let $\mathcal{G}$ be an MSG and $\mathfrak{m}_q\,[u\rangle\,\mathfrak{m}_{q'}$ be an arched firing sequence of the low-level Petri net of $\widehat{\mathcal{G}}$. Then u is the linear extension of some basic MSC M_u. Recall now that each transition t of the low-level Petri net of $\widehat{\mathcal{G}}$ corresponds to a transition Comp(t) of $\widehat{\mathcal{G}}$ which is defined as an edge $q_1 \xrightarrow{M} q_2$ from $\mathcal{G}$. If an arched firing sequence $\mathfrak{m}_q\,[u\rangle\,\mathfrak{m}_{q'}$ satisfies $q \neq q'$ and there is some edge a such that all transitions t that appear in u satisfy Comp$(t) = a$ then a equals $q \xrightarrow{\pi^\circ(M_u)} q'$.

Let j be some instance and q some node of $\mathcal{G}$. The behavior of instance j within a firing sequence of the netchart $\widehat{\mathcal{G}}$ from $\mathfrak{m}_q$ may be projected to a path from q in Q. Intuitively the local state and the behavior of instance j along a firing sequence corresponds to some token moving from places to places, all located at instance j, some of them corresponding to a state of $\mathcal{G}$. The idea here is simply to collect the sequence of states of $\mathcal{G}$ visited by instance j. Formally we associate inductively each firing sequence $s = \mathfrak{m}_q\,[u\rangle\,\mathfrak{m}'$ in the low-level Petri net of $\widehat{\mathcal{G}}$ with a path $s{\downarrow}j$ in $\mathcal{G}$ called the *projection of s on instance j* as follows:

- If s is the empty firing sequence restricted to $\mathfrak{m}_q$ then $s{\downarrow}j = q$;
- If $s = s' \cdot f$ where $f = \mathfrak{m}\,[a\rangle\,\mathfrak{m}'$ then
  - $s{\downarrow}j = s'{\downarrow}j \cdot t$ if $\mathrm{Ins}(\rho(a)) = j$, Comp$(a) = t$, and $\sum_{q' \in Q} \mathfrak{m}'(q', j) = 1$;
  - and $s{\downarrow}j = s'{\downarrow}j$ otherwise.

4.4 Unfolding Strategy

We conclude this section by introducing our unfolding approach with the help of an example. Let $\mathcal{G}_1 = (Q_1, \imath_1, A, \longrightarrow_1, F_1)$ and $\mathcal{G}_2 = (Q_2, \imath_2, A, \longrightarrow_2, F_2)$ be two MSGs

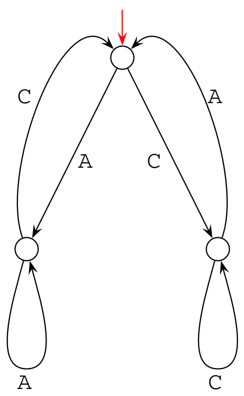

Fig. 10. MSG G_2'

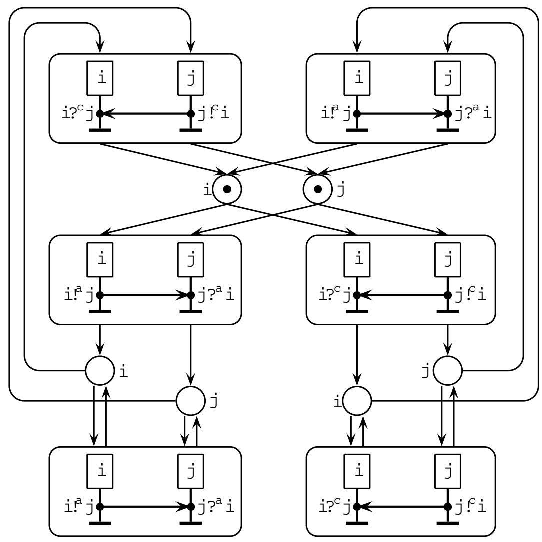

Fig. 11. Correct implementation of $(A + C)^\star$

over a subset of actions $A \subseteq \Sigma$. A *morphism* $\sigma : \mathcal{G}_1 \to \mathcal{G}_2$ *from* $\mathcal{G}_1$ *to* $\mathcal{G}_2$ is a mapping $\sigma : Q_1 \to Q_2$ from Q_1 to Q_2 such that $\sigma(\imath_1) = \imath_2$, $\sigma(F_1) \subseteq F_2$, and $q_1 \xrightarrow{a}_1 q_1'$ implies $\sigma(q_1) \xrightarrow{a}_2 \sigma(q_1')$. In particular, $L(\mathcal{G}_1) \subseteq L(\mathcal{G}_2)$. Moreover if $\mathcal{G}_2$ is globally-cooperative then $\mathcal{G}_1$ is globally-cooperative, too. A morphism $\sigma : \mathcal{G}_1 \to \mathcal{G}_2$ is called *full* if the following two requirements are satisfied: $\sigma(F_1) = F_2 \cap \sigma(Q_1)$ and for all nodes $q_1 \in Q_1$ and all actions $a \in A$, if $\sigma(q_1) \xrightarrow{a}_2 q_2'$ for some $q_2' \in Q_2$ then $q_1 \xrightarrow{a}_1 q_1'$ for some $q_1' \in Q_1$ such that $\sigma(q_1') = q_2'$. In that case we have $L(\mathcal{G}_1) = L(\mathcal{G}_2)$.

Our strategy is motivated by the following example.

EXAMPLE 4.9. We continue Example 4.6 and consider the MSG $\mathcal{G}_2'$ depicted in Figure 10. Clearly $\mathcal{G}_2'$ accepts $(A + C)^\star$ similarly to $\mathcal{G}_2$. Note here that there is an obvious full morphism from $\mathcal{G}_2'$ onto $\mathcal{G}_2$ which leads us to call informally $\mathcal{G}_2'$ an *unfolding* of $\mathcal{G}_2$. The netchart $\widehat{\mathcal{G}_2'}$ is depicted in Fig. 11. It is not difficult to check that this netchart accepts $(A + C)^\star$, too. Thus $\widehat{\mathcal{G}_2'}$ is a correct implementation of $\mathcal{G}_2$.

This example shows that in some cases it is sufficient to unfold the MSG in order to ensure that the simple translation into netcharts from Definition 4.4 yields a correct implementation. In the two next sections we show that this approach is valid for any gc-MSG.

5 Unfolding of a Globally-Cooperative MSG

In the rest of the paper we fix a *globally-cooperative* MSG $\mathcal{G} = (Q, \imath, \Sigma, \longrightarrow, F)$ where each MSC from Σ is FIFO. The aim of this section is to associate with $\mathcal{G}$ a family of MSGs called *boxes* and *triangles* which are defined inductively. The last box built by this construction will be called the *unfolding* of $\mathcal{G}$ (Def. 5.1). Boxes and triangles are associated with an initial node that may not correspond to the initial node of $\mathcal{G}$. They are associated also with a subset of MSCs $A \subseteq \Sigma$. For these reasons, for any node $q \in Q$ and any subset of actions $A \subseteq \Sigma$, we let $\mathcal{G}_{A,q}$ denote the MSG $(Q, q, A, \longrightarrow_A, F)$ where $\longrightarrow_A$ is the restriction of $\longrightarrow$ to the edges labeled by MSCs in A: $\longrightarrow_A = \longrightarrow \cap (Q \times A \times Q)$.

We shall proceed inductively on directed graphs over $\mathcal{I}$. For each directed graph $T \subseteq \mathcal{I}^2$ we let $\Sigma_T \subseteq \Sigma$ denote the subset of basic MSCs from Σ whose communication graph is included in T. For convenience we put $\mathcal{G}_{T,q} = \mathcal{G}_{\Sigma_T,q}$. We shall define the box $\Box_{T,q}$ for all nodes $q \in Q$ and all subgraphs $T \subseteq \mathcal{I}^2$. The box $\Box_{T,q}$ is a pair $(\mathcal{B}_{T,q}, \beta_{T,q})$ where $\mathcal{B}_{T,q}$ is an MSG over T and $\beta_{T,q} : \mathcal{B}_{T,q} \to \mathcal{G}_{T,q}$ is a morphism. Similarly, we shall define the triangle $\triangle_{T,q}$ for all nodes q and all *non-empty* subgraphs T. The triangle $\triangle_{T,q}$ is a pair $(\mathcal{T}_{T,q}, \tau_{T,q})$ where $\mathcal{T}_{T,q}$ is an MSG over Σ_T and $\tau_{T,q} : \mathcal{T}_{T,q} \to \mathcal{G}_{T,q}$ is a morphism. Since $\mathcal{G}$ is globally-cooperative (Def. 4.3), all boxes $\mathcal{B}_{T,q}$ and all triangles $\mathcal{T}_{T,q}$ are globally-cooperative, too.

The *height* of a box $\Box_{T,q}$ or a triangle $\triangle_{T,q}$ is the cardinality of T. Boxes and triangles are defined inductively on the height. We first define the box $\Box_{\emptyset,q}$ for all nodes $q \in Q$. Then triangles of height h are built upon boxes of height $g < h$ and boxes of height h are built upon triangles of height h. More precisely each box $\Box_{T,q}$ is made of copies of triangles $\triangle_{T,q'}$. The precise construction of $\Box_{T,q}$ will depend on the connectivity of the directed graph T. Moreover we shall make use of the hypothesis that $\mathcal{G}$ is globally-cooperative when defining the construction of the $\Box_{T,q}$ associated with a non-connected graph T.

This family of boxes and triangles will lead us to the definition of the unfolding of $\mathcal{G}$ which is simply the box $\mathcal{B}_{T,q}$ with $T = \mathcal{I}^2$ and $q = \imath$.

DEFINITION 5.1. *The* unfolding $\mathcal{G}_{\mathrm{Unf}}$ *of* $\mathcal{G} = (Q, \imath, \Sigma, \longrightarrow, F)$ *is the box* $\mathcal{B}_{\mathcal{I}^2,\imath}$.

Along the definition of boxes we will observe that each morphism $\beta_{T,q} : \mathcal{B}_{T,q} \to \mathcal{G}_{T,q}$ is full. This is precisely the main property of boxes as opposed to triangles.

The base case of the induction deals with boxes of height 0. For all nodes $q \in Q$, the box $\Box_{\emptyset,q}$ consists of the morphism $\beta_{\emptyset,q} : \{q\} \to Q$ that maps q to itself together with the MSG $\mathcal{B}_{\emptyset,q} = (\{q\}, q, \emptyset, \emptyset, F_{\emptyset,q})$ where $F_{\emptyset,q} = \{q\}$ if $q \in F$ and $F_{\emptyset,q} = \emptyset$ otherwise. More generally a node of a box or a triangle is final if it is associated with a final node of $\mathcal{G}$.

5.1 Building Triangles from Boxes

Triangles are made of boxes of lower height. Boxes are inserted into a triangle inductively along a tree-like structure and several copies of the same box may appear within a triangle. We need to keep track of this structure in order to prove properties of triangles (and boxes) inductively. This requires to distinguish between nodes inserted within different copies of different boxes or different copies of the same box. To achieve this, each node of a triangle is equipped with a *rank* $k \in \mathbb{N}$ such that all nodes with the same rank come from the same copy of the same box. For these reasons, a node of a triangle $\triangle_{T^\circ,q^\circ} = (\mathcal{T}_{T^\circ,q^\circ}, \tau_{T^\circ,q^\circ})$ is encoded as a quadruple $v = (w, T, q, k)$ such that w is a node from the box $\Box_{T,q}$ with $T \subsetneq T^\circ$; moreover v is added within the k-th box inserted into the triangle in construction. By convention the node v maps to the node $\tau_{T^\circ,q^\circ}(v) = \beta_{T,q}(w) \in Q$, i.e. the insertion of boxes preserves the correspondence to the nodes of $\mathcal{G}$. Thus the morphism τ_{T°,q° of a triangle $\triangle_{T^\circ,q^\circ}$ is encoded in the data structure of its nodes. We denote by $\mathcal{B}' = \textsc{Mark}(\mathcal{B}, T, q, k)$ the generic process that creates a copy $\mathcal{B}'$ of an MSG $\mathcal{B}$ by replacing each node w of $\mathcal{B}$ by $v = (w, T, q, k)$.

The construction of the triangle $\triangle_{T^\circ,q^\circ}$ starts with using this marking procedure and building a copy $\textsc{Mark}(\square_{\emptyset,q^\circ}, \emptyset, q^\circ, 1)$ of the base box $\square_{\emptyset,q^\circ}$ which gets rank $k = 1$ and whose marked initial node $(\imath_{\square,\emptyset,q^\circ}, \emptyset, q^\circ, 1)$ becomes the initial node of $\triangle_{T^\circ,q^\circ}$. Along the construction of this triangle, an integer variable k counts the number of boxes already inserted in the triangle to make sure that all copies inserted get distinct ranks. The construction of the triangle $\triangle_{T^\circ,q^\circ}$ proceeds by successive insertions of copies of boxes according to the single following rule.

A new copy of the box $\square_{T',q'}$ is inserted into the triangle $\triangle_{T^\circ,q^\circ}$ in construction if there exists a node $v = (w, T, q, l)$ in the triangle in construction and a basic MSC $M \in \Sigma_{T^\circ}$ such that

T_1*: $\beta_{T,q}(w) \xrightarrow{M} q'$ in the MSG $\mathcal{G}_{T^\circ,q^\circ}$;*

T_2*: $T \subsetneq T' \subsetneq T^\circ$ and $T' = T \cup CG(M)$;*

T_3*: no edge labeled by M relates so far v to the initial node of some copy of $\square_{T',q'}$ in the triangle in construction.*

In that case an edge labeled by M is added in the triangle in construction from v to the initial node of the new copy of the box $\square_{T',q'}$.

Note here that Condition T_2 ensures that inserted boxes have height at most $|T^\circ| - 1$. By construction all copies of boxes inserted in a triangle are related in a tree-like structure built along the application of the above rule. It is easy to implement the construction of a triangle from boxes as specified by the insertion rule above by means of a list of inserted boxes whose possible successors have not been investigated, in a depth-first-search or breadth-first-search way. Note here that if a new copy of the box $\square_{T',q'}$ is inserted and connected from $v = (w, T, q, l)$ then $T \subsetneq T'$ thus the communication graph T grows along the branches of this tree-structure. This shows that this insertion process eventually stops and the resulting tree has depth at most $|T|$. Moreover, since we start from the empty box and edges in boxes $\square_{T,q}$ carry basic MSCs from Σ_T, we get the next key property.

LEMMA 5.2. *If a word $u \in \Sigma^\star$ leads in the triangle $\triangle_{T^\circ,q^\circ}$ from its initial node to some node $v = (w, T, q, l)$ then the communication graph of u is precisely T.*

Note also that it is easy to check that the mapping τ_{T°,q° induced by the data structure builds a morphism from $\triangle_{T^\circ,q^\circ}$ to $\mathcal{G}_{T^\circ,q^\circ}$. However this morphism may not be full in some cases. The role of boxes is precisely to take care of this drawback with the help of the next notion.

DEFINITION 5.3. *Let $T^\circ \subseteq \mathcal{I}^2$ be a subgraph of $\mathcal{I}^2$ and q°, q' be two nodes of $\mathcal{G}$. The set of missing edges $\textsc{Missing}(T^\circ, q^\circ, q')$ consists of all pairs (v, M) where $v = (w, T, q, l)$ is a node of $\triangle_{T^\circ,q^\circ}$ and M is a basic MSC such that*

- *$\beta_{T,q}(w) \xrightarrow{M} q'$ in the MSG $\mathcal{G}_{T^\circ,q^\circ}$;*
- *$T \subsetneq T \cup CG(M) = T^\circ$.*

Note here that the insertion rule T_2 for triangles forbids to insert a box $\mathcal{B}_{T^\circ,q}$ and to add an edge labeled by M from node v. This missing edge will be added into boxes of height $|T^\circ|$ in order to get a full morphism.

5.2 Building Boxes from Triangles

Boxes $\Box_{T^\circ,q^\circ}$ are made of triangles $\triangle_{T^\circ,q}$ associated with the same directed graph T°. Again several copies of the same triangle are often necessary to build a box and the structure relating these triangles plays a crucial role. For this reason we adopt a similar data structure: A node w of a box $\Box_{T^\circ,q^\circ}$ is a quadruple (v, T°, q, k) where v is a node of the triangle $\triangle_{T^\circ,q}$ and $k \in \mathbb{N}$. The rank k will allow us to distinguish between different copies of the same triangle. The construction of boxes uses here again an integer variable k that counts the number of triangles already inserted in the box in construction to make sure that all copies inserted get distinct ranks. On the other hand the parameter T is useless here but we keep it to get a uniform data structure.

As announced in the introduction of this section the construction of the box $\Box_{T^\circ,q^\circ}$ depends on the connectivity of T°. Recall that an instance $i \in \mathcal{I}$ is *active* in the directed graph $T^\circ \subseteq \mathcal{I}^2$ if there is an edge $(i,j) \in T^\circ$ or an edge $(j,i) \in T^\circ$ for some instance $j \neq i$. Moreover a directed graph $T^\circ \subseteq \mathcal{I}^2$ is *connected* if the symmetric closure of its restriction to its active instances is connected.

We assume first that $T^\circ \subseteq \mathcal{I}^2$ is a *non-connected* directed graph and define the box $\Box_{T^\circ,q^\circ}$. The definition of boxes with a connected directed graph is postponed to the next subsection. The construction of the box $\Box_{T^\circ,q^\circ}$ starts with building a copy $\textsc{Mark}(\triangle_{T^\circ,q^\circ}, T^\circ, q^\circ, 1)$ of the triangle $\triangle_{T^\circ,q^\circ}$ which gets rank $k = 1$ and whose marked initial node $(\imath_{\triangle,T^\circ,q^\circ}, \emptyset, q^\circ, 1)$ becomes the initial node of $\Box_{T^\circ,q^\circ}$. The construction of the box $\Box_{T^\circ,q^\circ}$ proceeds then by successive insertions of copies of triangles in a tree-like structure according to the following rule (which differs from [4]).

> *A new copy of the triangle $\triangle_{T^\circ,q'}$ is inserted into the box $\Box_{T^\circ,q^\circ}$ in construction if there exists a node $w = (v, T^\circ, q, l)$ in the box in construction and a basic MSC $M \in \Sigma_{T^\circ}$ such that we have $(v, M) \in$* $\textsc{Missing}(T^\circ, q, q')$ *and no edge labeled by M relates so far w to the initial node of some copy of $\triangle_{T^\circ,q'}$ in the box in construction. In that case an edge labeled by M is added in the box from w to the initial node of the new copy of the triangle $\triangle_{T^\circ,q'}$.*

At each step of this procedure we have a morphism from the box in construction to $\mathcal{G}$ which is encoded in the data-structure of nodes. In particular the initial node of each triangle $\triangle_{T^\circ,q}$ maps to node q of $\mathcal{G}$.

By means of Lemma 5.2 the definition of missing edges (Def. 5.3) leads us to the following property.

LEMMA 5.4. *Within a box $\Box_{T^\circ,q^\circ}$ associated with a* non-connected *graph T°, if a word $u \in \Sigma^\star$ leads from the initial node of a triangle to the initial node of another triangle then the communication graph of u is precisely T°.*

Recall now that T° is not connected and $\mathcal{G}$ is globally-cooperative. Therefore a branch of the tree-structure of a box in construction cannot involve twice the same triangle, otherwise we get a loop with communication graph T° in $\mathcal{G}$ which contradicts the definition of a globally-cooperative MSG. It follows that this procedure stops and the depth of the resulting tree-structure is at most $|Q|$. As a consequence the size of a box is exponential in the size of the given HMSC.

5.3 Building Boxes with a Connected Graph

We come now to the definition of boxes associated with a connected directed graph. This part is more subtle than the two previous constructions which have a tree-structure: Both do not create new loops in the unfolding. On the contrary the construction of boxes associated with a connected directed graph essentially deals with loops.

Let $T^\circ \subseteq \mathcal{I}^2$ be a connected (non-empty) directed graph. Basically the connected box $\square_{T^\circ,q^\circ}$ collects all triangles $\triangle_{T^\circ,q}$ for all nodes $q \in Q$. Each triangle is replicated a fixed number of times and copies of triangles are connected in some very specific way.

The construction of the box $\square_{T^\circ,q^\circ}$ consists in two steps. First m copies of each triangle $\triangle_{T^\circ,q}$ are inserted in the box. Moreover the first copy of $\triangle_{T^\circ,q^\circ}$ gets rank 1 and the first copy of its initial node becomes the initial node of the box in construction. The actual value of m will be discussed below. For simplicity's sake we require also that copies of the same triangle have consecutive ranks: In particular copies of $\triangle_{T^\circ,q^\circ}$ get ranks 1 to m. In a second step edges are added to connect these triangles to each other. The idea here is to take care of the missing edges in order to get a full morphism: For each triangle $\triangle_{T^\circ,q}$, for each node $q' \in Q$, and for each missing edge $(v, M) \in \textsc{Missing}(T^\circ, q, q')$ we add an edge labeled by M from each copy of node v to some copy of the initial node of triangle $\triangle_{T^\circ,q'}$.

In this process of connecting triangles we require two key properties:

C_1: *No added edge connects two nodes from the same copy of the same triangle: There is no added edge from node (v, T°, q, l) with rank l to $(\imath_{\triangle,T^\circ,q}, T^\circ, q, l)$.*

C_2: *At most one edge connects one copy of $\triangle_{T^\circ,q}$ to one copy of $\triangle_{T^\circ,q'}$: If we add from a copy of $\triangle_{T^\circ,q}$ of rank l an edge $(v_1, T^\circ, q, l) \xrightarrow{M_1} (\imath_{\triangle,T^\circ,q'}, T^\circ, q', l')$ and an edge $(v_2, T^\circ, q, l) \xrightarrow{M_2} (\imath_{\triangle,T^\circ,q'}, T^\circ, q', l')$ to the same copy of $\triangle_{T^\circ,q'}$ then $v_1 = v_2$ and $M_1 = M_2$.*

Condition C_1 requires simply two copies of each triangle. The number of added edges from a fixed copy of $\triangle_{T^\circ,q}$ to copies of $\triangle_{T^\circ,q'}$ is $|\textsc{Missing}(T^\circ, q, q')|$. It follows that the two conditions above require only

$$m = \max_{q,q' \in Q} |\textsc{Missing}(T^\circ, q, q')| + 1$$

copies of each triangle. The construction of the box $\square_{T^\circ,q^\circ}$ starts with the insertion of m copies of each triangle $\triangle_{T^\circ,q}$. Then for a fixed copy of $\triangle_{T^\circ,q}$ and for a fixed node q' we add at most m edges as follows: For each missing edge $(v, M) \in \textsc{Missing}(T^\circ, q, q')$ the copy of node v is connected to a distinct copy of the initial node of triangle $\triangle_{T^\circ,q'}$. In case $q = q'$ we make sure that v does not get connected along this process to the initial node of the triangle it belongs to.

From the definition of missing edges (Def. 5.3) it follows that the data-structure defines a morphism from the box $\Box_{T^\circ,q^\circ}$ to $\mathcal{G}_{T^\circ,q^\circ}$. Furthermore Lemma 5.2 yields the following useful property.

LEMMA 5.5. *Within a box $\Box_{T^\circ,q^\circ}$ associated with a* connected *graph T°, if a non-empty word $u \in \Sigma^\star$ leads from the initial node of a triangle to the initial node of a triangle then the communication graph of u is precisely T°.*

6 Properties of the Unfolding

6.1 Main Result

The constructions of triangles and boxes yield morphisms to $\mathcal{G}_{T,q}$ that are built inductively on the data-structure. These morphisms are useful in particular to check that the construction of a box with a non-connected directed graph eventually stops because $\mathcal{G}$ is globally-cooperative. We can also check by induction the following useful property.

LEMMA 6.1. *The morphism $\beta_{T,q}$ from a box $\mathcal{B}_{T,q}$ to $\mathcal{G}_{T,q}$ is full.*

Following Definition 5.1 the last box built yields the unfolding MSG $\mathcal{G}_{\mathrm{Unf}}$ together with a full morphism $\beta_{\mathrm{Unf}} : \mathcal{G}_{\mathrm{Unf}} \to \mathcal{G}$ which ensures that $L(\mathcal{G}_{\mathrm{Unf}}) = L(\mathcal{G})$. By Proposition 4.7 we have also $L(\mathcal{G}_{\mathrm{Unf}}) \subseteq L(\widehat{\mathcal{G}_{\mathrm{Unf}}})$. We will prove below that the converse inclusion relation holds (Lemma 6.6) by induction on the structure of boxes and triangles: Thus $L(\mathcal{G}) = L(\widehat{\mathcal{G}_{\mathrm{Unf}}})$. In that way we get our main result.

THEOREM 6.2. *For any globally-cooperative MSG $\mathcal{G}$ the unfolding MSG $\mathcal{G}_{\mathrm{Unf}}$ leads to a netchart $\widehat{\mathcal{G}_{\mathrm{Unf}}}$ such that $L(\mathcal{G}) = L(\widehat{\mathcal{G}_{\mathrm{Unf}}})$.*

Thus our unfolding procedure builds an unfolded globally-cooperative MSG for which the naive construction of a corresponding netchart yields a correct implementation of the specification.

6.2 Properties of Arched Firing Sequences

Let T be a non-empty subgraph of $\mathcal{I}^2$ and $q \in Q$. Let v be a node from the triangle $\mathcal{T}_{T,q}$. By construction of $\mathcal{T}_{T,q}$, v is a quadruple (w, T', q', k') such that w is a node from the box $\Box_{T',q'}$ and $k' \in \mathbb{N}$. Then we say that the *box location* of v is $l^\Box(v) = (T', q', k')$. We define the *sequence of boxes* visited along a path $s = v \xrightarrow{u} v'$ in $\mathcal{T}_{T,q}$ as follows:

– If the length of s is 0 then s corresponds to node v of $\mathcal{T}_{T,q}$ and $\mathbb{L}^\Box(s) = l^\Box(v)$.
– If s is a product $s = s' \cdot t$ where t is the edge $v'' \xrightarrow{a} v'$ then two cases appear:
  - If $l^\Box(v'') = l^\Box(v')$ then $\mathbb{L}^\Box(s) = \mathbb{L}^\Box(s')$;
  - If $l^\Box(v'') \neq l^\Box(v')$ then $\mathbb{L}^\Box(s) = \mathbb{L}^\Box(s').l^\Box(v')$.

Due to the tree-like structure of triangles we have the following obvious property.

PROPOSITION 6.3. *Let $\mathcal{T}_{T,q}$ be a triangle with T a non-empty subgraph of $\mathcal{I}^2$. Let s be an* arched *firing sequence of the low-level Petri net of $\widehat{\mathcal{T}_{T,q}}$. Then $\mathbb{L}^\Box(s{\downarrow}k) = \mathbb{L}^\Box(s{\downarrow}k')$ for each instance $k, k' \in \mathcal{I}$.*

Similarly to triangles, we define the *triangle location* $l^{\triangle}(w)$ of a node w in a box $\mathcal{B}_{T,q}$ and the *sequence of triangles* $\mathbb{L}^{\triangle}(s)$ visited along a path $s = w \xrightarrow{u} w'$ in $\mathcal{B}_{T,q}$. The tree-like structure of *unconnected* boxes yields a property similar to Proposition 6.3. We aim now at establishing a similar property for connected boxes (Prop. 6.5).

Let i, j be two distinct instances. For each firing sequence $s = \mathfrak{m}\,[u\rangle\, \mathfrak{m}'$ of the low-level Petri net of a netchart we define the projection of s on i w.r.t. (i, j) as the sequence of messages $send(s, i, j) = m_1...m_n$ such that the sequence of send actions from i to j in u consists of $i!^{m_1}j$,...,$i!^{m_n}j$. Similarly we define the projection of s on j w.r.t. (i, j) as the sequence of messages $receive(s, i, j) = m_1...m_n$ such that the sequence of receive actions on j from i in u consists of $j?^{m_1}i$,...,$j?^{m_n}i$. It is clear that if a firing sequence $s = \mathfrak{m}\,[u\rangle\, \mathfrak{m}'$ of the low-level Petri net of a netchart corresponds to a FIFO basic MSC then $send(s, i, j) = receive(s, i, j)$ for each pair of distinct instances $i, j \in \mathcal{I}$. This observation leads us to the next result.

LEMMA 6.4. *Let $\mathcal{B}_{T,q}$ be a box with T a non-empty* connected *subgraph of $\mathcal{I}^2$ and let i, j be two distinct instances such that $(i, j) \in T$. Let s be an* arched *firing sequence of the low-level Petri net of the netchart $\widehat{\mathcal{B}_{T,q}}$ that corresponds to a FIFO basic MSC. Then $\mathbb{L}^{\triangle}(s{\downarrow}i) = \mathbb{L}^{\triangle}(s{\downarrow}j)$.*

Proof. Since s is arched the first (resp. last) triangles coincide in $\mathbb{L}^{\triangle}(s{\downarrow}i)$ and $\mathbb{L}^{\triangle}(s{\downarrow}j)$. This result follows now from the three next observations. First, let m be a message in $send(s, i, j)$. Due to the the definition of a low-level Petri net, the message m corresponds to a unique transition $t = i!^{m}j$ in the low-level Petri net of the netchart $\widehat{\mathcal{B}_{T,q}}$ and moreover Comp(t) is an edge from $\mathcal{B}_{T,q}$. Thus the sequence of messages $send(s, i, j)$ maps in a natural way to a sequence of edges of the connected box $\mathcal{B}_{T,q}$ and consequently to the sequence of corresponding triangles. Second Lemma 5.5 ensures that at least one send action from i to j occurs when the path $s{\downarrow}i$ goes through a triangle of $\mathcal{B}_{T,q}$. Third, due to Condition C_1 of the construction of connected boxes, when the path $s{\downarrow}i$ goes out of a triangle then it enters into a *distinct* triangle.

These three facts imply that $send(s, i, j)$ is enough to recover the sequence of triangles $\mathbb{L}^{\triangle}(s{\downarrow}i)$ visited by i along s. A similar observation holds for the process j and $receive(s, i, j)$. We can now conclude easily. If s is an arched firing sequence of the low-level Petri net of $\widehat{\mathcal{B}_{T,q}}$ that corresponds to a FIFO basic MSC, then $send(s, i, j) = receive(s, i, j)$ hence $\mathbb{L}^{\triangle}(s{\downarrow}i) = \mathbb{L}^{\triangle}(s{\downarrow}j)$. ∎

PROPOSITION 6.5. *Let $\mathcal{B}_{T,q}$ be a box with T a non-empty connected subgraph of $\mathcal{I}^2$. Let $s = \mathfrak{m}\,[u\rangle\, \mathfrak{m}'$ be an* arched *firing sequence of the low-level Petri net of $\widehat{\mathcal{B}_{T,q}}$ that corresponds to a FIFO basic MSC M_s. Then there exists an* arched *firing sequence $s^{\dagger} = \mathfrak{m}\,\left[u^{\dagger}\right\rangle\, \mathfrak{m}'$ of the low-level Petri net of $\widehat{\mathcal{B}_{T,q}}$ that corresponds to a FIFO basic MSC $M_{s^{\dagger}}$ such that*

- $\pi^{\circ}(M_s) = \pi^{\circ}(M_{s^{\dagger}})$,
- $\mathbb{L}^{\triangle}(s^{\dagger}{\downarrow}k) = \mathbb{L}^{\triangle}(s^{\dagger}{\downarrow}k')$ *for each pair of instance $k, k' \in \mathcal{I}$.*

Proof. Consider first two *active* instances k and k' of T. Lemma 6.4 ensures that $\mathbb{L}^{\triangle}(s{\downarrow}k) = \mathbb{L}^{\triangle}(s{\downarrow}k')$ because T is connected. Now the processes that are not active in T produce in M only ϵ-actions because all transitions that may take place on these processes in the low-level Petri net of $\widehat{\mathcal{B}_{T,q}}$ are ϵ-events. Thus we can force them to behave like a fixed active process k of T. The result is an MSC $M_{s^\dagger}$ that differs from M only in ϵ-events located on non-active processes. Consequently, $\pi^\circ(M_s) = \pi^\circ(M_{s^\dagger})$ and $\mathbb{L}^\triangle(s^\dagger{\downarrow}k) = \mathbb{L}^\triangle(s^\dagger{\downarrow}k')$ for each non-active instance k' of T. ■

6.3 Main Technical Result

LEMMA 6.6. *We have* $L(\widehat{\mathcal{G}_{\mathrm{Unf}}}) \subseteq L(\mathcal{G}_{\mathrm{Unf}})$.

Proof. We proceed by induction. We show for each natural $n \in \{0, 1, 2, ..., |\mathcal{I}^2|\}$ the property $H(n)$ which consists of two similar sub-properties:

1. For each $T \subseteq \mathcal{I}^2$ with $1 \leqslant |T| \leqslant n+1$ and all nodes $q \in Q$ if $s = \mathfrak{m}_v\,[u\rangle\,\mathfrak{m}_{v'}$ is an arched firing sequence of the low-level Petri net of $\widehat{\mathcal{T}_{T,q}}$ that corresponds to a FIFO basic MSC M then $\pi^\circ(M)$ leads in $\mathcal{T}_{T,q}$ from v to v'.
2. For each $T \subseteq \mathcal{I}^2$ with $0 \leqslant |T| \leqslant n$ and all nodes $q \in Q$ if $s = \mathfrak{m}_v\,[u\rangle\,\mathfrak{m}_{v'}$ is an arched firing sequence of the low-level Petri net of $\widehat{\mathcal{B}_{T,q}}$ that corresponds to a FIFO basic MSC M then $\pi^\circ(M)$ leads in $\mathcal{B}_{T,q}$ from v to v'.

The proof of Lemma 6.6 follows from $H(n)$ with $n = |\mathcal{I}|^2$. The base case $H(0)$ is obvious because for each $q \in Q$ and each singleton T, the box $\mathcal{B}_{\emptyset,q}$ and the triangle $\mathcal{T}_{T,q}$ consist of a single node.

Induction step of H: We assume now $H(n)$. We show $H(n+1)$ for connected boxes only, but the cases of triangles and unconnected boxes are similar. We consider some connected subgraph $T \subseteq \mathcal{I}^2$ with $|T| = n+1$ and some node $q \in Q$. First, we prove by induction that for each natural $d \in \mathbb{N}$ the intermediate property $P(d)$ holds:

$P(d)$: Let L be a sequence of triangles of $\mathcal{B}_{T,q}$ such that $1 \leqslant |L| \leqslant d$. Let $s = \mathfrak{m}_v\,[u\rangle\,\mathfrak{m}_{v'}$ be an arched firing sequence of the low-level Petri net of $\widehat{\mathcal{B}_{T,q}}$ that corresponds to a FIFO basic MSC M. If $\mathbb{L}^\triangle(s{\downarrow}k) = L$ for each process $k \in \mathcal{I}$ then $\pi^\circ(M)$ leads in $\mathcal{B}_{T,q}$ from v to v'.

The base case $P(1)$ follows basically from the induction hypothesis $H(n)$ because in this case s can be viewed as an arched firing sequence of $\widehat{\mathcal{T}_{T,q}}$.

Induction step of P: We assume now that $P(d)$ holds and we prove $P(d+1)$. Let $L.l$ be a sequence of triangles with $|L.l| = d+1$ and let $s = \mathfrak{m}_v\,[u\rangle\,\mathfrak{m}_{v'}$ be an arched firing sequence of the low-level Petri net of $\widehat{\mathcal{B}_{T,q}}$ that corresponds to a FIFO basic MSC M such that $\mathbb{L}^\triangle(s{\downarrow}k) = L.l$ for each process $k \in \mathcal{I}$. Due to the structure of connected boxes, we claim that we can find an other arched firing sequence $s' = s_1 \cdot s_2 \cdot s_3$ for which $s_1 = \mathfrak{m}_v\,[u_1\rangle\,\mathfrak{m}_{v_1}$, $s_2 = \mathfrak{m}_{v_1}\,[u_2\rangle\,\mathfrak{m}_{v_2}$ and $s_3 = \mathfrak{m}_{v_2}\,[u_3\rangle\,\mathfrak{m}_{v'}$ are three arched firing sequences such that s_2 is non-empty and

S1. $u_1.u_2.u_3$ corresponds to a linear extension of M,
S2. each transition t that appears in u_3 comes from an edge Comp(t) of $\mathcal{B}_{T,q}$ that occurs within the last triangle l visited along s',
S3. each transition t that appears in u_2 satisfies Comp$(t) = a$ where a is the unique edge (by Condition C_2 of connected boxes) of $\mathcal{B}_{T,q}$ that relies the two last triangles visited along s'.

In particular, Condition S1 implies that $\mathbb{L}^{\triangle}(s{\downarrow}k) = \mathbb{L}^{\triangle}(s'{\downarrow}k)$ for each process $k \in \mathcal{I}$. Conditions S2 and S3 ensure that $\mathbb{L}^{\triangle}(s_3{\downarrow}k) = l$ and $\mathbb{L}^{\triangle}(s_1{\downarrow}k) = L$. Moreover, Remark 4.8 shows that s_1, s_2 and s_3 correspond respectively to some basic MSCs M_1, M_2 and M_3. Then by Condition S1 we have $M = M_1 \cdot M_2 \cdot M_3$. Therefore these three basic MSCs are FIFO because M is FIFO. Using the induction hypothesis $P(d)$ we deduce that $v \xrightarrow{\pi^\circ(M_1)} v_1$ and $v_2 \xrightarrow{\pi^\circ(M_3)} v'$ in $\mathcal{B}_{T,q}$. To conclude, we use Remark 4.8 with S3 and obtain that a is actually the edge $v_1 \xrightarrow{\pi^\circ(M_2)} v_2$ of $\mathcal{B}_{T,q}$. As a result $v \xrightarrow{\pi^\circ(M)} v'$ is a path of $\mathcal{B}_{T,q}$. This conclude the proof of $P(d+1)$.

We return now to the proof of $H(n+1)$. Let $s = \mathfrak{m}_v\,[u\rangle\,\mathfrak{m}_{v'}$ be an arched firing sequence of the low-level Petri net of $\widehat{\mathcal{B}_{T,q}}$ that corresponds to some FIFO basic MSC M. By Proposition 6.5, there exists an arched firing sequence $s^\dagger = \mathfrak{m}_v\,\left[u^\dagger\right\rangle\,\mathfrak{m}_{v'}$ of the low-level Petri net of $\widehat{\mathcal{B}_{T,q}}$ that corresponds to a FIFO basic MSC $M^\dagger$ such that (*) $\pi^\circ(M) = \pi^\circ(M^\dagger)$ and $\mathbb{L}^{\triangle}(s^\dagger{\downarrow}i) = \mathbb{L}^{\triangle}(s^\dagger{\downarrow}j) = L$ for each pair of instances $i, j \in \mathcal{I}$. Then we can apply $P(|L|)$ together with (*) to get that $v \xrightarrow{\pi^\circ(M)} v'$. This conclude the proof of $H(n+1)$. ■

References

1. Alur R. and Yannakakis M.: *Model Checking of Message Sequence Charts*. CONCUR, LNCS **1664** (1999) 114–129
2. Baudru N. and Morin R.: *Safe Implementability of Regular Message Sequence Charts Specifications*. Proc. of the ACIS 4th Int. Conf. SNDP (2003) 210–217
3. Baudru N. and Morin R.: *The Pros and Cons of Netcharts*. CONCUR, LNCS **3170** (2004) 99–114
4. Baudru N. and Morin R.: *Unfolding Synthesis of Asynchronous Automata*. Accepted to the International Computer Science Symposium in Russia (CSR), LNCS (2006) – 12 pages.
5. Bollig B. and Leucker M.: *Message-Passing Automata are expressively equivalent to EMSO Logic*. CONCUR, LNCS **3170** (2004) 146–160
6. Caillaud B., Darondeau Ph., Hélouët L. and Lesventes G.: *HMSCs as partial specifications... with PNs as completions*. LNCS **2067** (2001) 87–103
7. Diekert V. and Rozenberg G.: *The Book of Traces*. (World Scientific, 1995)
8. Genest B., Muscholl A., Seidl H. and Zeitoun M.: *Infinite-Node High-Level MSCs: Model-Checking and Realizability*. ICALP, LNCS **2380** (2002) 657–668
9. Genest B., Muscholl A. and Kuske D.: *A Kleene Theorem for a Class of Communicating Automata with Effective Algorithms*. DLT, LNCS **3340** (2004) 30–48
10. Henriksen J.G., Mukund M., Narayan Kumar K., Sohoni M. and Thiagarajan P.S.: *A theory of regular MSC languages*. Information and computation **202** (2005) 1–38
11. Holzmann G.J.: *Early Fault Detection*. TACAS, LNCS **1055** (1996) 1–13
12. ITU-TS: *Recommendation Z.120: Message Sequence Charts*. (Geneva, 1996)
13. Lamport L.: *Time, Clocks and the Ordering of Events in a Distributed System*. Communications of the ACM **21,7** (1978) 558–565
14. Métivier Y.: *On Recognizable Subsets of Free Partially Commutative Monoids*. Theor. Comput. Sci. **58** (1988) 201–208
15. Morin R.: *On Regular Message Sequence Chart Languages and Relationships to Mazurkiewicz Trace Theory*. FoSSaCS, LNCS **2030** (2001) 332–346
16. Morin R.: *Recognizable Sets of Message Sequence Charts*. STACS, 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. Ochmański E.: *Regular behaviour of concurrent systems*. Bulletin of the EATCS **27** (Oct. 1985) 56–67
20. Pratt V.: *Modelling concurrency with partial orders*. International Journal of Parallel Programming **15** (1986) 33–71
21. Thomas W.: *On Logics, Tilings, and Automata*. ICALP, LNCS **510** (1991) 441–454
22. Zielonka W.: *Notes on finite asynchronous automata*. RAIRO, Theoretical Informatics and Applications **21** (Gauthiers-Villars, 1987) 99–135