Checking non-divergence, channel-bound and global-cooperation using SAT-solvers

Florent Avellaneda & Rémi Morin
Laboratoire d'Informatique Fondamentale de Marseille — CNRS, UMR 6166 — Aix-Marseille université
163, avenue de Luminy, Case 901, F-13288 Marseille Cedex 9, France
Email: florent.avellaneda,remi.morin@lif.univ-mrs.fr

***Abstract*—Divergence appears in message sequence chart specifications when an unbounded number of messages are pending within a communication channel. Several algorithms and tools have been already developed to detect this property. This paper explains why checking non-divergence is very close to the Boolean satisfiability problem and shows how SAT-solvers can be used to check this property efficiently. We show also how some other close properties can be checked similarly.**

***Keywords*-Message sequence charts, SAT-solvers, buffer size, message-passing systems, realizability, formal methods**

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) and are close to other formalisms such as sequence diagrams of UML. An MSC gives a graphical description of the intended communications between processes. Usually it abstracts away from the values of variables and the actual contents of messages. However, this formalism can be used at some early stage of design to detect errors in specifications [1], [2], [3], [5], [7], [8], [12], [13]. In this work we focus on detecting *process divergence*, as introduced in [2], verifying the *buffer size* of channels, as already investigated in [12], and checking *global-cooperation* [7], [8].

Message sequence graphs (MSGs), or equivalently high-level MSCs, are a usual formalism to describe possibly infinite sets of scenarios in some algebraic way. Numerous tools have already been developed to draw specifications of protocols in the form of MSGs. Some of them implement also formal verification techniques, see e.g. [3], [5], [10]. For the basic properties we consider, an almost instantaneous checking for relatively small MSGs would be appreciated. However, since MSGs can be defined in a modular way, dealing with large graphs efficiently would be also welcome.

Because asynchronous distributed systems provide no information about the relative speed of processes or the delay for a message to be delivered, *divergence* can appear in specifications: This means that there is no bound for the number of pending messages along an execution of specified scenarios. However a simple criterion allows us to decide whether a given MSG is not divergent [2, Th. 5]: We have to check that all connected components of the communication graph of each loop are strongly connected. In [2], Ben-Abdallah and Leue derived from their criterion an algorithm to check divergence, which is *quadratic in the number of processes* but exponential in the number of nodes. One has simply to search for all simple loops and to compute the strongly connected components of the resulting communication graphs. It follows that checking divergence is in NP. Now an interesting alternative algorithm was suggested in [1]: It consists in first fixing a diverging channel and some associated partition of processes and next searching for a simple loop matching these message exchanges, by computing the strongly connected components of the MSG restricted to the matching nodes. This second approach is *quadratic in the number of nodes* but exponential in the number of processes. These two simple algorithms are somehow opposite to each other but each may be efficient in some cases, according to the number of nodes and processes. For that reason, designing an efficient algorithm for checking divergence may be difficult.

Detecting divergence in MSGs is actually known to be NP-complete [1, Th. 7]. In this paper we give a quadratic reduction from divergence to the Boolean satisfiability problem (SAT) which allows us to use SAT-solvers to detect divergence in MSC specifications. In that way we take advantage of all the works on SAT-solvers. Still we do not claim that any NP-complete problem should be solved by means of reductions to SAT. We give also a linear reduction from SAT to divergence which shows that Divergence and SAT are formally close problems: Consequently efficient SAT-solvers should be able to detect divergence efficiently. This statement is confirmed by some preliminary experiments over small and large MSGs which show that the *cost* of reducing to SAT appears to be *constant* in practice.

Given a non-divergent MSG, a natural issue is to compute a *buffer size* for channels so that any pending message can be stored within the system before it gets delivered. As established by Lohrey and Muscholl, checking whether a buffer size is correct for all scenarios of a (possibly divergent) MSG is co-NP-complete [12, Th. 4.6]. Since the MSG used in the proof of this theorem shows no loop, this result extends to the particular case of non-divergent MSGs. Thus, computing an optimal appropriate buffer size for a non-divergent MSG is hard. In this paper we present a way to reduce to SAT the verification of a buffer size for a given

channel. Further this technique can be extended to weighted MAX-SAT solvers in order to compute the *optimal buffer size* for a given channel.

Another key property of MSGs called *global-cooperation* [7], [8] has been investigated in the literature: It requires that the communication graph of any loop is connected. Similarly to non-divergence, checking global-cooperation is co-NP-complete and two natural algorithms can be designed to solve this problem. Globally-cooperative and non-divergent MSGs are often called locally synchronized [13] or bounded [1]. They benefit from specific model-checking techniques and are known to be implementable by message-passing systems [11]. That is why global-cooperation is an important property to check while designing a protocol with MSGs. Similarly to divergence, we present a quadratic reduction from Global-Cooperation to SAT.

This paper is organized as follows. In the next section we introduce the formalism of MSCs and MSGs together with the notion of communication graph. In Section II, we recall the three basic properties we want to check. Next Section III presents all our reductions to SAT that we use to check non-divergence, global-cooperation, and channel-bound. Section IV provides some numerical comparisons to other tools, namely MSCan [5] and SOFAT [10]. Finally Section V analyses the cost of these reductions to check properties of MSGs.

I. Protocol specifications with message sequence charts

Following a classical approach in concurrency theory the executions of a distributed system are regarded as labeled partial orders called Message Sequence Charts (MSCs). In this paper, similarly to [6], [7], [11] the actual content of messages is abstracted from the notion of MSCs. For simplicity's sake we consider also, as usual, MSCs for which each sending corresponds to some receipt. However the results and techniques presented here could be extended to the framework of compositional MSCs [9].

A. Labeled partial orders (or partial words)

A *partial word* 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 partial word can be seen as an abstraction of an execution of a concurrent system. Then the elements of E are *events* and the letter $\xi(e)$ describes the basic action that is performed by event $e \in E$. Furthermore, the order $\preccurlyeq$ describes the causal dependence between events. A *linear extension* of t is a total order $\leqslant$ over E that respects the causal order, i.e. $\preccurlyeq \subseteq \leqslant$. It corresponds intuitively to a sequential view of the execution t. By $\mathrm{LE}(t) \subseteq \Sigma^\star$, we denote the set of linear extensions of a partial word t over Σ.

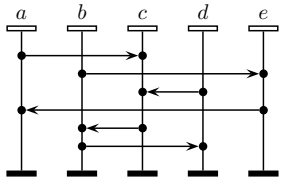

Figure 1. A Message Sequence Chart

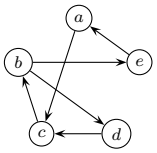

Figure 2. and its communication graph

B. Message sequence charts

Message sequence charts are defined by the Z.120 recommendation of the ITU-T with a formal syntax and graphical rules (Fig. 1). They can be seen also as particular partial words over some alphabet that we introduce first. Let $\mathcal{I}$ be a finite set of processes (also called *instances*). For any instance $i \in \mathcal{I}$, the alphabet Σ_i is the disjoint union of the set of *send actions* $\Sigma_i^! = \{i!j \mid j \in \mathcal{I} \setminus \{i\}\}$ and the set of *receive actions* $\Sigma_i^? = \{i?j \mid j \in \mathcal{I} \setminus \{i\}\}$. The alphabets Σ_i are disjoint and we let $\Sigma_\mathcal{I} = \bigcup_{i \in \mathcal{I}} \Sigma_i$. Given an action $a \in \Sigma_\mathcal{I}$, we denote by $\mathrm{Ins}(a)$ the unique instance i such that $a \in \Sigma_i$, that is the particular instance on which each occurrence of action a occurs. Finally, for any partial word $(E, \preccurlyeq, \xi)$ over $\Sigma_\mathcal{I}$ and any $e \in E$, we denote by $\mathrm{Ins}(e)$ the instance on which the event e occurs : $\mathrm{Ins}(e) = \mathrm{Ins}(\xi(e))$.

We denote by $\mathcal{K} = \{(i,j) \in \mathcal{I} \times \mathcal{I} \mid i \neq j\}$ the set of all channels. Let $M = (E, \preccurlyeq, \xi)$ be a partial word over $\Sigma_\mathcal{I}$. We denote by $\#^a(M)$ the number of events $e \in E$ such that $\xi(e) = a$ and by $\downarrow_M f = \{e \in E \mid e \preccurlyeq f\}$ the downward-closed set of events below f. Now two events $e, f \in E$ match each other if e sends a message from i to j and f receives this message on j: Formally, we put $e \rightsquigarrow f$ if $\xi(e) = i!j$, $\xi(f) = j?i$, and $\#^{i!j}(\downarrow_M e) = \#^{j?i}(\downarrow_M f)$. For any two events $e, f \in E$ we write $e \mathrel{—\!\!\prec} f$ if $e \prec f$ and $e \prec f' \preccurlyeq f$ implies $f' = f$, i.e. e is below f in the Hasse diagram of M.

Definition I.1 *An* MSC *is a partial word $M = (E, \preccurlyeq, \xi)$ over $\Sigma_\mathcal{I}$ 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*:* $\forall e, f \in E$*:* $e \rightsquigarrow f \Rightarrow e \preccurlyeq f$
M_3*:* $\forall e, f \in E$*:* $[e \mathrel{—\!\!\prec} f \wedge \mathrm{Ins}(e) \neq \mathrm{Ins}(f)] \Rightarrow e \rightsquigarrow f$
M_4*:* $\forall (i,j) \in \mathcal{K} : \#^{i!j}(M) = \#^{j?i}(M)$

Condition M_1 asserts that events occurring on the same instance are linearly ordered: Non-deterministic choice cannot be described within an MSC. Axiom M_2 formalizes that the receipt of any message will occur after the corresponding

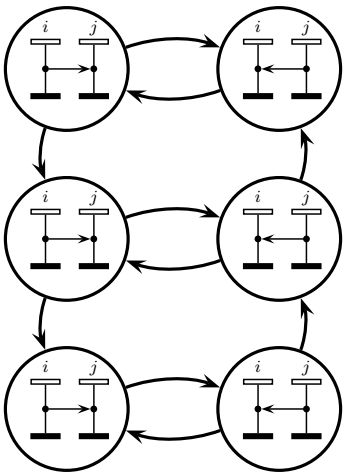

Figure 3. Simplified sliding window protocol (window size=3)

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 M_4 ensures that there is a one-to-one correspondence between send and receive actions for each channel. We denote by $\mathbb{MSC}$ the set of all MSCs.

C. Message sequence graphs and communication graphs

We come now to the definition of the concatenation of two MSCs. Let $M_1 = (E_1, \preccurlyeq_1, \xi_1)$ and $M_2 = (E_2, \preccurlyeq_2, \xi_2)$ be two MSCs. The product $M_1 \cdot M_2$ is the MSC $(E, \preccurlyeq, \xi)$ where $E = E_1 \uplus E_2$, $\xi = \xi_1 \cup \xi_2$ and the partial order $\preccurlyeq$ is the transitive closure of $\preccurlyeq_1 \cup \preccurlyeq_2 \cup \{(e_1, e_2) \in E_1 \times E_2 \mid \mathrm{Ins}(e_1) = \mathrm{Ins}(e_2)\}$.

DEFINITION I.2 *[1], [5], [8], [11], [12], [13] A* Message Sequence Graph *(for short: an MSG) is quadruple $\mathcal{G} = (V, \imath, \rightarrow, \mu)$ that consists of a finite set of nodes V with a distinguished initial node $\imath \in V$, a set of edges $\rightarrow \subseteq V \times V$, and a labeling $\mu : V \rightarrow \mathbb{MSC}$ which maps each node to some MSC.*

An example of an MSG describing a simplified sliding window protocol is depicted in Fig. 3. The language $\mathcal{L}(\mathcal{G})$ recognized by an MSG $\mathcal{G}$ collects all MSCs M for which there exists a path $\imath = v_0 \rightarrow \ldots \rightarrow v_n$ such that $M = \mu(v_0) \cdot \ldots \cdot \mu(v_n)$. For convenience we assume that all nodes of an MSG are reachable from its initial node. MSGs are usually provided with a subset $F \subseteq V$ of final nodes. In that case the definition of $\mathcal{L}(\mathcal{G})$ requires additionally that $v_n \in F$ and we would require then that each node is co-reachable from the final nodes. However this feature plays no role in our study, so we simply omit it here.

DEFINITION I.3 *[2], [8], [11] The* communication graph *of an MSC M is the directed graph whose nodes are the processes which send or receive a message in M (called the* active processes *of M) and such that there is an edge from i to j whenever M specifies a message from i to j.*

For instance the communication graph of the MSC from Fig. 1 is depicted on Fig. 2. Observe here that this graph is strongly connected. Given an MSG $\mathcal{G}$ and any subset of nodes $S \subseteq V$, the communication graph of S is simply the union of the communication graphs of MSCs carried by nodes from S. The communication graph of a subset of edges $L \subseteq V \times V$ is the communication graph of the corresponding set of nodes. In particular the communication graph of a loop in $\mathcal{G}$ is the communication graph of the subset of nodes occurring in this loop.

II. THREE KEY PROPERTIES TO CHECK

In this section we recall the definition of three basic properties that have been already investigated in the literature, namely channel-bound [12], divergence [2] and global-cooperation [7], [8].

A. Channel-bound and buffer size

Let $M = (E, \preccurlyeq, \xi)$ be an MSC. An execution of M is a sequential view of its events, that is, a linear extension $s = (E, \leqslant, \xi)$ of M. Each stage of this execution corresponds to a prefix $t \leqslant s$. Now the maximal number of messages pending within the channel from i to j along the execution s is $\max_{t \leqslant s}(\#^{i!j}(t) - \#^{j?i}(t))$. Since this value depends on s, the *width* of M for channel (i, j) is defined as follows:

$$W_{i,j}(M) = \max_{s \in \mathrm{LE}(M)} \max_{t \leqslant s} (\#^{i!j}(t) - \#^{j?i}(t))$$

captures the maximal number of messages pending at any stage of any execution of M. For an MSG $\mathcal{G}$, we consider the least upper bound $W_{i,j}(\mathcal{G}) = \sup_{M \in \mathcal{L}(\mathcal{G})} W_{i,j}(M)$. Note here that the MSC language $\mathcal{L}(\mathcal{G})$ is infinite as soon as $\mathcal{G}$ contains loops. Thus $W_{i,j}(\mathcal{G})$ can be infinite, too. The value $W_{i,j}(\mathcal{G})$ is interesting since it represents the optimal size of the buffer from i and j so that no overflow can occur in any execution of $\mathcal{G}$.

DEFINITION II.1 *A channel (i, j) is* bounded *in an MSG $\mathcal{G}$ if $W_{i,j}(\mathcal{G}) < \infty$. An MSG $\mathcal{G}$ is channel-bounded if all channels are bounded in $\mathcal{G}$.*

To our knowledge, the problem of computing $W_{i,j}(\mathcal{G})$ efficiently in practice has not been considered yet in the literature. However it is proved in [12, Th. 4.6] that computing the optimal buffer size $\max_{(i,j) \in \mathcal{K}} W_{i,j}(\mathcal{G})$ is NP-difficult. More precisely checking whether some given buffer size is larger than this optimal size is co-NP-complete.

B. Divergence vs. bounded channel width

An MSG is called divergent if $W_{i,j}(\mathcal{G}) = \infty$ for some channel (i, j), which means that the number of potentially pending messages is unbounded. A nice criterion to detect divergence in MSGs was established in [2, Th. 5]. For

convenience we introduce here this characterization for some fixed channel.

DEFINITION II.2 *A channel (i,j) is* divergent *in some MSG $\mathcal{G}$ if there exists some loop whose communication graph contains an edge from i to j but no path from j to i.*

An MSG is called *divergent* if there exists a loop whose communication graph contains a connected component that is not strongly connected. Thus an MSG $\mathcal{G}$ is *divergent* iff at least one channel is divergent in $\mathcal{G}$. This definition coincides with the criterion introduced in [2]. It is clear that detecting divergence in an MSG is in NP. Actually checking non-divergence is known to be in co-NP-complete [1, Th. 7].

C. Global-cooperation and system synthesis

Globally-cooperative MSGs play an important role in the theory of MSCs because they can be implemented in the form of a message-passing system with possibly unbounded buffers by means of some control information added to messages [8]. Moreover if the MSG is also non-divergent then this system requires only bounded buffers [11]. Again we introduce the notion of global-cooperation for a fixed channel.

DEFINITION II.3 *A channel (i,j) is* globally-cooperative *in some MSG $\mathcal{G}$ if i and j are connected in the communication graph of any loop for which both i and j are active.*

As opposed to divergence, global-cooperation is a usual requirement to guarantee a nice behaviour of the system. For that reason, an MSG $\mathcal{G}$ is called *globally-cooperative* if *all* channels are globally-cooperative in $\mathcal{G}$. Note here that the channel (i,j) is globally-cooperative as soon as (j,i) is globally-cooperative. Furthermore an MSG is globally-cooperative iff the communication graph of any loop is connected [7], [8]. Checking whether an MSG is globally-cooperative is known to be co-NP-complete [8, Prop. 6].

III. USING SAT-SOLVERS

In this section we fix some MSG $\mathcal{G} = (V, \imath, \Delta, \mu)$. We explain here how to use SAT-solvers to detect divergence in a channel and to check global-cooperation of $\mathcal{G}$. We also explain how to check whether a given buffer size is appropriate for some channel. Further we present a way to compute the optimal buffer size of a channel.

A. From Divergence to SAT

Our formal reduction from divergence to SAT makes use of the following basic observation.

PROPOSITION III.1 *A channel (i,j) is divergent in the MSG $\mathcal{G}$ if and only if there exists a set of nodes L that forms* disjoint *simple loops and whose communication graph contains an edge from i to j but no path from j to i.*

Let (i,j) be a fixed channel. We shall build a Boolean formula $\Phi_{\text{div}}(\mathcal{G},i,j)$ with two kinds of variables: The edges $\delta \in \Delta$ and the processes $k \in \mathcal{I}$. For any edge $\delta = q \to q'$ from Δ we let $\mathrm{dom}(\delta) = q$ and $\mathrm{cod}(\delta) = q'$ denote the domain and the codomain of δ. Moreover we write $s \to_\delta r$ if the MSC carried by q or q' shows a message from s to r.

The edge variables δ assigned to true correspond to a subset of edges which will form a set of disjoint simple loops in any assignment satisfying $\Phi_{\text{div}}(\mathcal{G},i,j)$. To do so we observe first that a subset of edges forms a set of disjoint simple loops if and only if the three next properties are satisfied:

- an edge leaves a node iff some edge enters this node,
- there is at most one edge leaving a given node,
- there is at most one edge entering a given node.

These conditions can be formalized by the following Boolean conditions:

$$\mathsf{Disjoint\_Loops} = \bigwedge \begin{cases} \displaystyle\bigwedge_{\delta\in\Delta} \left(\delta \to \bigvee_{\mathrm{dom}(\delta')=\mathrm{cod}(\delta)} \delta' \right) \\ \displaystyle\bigwedge_{\delta\in\Delta} \left(\delta \to \bigvee_{\mathrm{cod}(\delta')=\mathrm{dom}(\delta)} \delta' \right) \\ \displaystyle\bigwedge_{\delta\neq\delta' \text{ and } \mathrm{dom}(\delta)=\mathrm{dom}(\delta')} (\neg\delta \vee \neg\delta') \\ \displaystyle\bigwedge_{\delta\neq\delta' \text{ and } \mathrm{cod}(\delta)=\mathrm{cod}(\delta')} (\neg\delta \vee \neg\delta') \end{cases}$$

We will also distinguish a subset of processes, those assigned to true. The formula will require that all processes reachable from the fixed process j in the communication graph of the selected edges are assigned to true (but possibly some other processes, too). In particular the variable j should be assigned to true. This requirement can be formalized in CNF as follows:

$$\mathsf{Reachable}_j = j \wedge \bigwedge_{s\to_\delta r} (\delta \wedge s) \to r$$

This expresses the fact that if process s is reachable from j and δ is selected then r is reachable from j, provided that the domain or codomain of δ carries an MSC that specifies a message exchange from s to r.

Recall now that the channel (i,j) is divergent in $\mathcal{G}$ if there exists a set of disjoint simple loops in $\mathcal{G}$ whose communication graph shows a direct message exchange from i to j but no path from j to i (Prop. III.1), which means that the variable i must be assigned to false. Thus it remains only to express the following condition

$$\mathsf{Divergence}_{i,j} = (\neg i) \wedge \bigvee_{i\to_\delta j} \delta$$

To conclude we consider the formula

$$\Phi_{\text{div}}(\mathcal{G},i,j) = \mathsf{Disjoint\_Loops} \wedge \mathsf{Reachable}_j \wedge \mathsf{Divergence}_{i,j}.$$

THEOREM III.2 *The channel (i,j) is divergent in $\mathcal{G}$ iff the Boolean formula $\Phi_{div}(\mathcal{G},i,j)$ is satisfiable.*

Note that the number of clauses in $\Phi_{\text{div}}(\mathcal{G}, i, j)$ is at most $3 + 2 \times |\Delta| + 2 \times |\Delta|^2 + |\mathcal{I}|^2 \times |\Delta|$. Moreover the number of literals in each clause is at most $1+|\Delta|$. Interestingly, in case (i, j) is divergent then any assignment satisfying $\Phi_{\text{div}}(\mathcal{G}, i, j)$ provides a counter-example for the non-divergence of this channel in the form of a simple loop. Thus, in order to detect divergence in an MSG $\mathcal{G}$ we need simply to use a SAT-solver and ask about the satisfiability of $\Phi_{\text{div}}(\mathcal{G}, i, j)$ for all channels (i, j).

REMARK III.3 The formula Disjoint_Loops ensures that the edges assigned to true form disjoint loops in $\mathcal{G}$. Therefore the codomain of any of these edges is also the domain of some of these edges. Consequently, in order to avoid redundancy, we can restrict the conjunction in $\mathsf{Reachable}_j$ and the disjunction in $\mathsf{Divergence}_{i,j}$ to MSCs appearing in the domains of these edges. Another basic improvement would be to compute first the strongly connected components of $\mathcal{G}$ and check each component separately. These two remarks will be valid below for global-cooperation, too.

REMARK III.4 In order to detect divergence in $\mathcal{G}$, we can avoid checking each channel separately. For this we design a formula $\Psi_{\text{div}}(\mathcal{G})$ that is satisfiable if and only if $\Phi_{\text{div}}(\mathcal{G}, i, j)$ is satisfiable for some channel (i, j). To do so, we consider $2 \times |\mathcal{I}|$ new variables $(i_k)_{k\in\mathcal{I}}$ and $(j_k)_{k\in\mathcal{I}}$. We require that exactly one process i_k (resp. j_k) is assigned to true. Intuitively i_k is assigned to true if $k = i$ and j_k is assigned to true if $k = j$. Now $\Psi_{\text{div}}(\mathcal{G})$ is obtained from $\Phi_{\text{div}}(\mathcal{G}, i, j)$ by replacing the clause $\neg i$ by $\bigwedge_{k\in\mathcal{I}}(i_k \to \neg k)$, the clause j by $\bigwedge_{k\in\mathcal{I}}(j_k \to k)$, and the clause $\bigvee_{i\to_\delta j} \delta$ by $\bigwedge_{k,l\in\mathcal{I}}((i_k \wedge j_l) \to \bigvee_{k\to_\delta l} \delta)$.

B. From Global-Cooperation to SAT

Observe first that it is easy to remove from the MSG $\mathcal{G}$ all nodes labeled by an empty MSC and get an equivalent MSG: The removal of empty nodes preserves and reflects global-cooperation of each channel. It may lead to multiple initial states, but this issue is not relevant since the initial state plays no role in the property we consider. For that reason *we shall assume here that no node of $\mathcal{G}$ maps to the empty MSC.* Our formal reduction from Global-Cooperation to SAT makes use of the following fact.

PROPOSITION III.5 *Assume that no node of $\mathcal{G}$ is labeled by the empty MSC. Then $\mathcal{G}$ is globally-cooperative if and only if the communication graph of any* simple *loop is connected.*

EXAMPLE III.6 Consider the MSG with three nodes from Fig. 4. This MSG is *not* globally-cooperative. However the communication graph of any simple loop is connected. This shows that the removal of empty nodes in $\mathcal{G}$ is crucial for Proposition III.5.

REMARK III.7 The MSG from Figure 5 shows that Prop. III.5 is no longer valid if one considers *a fixed channel*.

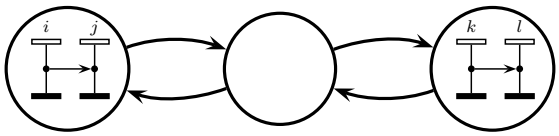

Figure 4. MSG from Example III.6

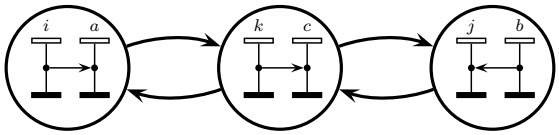

Figure 5. MSG from Rmk III.7

More precisely the channel (i, j) is *not* globally-cooperative in this MSG, but i and j are connected in the communication graph of any simple loop for which both i and j are active (since there is no such simple loop).

In order to check that $\mathcal{G}$ is globally-cooperative we adopt first the following approach: We check that for all distinct processes i and j, for all simple loops in $\mathcal{G}$, if i and j are both active then they are connected. Let us fix a channel (i, j). We build a Boolean formula $\Phi_{\overline{\text{gc}}}(\mathcal{G}, i, j)$ which is satisfiable iff there exists a *simple* loop for which i and j are active but not connected. The formula $\Phi_{\overline{\text{gc}}}(\mathcal{G}, i, j)$ uses $|\mathcal{I}| + N \times |\Delta|$ variables, where $N = |V|$ is the number of nodes in $\mathcal{G}$. Each process $k \in \mathcal{I}$ corresponds to a variable and for each $x \in [1, N]$ and each edge $\delta \in \Delta$ we consider a variable δ_x. For convenience we write $\delta_N = \delta$ for each $\delta \in \Delta$. The formula $\Phi_{\overline{\text{gc}}}(\mathcal{G}, i, j)$ consists of four parts and borrows the specification of Disjoint_Loops above:

$$\begin{aligned}\Phi_{\overline{\text{gc}}}(\mathcal{G}, i, j) \quad = \quad & \mathsf{Disjoint\_Loops} \wedge \mathsf{Junction} \\ & \wedge \mathsf{Connected}_j \wedge \mathsf{Unconnected}_{i,j}\end{aligned}$$

Thus we require again that the edges δ assigned to true form a collection of disjoint simple loops. We will formalize by Junction the property that these edges are *connected*: Consequently either they form a simple loop or no edge is assigned to true at all.

$$\mathsf{Junction} = \bigwedge \begin{cases} \bigwedge\limits_{\delta,\delta'\in\Delta \text{ with } \delta\neq\delta'} & \neg\delta_1 \vee \neg\delta'_1 \\ \bigwedge\limits_{\delta\in\Delta \text{ and } x\in[1,N-1]} & \delta_x \to \delta_{x+1} \\ \bigwedge\limits_{\delta\in\Delta \text{ and } x\in[2,N]} & (\delta_x \wedge \neg\delta_{x-1}) \to \bigvee\limits_{\text{cod}(\delta')=\text{dom}(\delta)} \delta'_{x-1} \end{cases}$$

This formula states that

- at most one edge δ is assigned to true at the first level; intuitively this edge can be regarded as the first step of a loop.
- any edge assigned to true at some level $x < N$ must be assigned to true at level $x + 1$: The set of selected edges increases at each level.

- the domain of any edge assigned to true at some level $x > 1$ but not at level $x-1$ should be the codomain of some edge assigned to true at level $x-1$. This ensures that the set of selected edges at each level is connected.

Since each simple loop in $\mathcal{G}$ contains at most N edges, any simple loop can be described according to this formula. Conversely any assignment satisfying $\mathsf{Disjoint\_Loops} \wedge \mathsf{Junction}$ describes a simple loop in $\mathcal{G}$ (or nothing at all).

Similarly to the formula $\mathsf{Reachable}_j$, we require now that all processes reachable or co-reachable from j be assigned to true. This is guaranteed by $\mathsf{Connected}_j$ below.

$$\mathsf{Connected}_j = j \wedge \bigwedge_{s \to_\delta r} ((\delta \wedge s \to r) \wedge (\delta \wedge r \to s))$$

Clearly any process in the connected component of j within the communication graph of the selected edges must be assigned to true in any assignment satisfying $\mathsf{Connected}_j$.

It remains to require that i and j are active processes in the selected simple loop, but i is assigned to false. This means that i and j appear in the communication graph of the simple loop but they are not connected. This condition is formalized by $\mathsf{Unconnected}_{i,j}$:

$$\mathsf{Unconnected}_{i,j} = \left(\bigvee_{s \to_\delta j \text{ or } j \to_\delta r} \delta \right) \wedge \left(\bigvee_{s \to_\delta i \text{ or } i \to_\delta r} \delta \right) \wedge \neg i$$

Note that this condition ensures that at least one edge is assigned to true in any satisfying assignment, so the edges assigned to true form a simple loop.

THEOREM III.8 *Provided that no node carries the empty MSC, the MSG $\mathcal{G}$ is* not *globally-cooperative iff the Boolean formula $\Phi_{\overline{\text{gc}}}(\mathcal{G}, i, j)$ is satisfiable for some channel (i, j).*

It is easy to see that there are at most $4 + 2 \times N \times |\Delta| + 3 \times |\Delta|^2 + 2 \times |\mathcal{I}|^2 \times |\Delta|$ clauses in $\Phi_{\overline{\text{gc}}}(\mathcal{G}, i, j)$. Moreover these clauses have at most $2 + |\Delta|$ literals.

REMARK III.9 Provided that we have an enumeration v_1, ..., v_N of all nodes, we could require additionally the minimality of the index of the codomain of the edge δ such that δ_1 is assigned to true. In that way all *symmetries* generated by $\mathsf{Junction}$ vanish, which may help the SAT-solver in some cases.

REMARK III.10 Similarly to Remark III.4, we can avoid checking each channel separately by building a single formula $\Psi_{\overline{\text{gc}}}(\mathcal{G})$ that is satisfiable iff $\Phi_{\overline{\text{gc}}}(\mathcal{G}, i, j)$ is satisfiable for some channel (i, j), i.e. $\mathcal{G}$ is *not* globally-cooperative.

C. From Channel-Bound to SAT

Let $B \in \mathbb{N}$ be a buffer size. The Channel-Bound problem for a *non-divergent* MSG $\mathcal{G}$, a buffer size B, and a channel (i, j) consists in checking whether $W_{i,j}(\mathcal{G}) \leqslant B$, i.e. no more than B messages may be pending within channel (i, j) in any execution of $\mathcal{G}$. The next result shows how to detect $B + 1$ messages in the fixed channel (i, j).

PROPOSITION III.11 *Let $\mathcal{G}$ be a non-divergent MSG with N nodes. If $W_{i,j}(\mathcal{G}) > B$ then there is a path $v_1 \to ... \to v_n$ of length $n \leqslant N \times |\mathcal{I}|$ such that $W_{i,j}(\mu(v_1) \cdot ... \cdot \mu(v_n)) > B$.*

Proof. Assume that $W_{i,j}(\mathcal{G}) \geqslant B + 1$. Then there exists some MSC $M = (E, \preccurlyeq, \xi)$, a linear extension $s = (E, \leqslant, \xi)$, and a prefix $s' \leqslant s$ such that $\#^{i!j}(s') - \#^{j?i}(s') \geqslant B + 1$. The MSC M is the product of MSCs carried by successive nodes from $\mathcal{G}$ along a path $\imath = v_1 \to ... \to v_n$ which starts from the initial node. We put $M_p = \mu(v_p)$ for each $p \leqslant n$. Thus we have $M = M_1 \cdot ... \cdot M_n$. Events from s' correspond to prefixes $M'_p \leqslant M_p$. Let M_q be the first MSC such that some message is sent in channel (i, j) but not received in M'_q. Let M_r be the last MSC such that some message is sent in channel (i, j) but not received in M'_q. We may assume that $M'_p = M_p$ for all $p < q$ and $r = n$. Since there are N nodes, we may assume also that $q \leqslant N$. We can now drop the requirement that $v_0 = \imath$ by considering intuitively that any node is initial. In other words we may assume that $q = 1$.

Let $J_p \subseteq \mathcal{I}$ be the subset of processes k such that some event in $M_1 \cdot ... \cdot M_p$ occurring on k does not belong to s'. In that case we say that k is *stuck in* M_p. Clearly $J_p \subseteq J_{p+1}$ for all p. Moreover $j \in J_1$ and $i \notin J_{r-1}$ because s' contains some sending from i to j in M_r. Let $p > 0$. We may assume that $k \in J_p \setminus J_{p-1}$ iff there exists some event e on k in M_p that depends causally on some event f in M_p such that f occurs on some process l that is stuck in M_{p-1}, i.e. $l \in J_{p-1}$ and $f \preccurlyeq e$. This means simply that we consider M' to be maximal in some sense: Processes stuck in M_p depend causally on processes stuck in M_{p-1}.

Assume now that we have $v_p = v_{p+l}$ with $J_p = J_{p+l}$ and $p + l \leqslant r$. Then the loop $v_p \to ... \to v_{p+l}$ contains no message from i to j. Otherwise its communication graph shows an edge from i to j but no path from j to i whereas $\mathcal{G}$ is not divergent. In that case we can remove this loop from the path we have considered. Consequently we may assume that $r \leqslant N \times |\mathcal{I}|$. ∎

Thus we need to explore only bounded paths in $\mathcal{G}$, similarly to bounded model-checking [4]. Now we can design a Boolean formula $\Phi_{\overline{\text{cb}}}(\mathcal{G}, i, j, B)$ that is satisfiable if and only if there exists some path in $\mathcal{G}$ of length at most $N \times |\mathcal{I}|$ which allows more than B messages pending in channel (i, j), i.e. $W_{i,j}(\mathcal{G}) \geqslant B + 1$. For simplicity's sake we shall assume here that there is at most one message from i to j in the MSC of each node.

The formula $\Phi_{\overline{\text{cb}}}(\mathcal{G}, i, j, B)$ uses $H = N \times |\mathcal{I}|$ rows of $B + 3 + |\mathcal{I}| + N$ Boolean variables that describe a path of length at most $N \times |\mathcal{I}|$ starting from any state. In each row x, or equivalently at each stage of this path, the first N variables $N_{a,x}$ (with $1 \leqslant a \leqslant N$) specify which node is reached by the path at each step: At most one of these

variables may be assigned to true. The next $|\mathcal{I}|$ variables $I_{p,x}$ (with $p \in \mathcal{I}$) keep track of the processes stuck at some stage. Finally the next $B+1$ variables $C_{b,x}$ (with $1 \leqslant b \leqslant B+1$) allow to count in unary the number of messages pending in channel (i,j). For convenience and technical reasons we consider also two additional Boolean variables $C_{0,x}$ and $C_{B+2,x}$ which must be assigned to respectively to true and false. Alternatively these variables can be regarded as constants in the subsequent formulae.

The formula $\Phi^{-}_{\text{cb}}(\mathcal{G}, i, j, B)$ consists of four parts:

$$\begin{aligned}\Phi^{-}_{\text{cb}}(\mathcal{G}, i, j, B) \quad = \quad & \mathsf{Valid\_Path} \wedge \mathsf{Stuck\_Processes} \\ & \wedge \mathsf{Counting} \wedge \mathsf{Overflow}\end{aligned}$$

The formula Valid_Path requires that the variables $N_{a,x}$ select a path in the MSG. These conditions are formalized as follows:

$$\bigwedge_{1 \leqslant x \leqslant H, 1 \leqslant a < b \leqslant N} (N_{a,x} \rightarrow \neg N_{b,x})$$

$$\bigwedge_{1 < x \leqslant H, 1 \leqslant a \leqslant N} \left(N_{a,x} \rightarrow \bigvee_{b \in \text{pred}(a)} N_{b,x-1} \right)$$

where $b \in \text{pred}(a)$ holds if there is an edge from node b to node a. To complete Valid_Path we require also that the first node of the path contains a message from i to j: We add the clause $\bigvee_{i \rightarrow_a j} N_{a,1}$ where $i \rightarrow_a j$ holds if there is a message from i to j in the MSC of node a.

Recall that we have assumed for simplicity's sake that there is at most one message from i to j in each node. We focus now on the maximal prefix of the MSC corresponding to the selected path for which the receipt by j of the first message from i does not occur. We derive from the causal dependencies in MSCs a subset of processes stuck after each node, i.e. the processes whose behaviour depends on the receipt of the first message from i to j. Initially, no process is stuck: Thus we require $\neg I_{p,0}$ for each process $p \in \mathcal{I}$. Actually we regard each $I_{p,0}$ as a constant equal to false in the subsequent formulae. On the other hand, process j is stuck after the first node, so we require that $I_{j,1}$ be assigned to true. More precisely, the subset of instances stuck after the first node is characterized by the next condition:

$$\bigwedge_{1 \leqslant a \leqslant N, p \in \text{Init-Blocked}(a)} (N_{a,1} \rightarrow I_{p,1})$$

where for each node a, the subset Init-Blocked$(a) \subseteq \mathcal{I}$ collects all processes which have an event in the MSC of a that depends causally on the receipt of a message from i to j. If a process is stuck at some stage, then it is stuck in the next stages too:

$$\bigwedge_{0 \leqslant x < H, p \in \mathcal{I}} (I_{p,x} \rightarrow I_{p,x+1})$$

If a process depends at some stage of some event occurring on a process stuck at the previous stage, then it gets stuck too. We denote by blocked$(a,p) \subseteq \mathcal{I}$ the subset of processes that depend causally on process p in the MSC of node a. Then we require that

$$\bigwedge_{2 \leqslant x < H, 1 \leqslant a \leqslant N, p \in \mathcal{I}, q \in \text{blocked}(a,p)} ((N_{a,x} \wedge I_{p,x-1}) \rightarrow I_{q,x})$$

These clauses form the formula Stuck_Processes. They require that for each process p that depends on the receipt of the first message from i to j at some stage x (or earlier) the variable $I_{p,x}$ is assigned to true in any satisfiable assignment of $\Phi^{-}_{\text{cb}}(\mathcal{G}, i, j, B)$.

We need to count in unary the number of pending messages from i to j at each stage x using the $B+1$ bits $C_{b,x}$ of each row. First, we make sure that $C_{b,x}$ describes a natural number in unary, initiated by 0, by requiring that the next three clauses

$$\neg C_{b,0} \wedge (C_{b,x} \rightarrow C_{b-1,x}) \wedge (\neg C_{b,x} \rightarrow \neg C_{b+1,x})$$

hold for all $0 \leqslant x \leqslant H$ and all $1 \leqslant b \leqslant B+1$. Here again, each $C_{b,0}$ is actually a constant equal to false. We also require the two clauses $C_{0,x}$ and $\neg C_{B+2,x}$ for each $0 \leqslant x \leqslant H$. Now if the selected path ends up at row x, that is, if no node is selected at row $x+1$, then we want the unary counter of row $y > x$ to be equal to the counter at row x. To do so, we require the two following clauses

$$\left(\left(\bigwedge_{1 \leqslant a \leqslant N} \neg N_{a,x} \right) \wedge \neg C_{b,x-1} \right) \rightarrow \neg C_{b,x}$$

$$\left(\left(\bigwedge_{1 \leqslant a \leqslant N} \neg N_{a,x} \right) \wedge C_{b,x-1} \right) \rightarrow C_{b,x}$$

for all $1 \leqslant x \leqslant H$ and all $1 \leqslant b \leqslant B+1$. Since the counter cannot decrement, we can require:

$$\bigwedge_{1 \leqslant x < H, 1 \leqslant b \leqslant B+1} (C_{b,x} \rightarrow C_{b,x+1})$$

Since each MSC in $\mathcal{G}$ contains at most one message from i to j, the counter may not be incremented twice at some stage:

$$(C_{b,x-1} \wedge \neg C_{b+1,x-1}) \rightarrow \neg C_{b+2,x}$$

for all $1 \leqslant b \leqslant B-1$ and all $1 \leqslant x \leqslant H$. It remains to increment the counter at each stage if needed. Actually we forbid the incrementation of the counter when necessary. We require that

$$(N_{a,x} \wedge \neg C_{b,x-1}) \rightarrow \neg C_{b,x}$$

for all $1 \leqslant b \leqslant B+1$, all $1 \leqslant x \leqslant H$ and all $1 \leqslant a \leqslant N$ such that the node a shows no message from i to j and

$$(N_{a,x} \wedge I_{p,x-1} \wedge \neg C_{b,x-1}) \rightarrow \neg C_{b,x}$$

for all $1 \leqslant a \leqslant N$, all $1 \leqslant b \leqslant B+1$ and all processes p such that the node a shows some message from i to j but the sending of that message depends causally on process p.

Finally the formula should be satisfiable only if the counter reaches $B+1$ after the last stage, which is formalized by the clause $\mathsf{Overflow} = C_{B+1,H}$. Thus, there exists a path in $\mathcal{G}$ of length at most $N \times |\mathcal{I}|$ whose corresponding MSC has a channel-width greater than $B+1$ if and only if $\Phi_{\overline{\text{cb}}}(\mathcal{G}, i, j, B)$ is satisfiable.

THEOREM III.12 *Let $\mathcal{G}$ be a non-divergent MSG, (i,j) be a channel, and B be a buffer size for (i,j). Then $W_{i,j}(\mathcal{G}) > B$ iff the Boolean formula $\Phi_{\overline{cb}}(\mathcal{G}, i, j, B)$ is satisfiable.*

We leave it to the reader to adapt the formalization of the Boolean formula $\Phi_{\overline{\text{cb}}}(\mathcal{G}, i, j, B)$ to the case where several messages from i to j may appear in the MSC of a node. For latter purposes, we stress here that the clause $\mathsf{Overflow} = C_{B+1,H}$ is equivalent to the alternative constraint

$$\mathsf{Overflow\_bis} = \bigwedge_{1 \leqslant b \leqslant B+1} C_{b,H}$$

because we have an unary counter.

D. Computing the optimal buffer size

Let $\mathcal{G}$ be a non-divergent MSG and (i,j) be a fixed channel. Let $m_{i,j} = \max_{v \in V} \#^{i!j}(\mu(v))$ be the maximal number of messages sent from i to j in MSCs in $\mathcal{G}$. In order to check whether some buffer size B is appropriate for (i,j), we have explained above that it is sufficient to check the channel-width of MSCs described by a sequence of nodes of length at most $N \times |\mathcal{I}|$. It follows that $W_{i,j}(\mathcal{G}) \leqslant m_{i,j} \times N \times |\mathcal{I}|$.

This upper-bound can be applied in order to use weighted MAX-SAT solvers to compute $W_{i,j}(\mathcal{G})$. The idea rather simple. Consider the formula $\Phi_{\overline{\text{cb}}}(\mathcal{G}, i, j, B)$ with $B = m_{i,j} \times N \times |\mathcal{I}| + 1$. We need to distinguish two sets of clauses in $\Phi_{\overline{\text{cb}}}(\mathcal{G}, i, j, B)$. The first ones are those appearing in the formula $\mathsf{Overflow\_bis} = \bigwedge_{1 \leqslant b \leqslant B+1} C_{b,H}$ which requires that the number of messages pending in (i,j) after the selected prefix be equal to $B+1$. These clauses are given weight 1. The K other clauses of $\Phi_{\overline{\text{cb}}}(\mathcal{G}, i, j, B)$ are weighted with $B+2$. Let R be the maximal weight of the resulting weighted formula. Since these K clauses may be satisfied (by considering any path), we have $R \geqslant K \times (B+2)$. It follows that $W_{i,j}(\mathcal{G}) = R - K \times (B+2)$. Thus, it is possible to make use of formula $\Phi_{\overline{\text{cb}}}(\mathcal{G}, i, j, B)$ together with a MAX-SAT-solver to compute $W_{i,j}(\mathcal{G})$ for a given channel (i,j).

IV. COMPARISONS TO MSCAN AND OTHER TOOLS

A natural way to evaluate our approach is to draw a comparison between our prototype and some existing tools. We have studied first MSCan [5] which is a recent tool dedicated to the verification of basic properties of MSGs, including global-cooperation. For a fixed number of nodes n, we have considered random MSGs over $n/2$ processes. For each node, we have chosen randomly three successor nodes together with an MSC consisting of a single message exchange. Figure 6 details the average runtime τ_{MSCan} needed by MSCan to check global-cooperation and the average runtimes needed by the SAT-solver `minisat2` to check global-cooperation (τ_{GC}) and non-divergence (τ_{DIV}) on our computer (Intel® Xeon® E5620, 2,4 GHz, 6Go RAM).

Number of nodes	τ_{DIV}	τ_{GC}	τ_{MSCan}	τ_{SOFAT}
7	< 0.01	< 0.01	0.5	1
8	< 0.01	< 0.01	0.6	4
9	< 0.01	< 0.01	0.65	3
10	< 0.01	< 0.01	0.8	14
11	< 0.01	< 0.01	0.94	78
12	< 0.01	< 0.01	1.1	191
13	< 0.01	< 0.01	1.3	841
14	< 0.01	< 0.01	2.5	2700
15	< 0.01	< 0.01	3.5	
16	< 0.01	< 0.01	14	
17	< 0.01	< 0.01	21	
18	< 0.01	< 0.01	58	
19	< 0.01	< 0.01	210	
20	< 0.01	< 0.01	440	
21	< 0.01	< 0.01	550	
22	< 0.01	< 0.01	1800	
23	< 0.01	< 0.01	2700	
100	< 0.01	0.225		
200	0.02	1.5		
400	0.14	9.6		
600	0.46	30		
800	1	71		
1000	2	140		
2000	14	1100		
4000	120			
6000	383			

Figure 6. Average runtime in s. for random MSGs

Window size	τ_{DIV}	τ_{GC}	τ_{SOFAT}	τ_{MSCan}
10	< 0.01	0,01	0,5	0,63
20	< 0.01	0,04	1	1,1
30	< 0.01	0,10	1,8	2,8
40	< 0.01	0,22	5,0	7,1
50	< 0.01	0,35	10	16
60	< 0.01	0,60	21	30
70	< 0.01	0,86	37	50
80	< 0.01	1,1	60	86
90	< 0.01	1,4	98	140
100	< 0.01	1,90	150	210
200	0,01	10,6	2400	
400	0,02	63,4		
600	0,04	253		
800	0,06	470		
1000	0,08	1100		
10000	7,1			
20000	25			
50000	160			

Figure 7. Runtime in s. for the simplified sliding window protocol

In order to look at more intuitive instances of problems, we have next considered the simplified sliding window protocol described in Fig. 3. For all window sizes, this protocol is globally-cooperative and non-divergent. The comparison between MSCan and our prototype is reported in Fig. 7.

To complete the tables from Fig. 6 and Fig. 7, we have

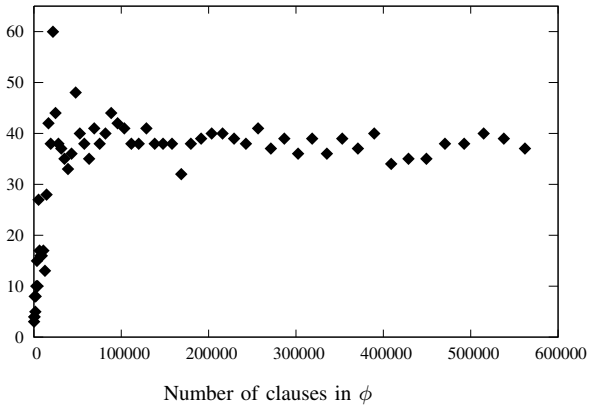

Figure 8. Ratio $\rho(\phi)$ for divergence with up to 100 queens

also compared this approach to another tool for MSCs called SOFAT [10]. However these values are not relevant because this tool checks several properties simultaneously, so we cannot measure the time needed for checking global-cooperation or non-divergence precisely. Further we wanted to compare our prototype to the tool MESA [3], but unfortunately we could not run the available binaries up to now. However our first experiments show that using reductions to SAT and SAT-solvers may be a good choice to check non-divergence and global-cooperation in MSGs.

V. Practical cost of our reductions to SAT

Global-cooperation and non-divergence are very similar properties based on the strongly connected components of the communication graphs of all simple loops. Still Fig. 6 and 7 show that checking divergence is much easier than checking global-cooperation even with only two processes. The main difference between our two reductions is the number of variables used to represents (sets of) simple loops: $\Phi_{\overline{\text{gc}}}(\mathcal{G}, i, j)$ requires $|V|$ times more variables. In this section we compare the cost of these two reductions in practice. To do so we use two simple and similar reductions from SAT to divergence and global-cooperation. These two linear reductions allow us also to define, to measure and to compare the *cost* of our reductions.

A. Linear reduction from SAT to divergence

Let ϕ be a Boolean formula with V variables x_1, ..., x_V and N clauses C_1, ..., C_N. We build an MSG $\mathcal{G}_{\text{div}}(\phi)$ with $V+2$ processes and $1+\sum_{k=1}^{N} d_k$ nodes where the degree d_k is the number of literals in C_k. Each variable is regarded as a process and we consider two new processes denoted by `true` and `false`. Each literal of each clause C_k is identified with a node and we consider an additional node denoted by $\imath$. This initial node is associated with the MSC with a single message exchange from `false` to `true`. On the other hand the label carried by the other nodes l is the MSC that consists of a single message exchange from `true` to the variable x, if $l = x$, and from x to `false` if $l = \text{not}(x)$. Consider now the MSG $\mathcal{G}_{\text{div}}(\phi)$ such that we have

- an edge from each literal of C_k to each literal of C_{k+1} for $1 \leqslant k < N$;
- an edge from $\imath$ to each literal of C_1 and an edge from each literal of C_N to $\imath$.

In particular any loop goes through $\imath$ so its communication graph is connected since it contains an edge from `false` to `true`. Moreover there is a path from `true` to `false` if and only if this graph is strongly connected, which means that some variable x is connected to `true` and `false`. Thus we can derive a satisfying assignment from any simple loop whose communication graph is not strongly connected. Conversely it is easy to derive from any satisfying assignment some simple loop whose communication graph is not strongly connected.

LEMMA V.1 *A Boolean formula ϕ is satisfiable iff the channel* (`false`,`true`) *is divergent in* $\mathcal{G}_{\text{div}}(\phi)$.

B. Cost for divergence

Theorem III.2 shows that checking the divergence of a channel (i, j) in some MSG $\mathcal{G}$ can be reduced simply to the satisfiability of $\Phi_{\text{div}}(\mathcal{G}, i, j)$. Now for a given Boolean formula ϕ we consider the time $\tau_1(\phi)$ needed by a SAT-solver to solve ϕ, and the time $\tau_2(\phi)$ needed by the same SAT-solver to solve the equivalent problem formalized by $\Phi_{\text{div}}(\mathcal{G}_{\text{div}}(\phi), \texttt{false}, \texttt{true})$. The ratio $\rho(\phi) = \tau_2(\phi)/\tau_1(\phi)$ represents the cost of our reduction to SAT to check divergence of $\mathcal{G}_{\text{div}}(\phi)$.

In order to evaluate this ratio, we have considered a benchmark of about 2000 Boolean formulae used for a competition of SAT-solvers in 2002. Then, for practical reasons, we have restricted our study to the problems such that τ_1 is less than 200s. Surprisingly we have observed that the ratio is at most 3 for all problems with less than 700 clauses. For larger problems with up to 150000 clauses, the ratio remained less than 220.

It is easy to encode the n-queens problem into the satisfiability of a Boolean formula. We have evaluated the ratio for up to 100 queens and observed that the ratio tends to the value 40 (see Fig. 8). This is a rather good surprise which remains unexplained. We have observed a similar situation when we added the requirement that queens must be on black squares or when we considered formulae corresponding to random MSGs.

C. Cost for global-cooperation

Consider two new processes denoted by $\top$ and $\bot$. Replace in $\mathcal{G}_{\text{div}}(\phi)$ the MSC carried by the initial node by the MSC with two message exchanges, one from `true` to $\top$, the other from $\bot$ to `false`. We denote by $\mathcal{G}_{\overline{\text{gc}}}(\phi)$ the resulting MSG. Similarly to Lemma V.1, we can establish the next result.

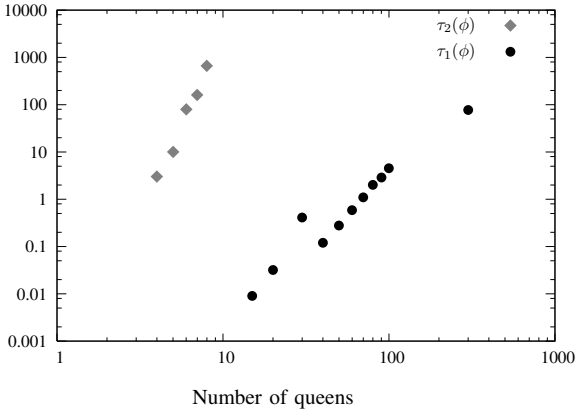

Figure 9. Runtimes τ_1 and τ_2 in s. for global-cooperation

LEMMA V.2 *A Boolean formula ϕ is satisfiable if and only if the MSG $\mathcal{G}_{\overline{gc}}(\phi)$ is* not *globally-cooperative.*

For a Boolean formula ϕ we consider the time $\tau_1(\phi)$ needed by a SAT-solver to solve ϕ, and the time $\tau_2(\phi)$ needed by the same SAT-solver to solve the equivalent problem formalized by $\Phi_{\overline{\text{gc}}}(\mathcal{G}_{\overline{\text{gc}}}(\phi), \texttt{false}, \texttt{true})$. The ratio $\rho(\phi) = \tau_2(\phi)/\tau_1(\phi)$ represents the cost of our reduction to SAT to check global-cooperation of $\mathcal{G}_{\overline{\text{gc}}}(\phi)$.

Similarly to Figure 8, we wanted to measure this ratio for SAT formulae encoding the n-queens problem. However this ratio cannot be measured easily. The reason is that the value of $\tau_2(\phi)$ is simply too large when $\tau_1(\phi)$ is significant, say larger than 10ms. This situation is depicted in Figure 9 which shows the experimental values of $\tau_1(\phi)$ and $\tau_2(\phi)$ for distinct numbers of queens.

VI. CONCLUSION

We have presented some reductions from Divergence and Global-Cooperation to SAT that allow us to take advantage of all the works on SAT-solvers to check MSC specifications. Preliminary experiments show that this approach is efficient in practice compared to specialized tools. However Global-Cooperation seems to be more complicated to check than Divergence at least when using the reduction presented in this paper. Some alternative reductions could be designed and tested in order to reduce this gap. We have formalized also a way to use SAT-solvers and MAX-SAT-solvers to check the buffer size of channels or even to compute the optimal buffer size for a channel. To our knowledge, there is no equivalent method or tool available so far. In order to ease further comparisons, our scripts to reduce MSGs to Boolean formulae and vice-versa are available at http://pageperso.lif.univ-mrs.fr/~florent.avellaneda/MSCChecking.

ACKNOWLEDGMENT

This work is partly supported by project ECSPER (ANR-09-JCJC-0069).

REFERENCES

[1] Rajeev Alur and Mihalis Yannakakis. Model checking of message sequence charts. In Jos C. M. Baeten and Sjouke Mauw, editors, *CONCUR*, volume 1664 of *Lecture Notes in Computer Science*, pages 114–129. Springer, 1999.

[2] Hanêne Ben-Abdallah and Stefan Leue. Syntactic detection of process divergence and non-local choice in message sequence charts. In Ed Brinksma, editor, *TACAS*, volume 1217 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 1997.

[3] Hanêne Ben-Abdallah and Stefan Leue. MESA: Support for scenario-based design of concurrent systems. In Bernhard Steffen, editor, *TACAS*, volume 1384 of *Lecture Notes in Computer Science*, pages 118–135. Springer, 1998.

[4] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In Rance Cleaveland, editor, *TACAS*, volume 1579 of *Lecture Notes in Computer Science*, pages 193–207. Springer, 1999.

[5] Benedikt Bollig, Carsten Kern, Markus Schlütter, and Volker Stolz. MSCan - a tool for analyzing MSC specifications. In Holger Hermanns and Jens Palsberg, editors, *TACAS*, volume 3920 of *Lecture Notes in Computer Science*, pages 455–458. Springer, 2006.

[6] Benedikt Bollig and Martin Leucker. Message-passing automata are expressively equivalent to EMSO logic. *Theoretical Computer Science*, 358(2-3):150–172, 2006.

[7] Blaise Genest, Dietrich Kuske, and Anca Muscholl. A Kleene theorem and model checking algorithms for existentially bounded communicating automata. *Information and Computation*, 204(6):920–956, 2006.

[8] Blaise Genest, Anca Muscholl, Helmut Seidl, and Marc Zeitoun. Infinite-state high-level MSCs: Model-checking and realizability. *Journal of Computer and System Sciences*, 72(4):617–647, 2006.

[9] Elsa L. Gunter, Anca Muscholl, and Doron Peled. Compositional message sequence charts. *International Journal on Software Tools for Technology Transfer*, 5(1):78–89, 2003.

[10] Loïc Hélouët. SOFAT, A Scenario Oracle and Formal Analysis Toolbox. http://www.irisa.fr/distribcom/Prototypes/SOFAT/, 2008. [Online; accessed 13-May-2011].

[11] Jesper G. Henriksen, Madhavan Mukund, K. Narayan Kumar, Milind A. Sohoni, and P. S. Thiagarajan. A theory of regular MSC languages. *Information and Computation*, 202(1):1–38, 2005.

[12] Markus Lohrey and Anca Muscholl. Bounded MSC communication. *Information and Computation*, 189(2):160–181, 2004.

[13] Anca Muscholl and Doron Peled. Message sequence graphs and decision problems on Mazurkiewicz traces. In Miroslaw Kutylowski, Leszek Pacholski, and Tomasz Wierzbicki, editors, *MFCS*, volume 1672 of *Lecture Notes in Computer Science*, pages 81–91. Springer, 1999.