Verification of Programs with Half-Duplex Communication

Gérard Cécé [1]

LIFC, Université de Franche-Comté, 16, route de Gray, 25030 Besançon Cedex, France

Alain Finkel

LSV, CNRS UMR 8643, ENS de Cachan, 61 av. du Pdt. Wilson, 94235 Cachan Cedex, France

Abstract

We consider the analysis of infinite *half-duplex systems* made of finite state machines that communicate over unbounded channels. The half-duplex property for two machines and two channels (one in each direction) says that each reachable configuration has at most one channel non empty.

We prove in this paper that such half-duplex systems have a recognizable reachability set. We show how to compute, in polynomial time, a symbolic representation of this reachability set and how to use that description to solve several verification problems. Furthermore, though the model of communicating finite state machines is Turing-powerful, we prove that membership of the class of half-duplex systems is decidable. Unfortunately,the natural generalization to systems with more than two machines is Turing-powerful. We also prove that the model-checking of those systems against PLTL (Propositional Linear Temporal Logic) or CTL (Computational Tree Logic) is undecidable. Finally, we show how to apply the previous decidability results to the Regular Model Checking. We propose a new symbolic reachability semi-algorithm with acceleration which successfully terminates on half-duplex systems of two machines and some interesting non half-duplex systems.

Key words: communicating finite state machines, half-duplex communication, symbolic verification, channel-recognizable reachability sets, decidability, regular model checking, PLTL model-checking, CTL model-checking

Email addresses: `Gerard.Cece@univ-fcomte.fr` (Gérard Cécé), `Alain.Finkel@lsv.ens-cachan.fr` (Alain Finkel).

[1] a part of this work has been supported by the VIRES – Esprit LTR #23498 project – during a post-

1 Introduction

There are several models which can be used to study distributed algorithms and communicating protocols. Systems of *communicating finite state machines (CFSMs)* consist of several processes represented by finite state machines that communicate over unbounded channels. This model is widely used, both for verification and validation, in tools based on specification languages such as ESTELLE and SDL [Tur93].

Communicating finite state machines CFSMs have the power of Turing machines since it is possible to simulate them by a system of two CFSMs [BZ83,FM97], thus non-trivial problems are undecidable for this model. This limitation motivates the study of classes of systems for which verification problems are possible, such as well-ordered systems [VC83], monogeneous systems [Fin90], linear systems [Fin90,JJ93], systems with recognizable channel properties [Pac87], completely specified protocols [Fin94], systems with non-perfect channels [AJ96b,AJ96a,CFP96]. Apart from the classes for which the membership is structural (e.g. completely specified protocols or lossy channel systems) it is often impossible, in general, to decide whether a given system belongs to these classes (e.g. monogeneous systems, linear systems or systems with recognizable channel properties). Therefore, one can not know when to use the theoretical results about them!

Our contribution We present a new class, *half-duplex systems* of two machines, which have several nice properties. Firstly, one can decide in *polynomial time* whether a given system is half-duplex (for systems of two machines). Secondly, we can also compute in polynomial time an *exact* representation of their reachability set. Once computed, this representation provides enough information to decide, still in polynomial time, whether a given configuration is reachable, whether a specified action is executed at least once, whether the reachability set of a half-duplex system is included into the reachability set of another half-duplex system and allows also to compute the exact size of a given channel.

Those positive results on half-duplex systems of two machines may surprise since one might think that the half-duplex property does not really reduce the power of the model as each machine can transfer data in turn. But this hypothesis is precisely too restrictive to simulate a Turing machine since it disables the channels to store the content of the tape: when an automaton sends messages, since it is not allowed to receive any information from the other automaton, its flow control can only goes through the subpart of its finite state machine made of sending actions. Thus the possible contents of its output channel cannot be more complicated than a regular language. It is this

doctoral position in the Institut Montefiore, Université de Liège, Belgium.

last point we use to provide a recognizable description of the reachability set.

We look for a generalization of these results for systems of any number of machines and channels. However, the *natural generalization* of half-duplex systems of two machines, systems such that each pair of machines linked by two channels has a half-duplex communication, can simulate a Turing machine (as we show, a system of three CFSMs is enough).

On another hand, we may generalize the half-duplex property in remarking that systems in which each reachable configuration has at most one non-empty channel, enjoy the same decidability results than half-duplex systems. However, this class of systems does not seem to be able to model, in practice, something else than half-duplex systems.

Another way to extend the previous results is to see half-duplex systems as a particular class of systems, with two machines, for which a symbolic reachability semi-algorithm, with acceleration, terminates.

Let us recall that the classical symbolic reachability semi-algorithm computes, at each step, $X \cup post(X)$ and it stops when it reaches the fixpoint, i.e. the reachability set. Inspired by the half-duplex results, we propose to design an accelerated version of this classical semi-algorithm in computing at each step, $X \cup post(X) \cup post_{1!}(X) \cup \ldots \cup post_{p!}(X)$, with $post_{c!}(X)$ a new kind of meta-transitions and p the number of channels used in the system. We prove that this semi-algorithm with acceleration, terminates on all half-duplex and quasi-stable systems (systems defined in [CF97]). Moreover, it also stops on the Alternating Bit Protocol, even if it is not a half-duplex system. Our semi-algorithm with acceleration (one may accelerate a large class of regular languages, comprising combination of loops which only send messages) also terminates when applied on simple systems in which one may find two loops of emissions while all the semi-algorithms using the existing symbolic technologies based on QDD [BG96], CQDD [BH97] or SLRE [FPIS03] will not terminate.

This technique to speed up the construction of the reachability set has been also applied for counters systems [BW94,FL02] and it has been implemented in the tools Lash [las], Fast[BFLP03] . This technique has been also used for lossy channels systems [ACABJ04] with the tool Trex [ABS01]. Let us remark that counters systems and lossy channels systems have, in general, non-recursive reachability sets.

We would like to emphasize that half-duplex systems is the unique model of CFSMs (with perfect or lossy fifo channels) for which the reachability set is regular, and it is effectively computable in polynomial time (the reachability set of a lossy channel system is regular but it is not computable [CFP96]) even if the initial set is a regular language.

Although half-duplex systems of two machines have a recognizable reachability set and thus enjoy decidable verification problems, there exist some limits to decidability. There exist undecidable verification properties on half-duplex systems (hence on quasi-stable systems too). Even on rather simple half-duplex systems, we show that PLTL, the Propositional Linear Temporal Logic, and CTL, the Computational Tree Logic, are not decidable. To do this, we simulate a Turing machine by an associated half-duplex system "controlled" by a temporal formulae" in PLTL or in CTL.

Outlines of the paper In section 2, we define the model of communicating finite state machines and some properties we want to verify on them. Then we show how to check those properties on systems for which a recognizable reachability set is available. Section 3 is dedicated to systems with two machines and two channels, one in each direction. For such half-duplex systems, we give a recognizable representation of their reachability set. This symbolic representation leads us to define the *symbolic reachability set* and the *symbolic reachability graph* of a half-duplex system. Then we provide a theorem to decide easily whether a system of two machines is half-duplex. We illustrate all these results on a classe of systems previously defined in the literature [GMY84]. In section 4, we analyse two extensions of these results for systems of more than two machines: unfortunately, the natural extensions of half-duplex communication become Turing-powerfull. Then, we show how to use the previous decidability results in the Regular Model Checking. We propose a new symbolic reachability semi-algorithm, with acceleration, which terminates on half-duplex systems; it also stops on some interesting non hafl-duplex systems (Alternating Bit Protocol, systems with two loops).

Finally, in section 5, we consider the analysis of half-duplex systems against the *propositional linear temporal logic (PLTL)*. We show that this logic is undecidable on half-duplex systems of two machines. Section 6 concludes the paper.

2 Systems of Communicating Finite State Machines (CFSMs)

2.1 Preliminaries and properties

The content of a queue is seen as a word over a given finite alphabet Σ. As usual, we note ε the empty word and Σ^* the set of all finite words over Σ. We also note $|x|$ the length of a word x, and $x.y$ or xy the concatenation of two words x and y.

To introduce our model, let us consider the communicating protocol of Fig. 1.

It involves two machines: a sender and a receiver. Whenever the sender has a message to transmit, it first warns the receiver by sending a starting symbol *start*. Then, it sends the main message over the alphabet of two letters $\{a, b\}$. Once this is done, it advises the receiver by sending an *end* symbol and waits for an acknowledgement. The receiver has a symmetrical behaviour. Formally we have:

Definition 1 *A* Communicating Finite State Machine (CFSM) *is a finite transition system given by a 4-tuple* $M = (Q, q_o, \Sigma, \delta)$ *where:*

- Q *is a finite set of* states*,*
- $q_o \in Q$ *is a* initial state*,*
- Σ *is a finite* alphabet *of messages, and*
- $\delta \subseteq Q \times (\mathbb{N} \times \{!, ?\} \times \Sigma) \times Q$ *is a finite set of* transitions*. We also consider* δ *as an application from* $Q \times (\mathbb{N} \times \{!, ?\} \times \Sigma)$ *to* 2^Q *which we extend, in the classical way, as an application from* $Q \times (\mathbb{N} \times \{!, ?\} \times \Sigma)^*$ *to* 2^Q *with* $(\mathbb{N} \times \{!, ?\} \times \Sigma)^*$ *the set of all words on the alphabet* $(\mathbb{N} \times \{!, ?\} \times \Sigma)$*.*

Note that the alphabet of M, in the usual transition system sense, is $(\mathbb{N} \times \{!, ?\} \times \Sigma)$ rather than Σ, i.e. M sees $j!a$ and $j?a$ as single symbols, which we henceforth write as $!a$ and $?a$ respectively when there is no ambiguity about the identity of the channel. Intuitively, $j!a$ denotes the emission of a in channel j, and $j?a$ denotes the reception of a from channel j. A state $q \in Q$ whose all outgoing edges are labelled with sending (resp. receiving) actions is called a *sending (resp. receiving) state.* If q is neither a sending state nor a receiving state then q is called a *mixed state.* We will need to consider transitions of δ which consist only of sending actions (resp. receiving actions). So we note $\delta^! = \{(q, j!a, q') \in \delta\}$ and $\delta^? = \{(q, j?a, q') \in \delta\}$.

Example 2 *According to the* Sender *of Fig. 1, we have:* $Q = \{1, 2, 3\}$, $q_o = 1$, $\Sigma = \{start, a, b, end, ack\}$, $\delta = \Big\{(1, 1!start, 2),\ (2, 1!a, 2),\ (2, 1!b, 2),\ (2, 1!end, 3),\ (3, 2?ack, 1)\Big\}$.

Now, we group communicating machines together in order to deal with *communicating systems.* Formally, we have:

Definition 3 *A* (communicating) system S *is a tuple* $S = (M_1, \ldots, M_n)$ *of CFSMs* $M_i = (Q_i, q_{o_i}, \Sigma, \delta_i)$*.*

In the remainder of the paper, S refers to a system $S = (M_1, \ldots, M_n)$ of CFSMs such that $M_i = (Q_i, q_{o_i}, \Sigma, \delta_i)$, and $\delta = \cup_{i=1,\ldots,n}\delta_i$ refers to the set of all transitions of the system. Let p be the number of channels used in S, we may rename the channels such that they belong to $\{1, \ldots, p\}$. Then, a *configuration* of S is a $(n + p)-$tuple $s = (\vec{q}; \vec{x})$ with $\vec{q} = (q_1, \ldots, q_n)$, $q_i \in Q_i$ and $\vec{x} = (x_1, \ldots, x_p)$, $x_i \in \Sigma^*$. An element $\vec{q} \in Q_1 \times \ldots \times Q_n$ is a *control*

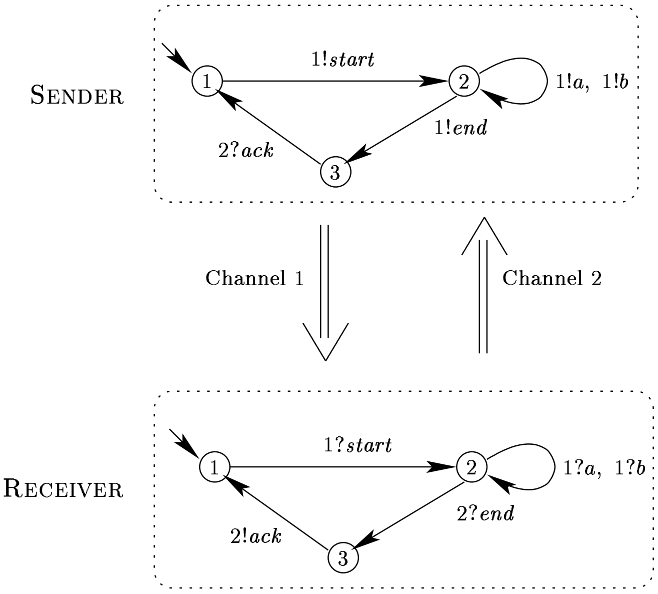


Fig. 1. The protocol S_1

state and an element $q \in Q_i$ for $i \in \{1, \ldots, n\}$ is a *local (control) state* (of machine i). The *state space* of S is the set of all configurations, that is to say: $Q_1 \times \ldots \times Q_n \times (\Sigma^*)^p$.

Remark 4 *Only the communicating actions of the system are taken into account.*

The operational semantic associated with a communicating system is defined by the firing of a transition which changes the current configuration in one step.

Definition 5 *Let S be a communicating system. A configuration $s' = (q'_1, \ldots, q'_n; x'_1, \ldots, x'_p)$ is* reachable *from another configuration $s = (q_1, \ldots, q_n; x_1, \ldots, x_p)$* by the firing of the transition t, *written $s \to s'$, or redundantly $s \xrightarrow{t} s'$, if one of the following two cases holds:*

(1) there exist $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, p\}$ and $a \in \Sigma$ such that $t = (q_i, j!a, q'_i) \in \delta_i$ and
   (a) $q'_k = q_k$ for all $k \neq i$
   (b) $x'_j = x_j.a$ and $x'_l = x_l$ for all $l \neq j$

(2) there exist $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, p\}$ and $a \in \Sigma$ such that $t = (q_i, j?a, q'_i) \in \delta_i$ and
   (a) $q'_k = q_k$ for all $k \neq i$
   (b) $a.x'_j = x_j$ and $x'_l = x_l$ for all $l \neq j$

Condition (1) above describes the output of message a by machine M_i in channel j. Condition (1).(a) says that local states of other machines are not affected by the transition. Condition (1).(b) updates the content of channel

j whereas the others channels stay unchanged. Condition (2) describes the reception of message a by machine M_i from channel j, which is mainly shown by the removal of message a from this channel.

As usual, we extend relation $\rightarrow$ to its reflexive and transitive closure $\xrightarrow{*}$. Furthermore, we note $s_1 \xrightarrow{t_1 t_2 \ldots t_m} s_{m+1}$ whenever there exist $s_2, \ldots, s_m$ such that $s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \ldots \xrightarrow{t_m} s_{m+1}$. The *initial configuration* of the system is $s_o = (\vec{q_o}; \vec{\varepsilon})$ with $\vec{q_o} = (q_{o_1}, \ldots, q_{o_n})$ and $\vec{\varepsilon} = (\varepsilon, \ldots, \varepsilon)$.

Example 6 *For system S_1 of Fig. 1, we have: $s_o \xrightarrow{(1,1!start,2)} (2,1; start, \varepsilon) \xrightarrow{(2,1!a,2)} (2,1; (start)a, \varepsilon)$. As there is no ambiguity, we could have written: $s_o \xrightarrow{!start} (2,1; start, \varepsilon) \xrightarrow{!a} (2,1; (start)a, \varepsilon)$.*

A configuration s is said *reachable* if $s_o \xrightarrow{*} s$. The *reachability set* of S is the set $RS(S)$ of all reachable configurations:

$$RS(S) = \left\{ s \,\middle|\, s_o \xrightarrow{*} s \right\}$$

Example 7

$$RS(S_1) = \left\{ \begin{array}{l} (1,1;\varepsilon,\varepsilon),\ \big(2,1;(start)\{a,b\}^*,\varepsilon\big),\ \big(3,1;(start)\{a,b\}^*(end),\varepsilon\big), \\ \big(2,2;\{a,b\}^*,\varepsilon\big), \big(3,2;\{a,b\}^*(end),\varepsilon\big),\ (3,3;\varepsilon,\varepsilon),\ \big(3,1;\varepsilon,(ack)\big) \end{array} \right\}$$

Since S can be viewed as a transition system, we also define its *reachability tree* and its *reachability graph*:

- the *reachability tree* of S is the labelled tree $RT(S)$ which root is labelled s_o and such that a node labelled s has a child labelled s' and the arc (s, s') is labelled with transition t if and only if $s \xrightarrow{t} s'$.
- the *reachability graph* of S is the labelled graph $RG(S)$ whose set of nodes is $RS(S)$ and whose nodes are labelled with their corresponding configurations in $RS(S)$. A node labelled s has a successor labelled s' and the arc (s, s') is labelled with transition t if and only if $s \xrightarrow{t} s'$.

The term half-duplex is commonly used to characterize a channel, between two machines, which can transmit messages in both directions but not simultaneously. The direction of the transmission can be set for a fixed amount of time (often 200msec) and then be switched [Hol91]. As a consequence, the reachability graph of such a system is finite since channels are bounded in function of the duration of each sending period and of the transmission's rate. In this paper, we use the term half-duplex with a less restrictive sense, without any notion of elapsed time.

Definition 8 *A system $S = (M_1, M_2)$ of two machines with two channels* *(one in each direction) is said* half-duplex *if each reachable configuration has at most one channel non empty.*

Example 9 *The communicating system S_1 of Fig. 1 is half-duplex.*

2.2 Properties and decidability results for systems of CFSMs

We present in this subsection some configurations and problems classically involved in verification of communicating systems, how difficult verification is and what provides a recognizable description of the reachability set. Remember that the notion of recognizable set is the natural extension (see [Ber79]), for sets of tuple of words, of the classical notion of recognizable (or regular) language.

Definition 10 *[Ber79] Let M be a monoid. A subset X of M is recognizable if there exist a finite monoid N, a morphism φ from M into N and a subset P of N such that $X = \varphi^{-1}(P)$.*

The monoid which is considered for systems of CFSM using p channels is $((\Sigma^*)^p, .)$, with . the concatenation such that $(x_1, \ldots, x_p).(y_1, \ldots, y_p) = (x_1y_1, \ldots, x_py_p)$ and with neutral element: $\vec{\varepsilon} = (\varepsilon, \ldots, \varepsilon)$.

Instead of using morphisms and monoids to manipulate recognizable sets, we use products of classical regular languages as stated in the following theorem.

Theorem 11 *(Mezei, see [Ber79]) A subset of $(\Sigma^*)^p$ is recognizable if and only if it is a finite union of sets of the form $X_1 \times \ldots \times X_p$ with X_i recognizable sets of Σ^*.*

Obviously, a subset $X \subseteq Q_1 \times \ldots \times Q_n \times (\Sigma^*)^p$ of the state space of a system $S = (M_1, \ldots, M_n)$ using p channels can be partitioned as follows:

$$X = \bigcup_{\vec{q} \in Q_1 \times \ldots \times Q_n} \{\vec{q}\} \times X_{\vec{q}}$$

Therefore, we say that X is *channel-recognizable* if all the $X_{\vec{q}}$ are recognizable sets of $(\Sigma^*)^p$.

Definition 12 *Let S be a communicating system, t be one of its transitions and $s = (\vec{q}; \vec{x})$, with $\vec{q} = (q_1, \ldots, q_n)$ and $\vec{x} = (x_1, \ldots, x_p)$, be one of its configurations.*

- *s is a* stable configuration *if all its channels are empty, i.e. $\vec{x} = \vec{\varepsilon}$.*

- *s is a* deadlock configuration *if $\vec{x} = \vec{\varepsilon}$ and each q_i is a receiving state. This implies that no transition is fireable from s.*
- *s is an* unspecified reception configuration *if there exists $i \in \{1, \ldots, n\}$ such that: q_i is a receiving state and $\delta(q_i, j?a) \neq \emptyset$ implies that $|x_j| > 1$ and $x_j \notin a\Sigma^*$.*

  (an unspecified reception configuration corresponds to a configuration in which a machine is definitively blocked by its inability to receive the head messages present in its input channels)
- *t is* executable *if it is fireable from a reachable configuration.*

Definition 13 (Problems of interest). *Let S be a communicating system,*

(1) the Reachability Problem *is to determine whether a given configuration of S is reachable.*
(2) the Deadlock Problem *is to determine whether there exists a reachable deadlock configuration.*
(3) the Unspecified Reception Problem *is to determine whether there exists an unspecified reception configuration.*
(4) the Executable Transition Problem *is to determine whether a transition of S is executable.*
(5) the Weak Boundedness Problem *is to determine whether the reachability set $RS(N)$ is finite.*
(6) the Strong Boundedness Problem *is to determine whether the set of all reachable contents of a given channel is finite.*
(7) the Recognizable Computation Problem *is to determine whether an effective description of $RS(S)$ is computable when it is known to be channel-recognizable.*

Example 14 *Let us consider S_1 in Fig. 1. All the seven problems stated in the previous definition are decidable on this system; in particular, channel 1 is unbounded but channel 2 is bounded by one.*

The following theorem states how difficult verification of communicating systems is.

Theorem 15 ([BZ83], [FM97]) *Systems of CFSMs have the power of Turing's machines. Hence, verification is, in general, undecidable on them.*

In [Pac82,Pac87], Pachl showed that problems (1) to (4) of Definition 13 are decidable under the sole assumption that the reachability set is channel-recognizable (even if no description of this reachability set is available!). But this result is not useful in practice since the decision procedure goes through an enumeration of all possible channel-recognizable sets! Furthermore, *knowing that a reachability set is channel-recognizable is not sufficient* to give a description of that set or even to decide the boundedness problem. One example of this point is given by *lossy channel systems* [CFP96]. On the other hand, having an effective channel-recognizable description of the reachability set makes all the problems of Definition 13 easy to solve.

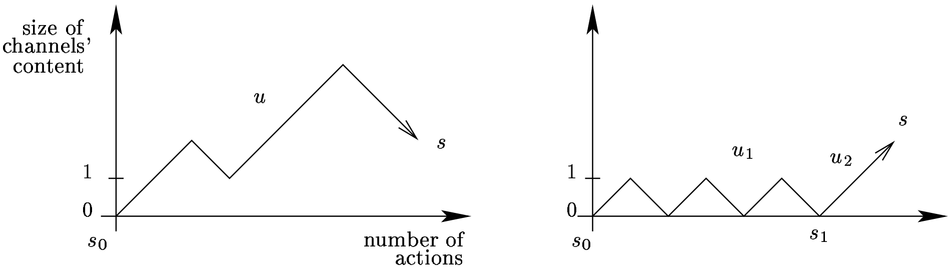
Theorem 16 *For a communicating system S whose reachability set is channel-recognizable and for which we have an effective description, the first six problems of interest are decidable.*

PROOF.

- Points 1,2,5 and 6 are self-evident. The reachability problem is reduced to the membership problem for a recognizable set. The deadlock problem is just a restriction of the reachability problem. The weak boundedness problem is reduced to finiteness of a recognizable set. The strong boundedness problem is reduced to finiteness of a projection from a recognizable set.
- Let us prove decidability of the unspecified reception problem (point 3). Consider the set $R'(S) = \Big\{(\vec{q}; x_1, \ldots, x_p) |\ (\vec{q}; x'_1, \ldots, x'_p) \in RS(S)$ with $x_i = a$ if $x'_i \in a\Sigma^*$ for a given $a \in \Sigma$, else $x_i = \varepsilon\Big\}$. Since we have a channel-recognizable description of $RS(S)$ then the finite set $R'(S)$ is computable. It is straightforward that $RS(S)$ has an unspecified reception configuration if and only if $R'(S)$ has one, which can be checked. So the unspecified reception problem is solved.
- Let us now consider the executable transition problem (point 4) for a given transition t. There are two sub-cases:
  $t = (q_i, j!a, q'_i) \in \delta_i$ is a sending transition. Then, t is executable whenever there exists a configuration like $(q_1, \ldots, q_i, \ldots, q_n; \vec{x})$ in $RS(S)$.
  $t = (q_i, j?a, q'_i) \in \delta_i$ is a receiving transition. Then, t is executable whenever there exists a configuration like $(q_1, \ldots, q_i, \ldots, q_n; x_1, \ldots, ax'_j, \ldots, x_p)$ in $RS(S)$.
  Both of these sub-cases are solvable since we have a channel-recognizable description of $RS(S)$. □

3 Half-duplex Systems of two Machines

In this section, we restrict ourselves to systems $S = (M_1, M_2)$ of two machines and two channels, the first one from M_1 to M_2 and the second one from M_2 to M_1.



(2.1) (2.2)

Fig. 2. Transformation of executions

3.1 A Symbolic Reachability Graph

First of all, we show that each reachable configuration of half-duplex systems of two machines is reached by a particular execution that we use to compute a channel-recognizable description of the reachability set.

Example 17 *Let us consider the communicating system S_1 in Fig. 1 which is half-duplex (refer to its reachability set in Example 2.3). The sequence of actions $u = \mathbf{1!}\boldsymbol{start}.\mathbf{1!}\boldsymbol{a}.1?start.\mathbf{1!}\boldsymbol{b}.\mathbf{1!}\boldsymbol{b}.\mathbf{1!}\boldsymbol{a}.1?a.1?b$ leads to the configuration $s = (2, 2; ba, \varepsilon)$. A representation of u is given in Fig. 2.1. Now, as shown in Fig. 2.2, u can be reordered in an execution split in two parts, $u_1 = \mathbf{1!}\boldsymbol{start}.1?start.\mathbf{1!}\boldsymbol{a}.1?a\ \mathbf{1!}\boldsymbol{b}.1?b$ and $u_2 = \mathbf{1!}\boldsymbol{b}.\mathbf{1!}\boldsymbol{a}$, such that u_1 is "bounded" (refer to next definition) by 1 and leads to a stable configuration s_1, and such that u_2 is exclusively made of sending actions. We shall show that this transformation is always available for half-duplex systems of two machines. Hence, we can compute the reachability set in two steps. The first one computes the finite set of all stable configurations (like s_1) reached by a "1-bounded" execution. Then, from each of those configurations, the second step computes all configurations reached by exclusively executing sending actions on one of the two channels. Because this last step uses only one machine, we obtain a channel-recognizable set. What follows formalizes and proves this informal description.*

First, we need a more precise description of executions and configurations reached by "bounded executions".

Definition 18 *Let S be a communicating system.*

- *an* execution *from a state s is a sequence of transitions $t_1 \dots t_m$ such that there exists a state s' which satisfies $s \xrightarrow{t_1 \dots t_m} s'$. From an execution u we note $u|_i$ the projection on transitions of machine i. Thus, $u|_i$ is the local execution induced by u on machine i.*
- *an execution $u = t_1 \dots t_m$ from a given state s_1 of S such that $s_i \xrightarrow{t_i} s_{i+1}$ is said* k-bounded *if both channels of all intermediate configurations s_i do not*

contain more than k messages.
- *an execution $u = t_1 \dots t_m$ from a given state s_1 of S such that $s_i \xrightarrow{t_i} s_{i+1}$ is said* half-duplex *if at most one channel of every intermediate configurations s_i is not empty.*
- *the* k-reachability set *of S is the largest subset $RS_k(S)$ of $RS(S)$ within which, each configuration s is reached by a k-bounded execution from s_o.*

Remark 19 *Given a communicating system S, for every integer $k \in \mathbb{N}$, the set $RS_k(S)$ is finite and computable.*

From an execution, we need to extract particular informations so we define:

- the morphism $\ell : \delta^* \to \Big(\mathbb{N} \times \{!, ?\} \times \Sigma\Big)^*$ such that $\ell\Big((q, z, q')\Big) = z$. From an execution, ℓ extracts the corresponding sequence of actions.
- for each j, the morphism $\mathrm{proj}_j : \delta^* \to \Sigma^*$ such that $\mathrm{proj}_j\Big((q, j!a, q')\Big) = \mathrm{proj}_j\Big((q, j?a, q')\Big) = a$ and for $k \neq j$, $\mathrm{proj}_j\Big((q, k!a, q')\Big) = \mathrm{proj}_j\Big((q, k?a, q')\Big) = \varepsilon$. From an execution, proj_j extracts the sequence of messages involving channel j.
- the isomorphism $\Phi : \Big(\mathbb{N} \times \{!, ?\} \times \Sigma\Big)^* \to \Big(\mathbb{N} \times \{!, ?\} \times \Sigma\Big)^*$ such that $\Phi(j!a) = j?a$ and $\Phi(j?a) = j!a$. Note that $\Phi^{-1} = \Phi$. The isomorphism Φ transforms a sending action in the corresponding receiving one and vice versa.

Lemma 20 *Let $S = (M_1, M_2)$ be a system. For every configuration $s = (q_1, q_2; x_1, x_2)$ of S reachable by a half-duplex execution, there exist $u_1 u_2$ an execution from s_o and a configuration s_1 such that the three following conditions hold:*

(1) $s_o \xrightarrow{u_1} s_1 \xrightarrow{u_2} s$,
(2) the execution u_1 from s_o is 1-bounded, s_1 is a stable configuration, and
(3) $\exists i \in \{1, 2\}$ such that $u_2 \in (\delta_i^!)^*$, $x_i = \mathrm{proj}_i(u_2)$ and $x_{3-i} = \varepsilon$.

PROOF. By induction on the length of a half-duplex execution u ending at s. The hypothesis is verified for the base case when $|u| = 0$. For the induction case, let us assume that the hypothesis is true for $|u| \leq n$ and consider the case where $|u| = n + 1$. Let $u' \in \delta^*$ and $t \in \delta$ be such that $u = u't$. Since the execution u from s_o is half-duplex, the execution u' from s_o is also half-duplex. Therefore, there exists, by hypothesis, an execution $u'_1 u'_2$ from s_o which satisfies:

(1) $s_o \xrightarrow{u'_1} s_1 \xrightarrow{u'_2} s' \xrightarrow{t} s$ with $s' = (q'_1, q'_2; x'_1, x'_2)$,
(2) the execution u'_1 from s_o is 1-bounded, s_1 is a stable configuration, and,
(3) $\exists i \in \{1, 2\}$ such that $u'_2 \in (\delta_i^!)^*$, $x'_i = \mathrm{proj}_i(u'_2)$ and $x'_{3-i} = \varepsilon$.

There are two cases depending on whether t is a receiving or a sending action.

$t = (q'_l, l!a, q_l) \in \delta_l$ is a sending action on channel $l \in \{1, 2\}$. If $u'_2 = \varepsilon$ then $x'_1 = x'_2 = \varepsilon$ and we can take $u_1 = u'_1$ and $u_2 = t$. Otherwise, $|x'_i| \geq 1$ and $x'_{3-i} = \varepsilon$ for a $i \in \{1, 2\}$. If $l \neq i$ then $x_l \neq \varepsilon$ but $x_{3-l} = x'_i \neq \varepsilon$, which implies that both channels of s are not empty, and contradicts the half-duplex property of the execution u from s_o. Therefore, $l = i$ and we can choose $u_1 = u'_1$ and $u_2 = u'_2 t$.

$t = (q'_{3-l}, l?a, q_{3-l}) \in \delta_{3-l}$ is a receiving action on channel $l \in \{1, 2\}$. Since $x'_{3-i} = \varepsilon$, we have $l = i$ and $|x'_i| > 1$. Then there exist $t' \in \delta^!_i$ and $u_3 \in (\delta^!_i)^*$ such that $u'_2 = t'u_3$. The transition t' matches with the transition t (i.e. $\Phi(\ell(t)) = \ell(t'))$ because s_1 is a stable configuration and t is the first reception after the emission t'. Therefore, since t and t' are not performed by the same machine, it is possible from s_1 to execute $t'tu_3$ which leads to s. Thus we can choose $u_1 = u'_1 t't$ and $u_2 = u_3$. □

Corollary 21 *Let $S = (M_1, M_2)$ be a half-duplex system. For every reachable configuration $s = (q_1, q_2; x_1, x_2)$ of S, there exist an execution u_1u_2 from s_o and a configuration s_1 such that the three following conditions hold:*

(1) $s_o \xrightarrow{u_1} s_1 \xrightarrow{u_2} s$,
(2) the execution u_1 from s_o is 1-bounded, s_1 is a stable configuration, and
(3) $\exists i \in \{1, 2\}$ such that $u_2 \in (\delta^!_i)^*$, $x_i = \mathrm{proj}_i(u_2)$ and $x_{3-i} = \varepsilon$.

PROOF. Since S is half-duplex, all of its executions are half-duplex. □

Definition 22 *From a given system, $S = (M_1, M_2)$, we distinguish the following subset of configurations:*

$$H(S) = \bigcup_{\substack{(q'_1, q'_2; \varepsilon, \varepsilon) \in RS_1(S),\\ (q_1, q_2) \in Q_1 \times Q_2}} \{q_1\} \times \{q_2\} \times L_1(q'_1, q_1) \times L_2(q'_2, q_2)$$

where $L_i(q'_i, q_i)$ *is the recognizable language accepted by the finite state automaton $(Q_i, q'_i, \Sigma, \{q_i\}, \delta'_i)$ with q'_i the initial state, q_i the single final state and $\delta'_i = \{(q, a, q') \,\big|\, (q, i!a, q') \in \delta_i\}$ the transition relation.*

Remark 23 *$H(S)$ is channel-recognizable since its channel part is a finite union of products of recognizable languages. Furthermore, the description of $H(S)$ is effective since each of the different $L_i(q'_i, q_i)$ are.*

Lemma 24 *The reachability set $RS(S)$ of a half-duplex system S is equal to $H(S)$.*

PROOF.

$H(S) \subseteq RS(S)$. Let $s = (q_1, q_2; x_1, x_2) \in H(S)$ then there exist $s' = (q'_1, q'_2; \varepsilon, \varepsilon) \in RS_1(S)$ and two paths u_1 and u_2 respectively in M_1 and M_2, composed exclusively of sending actions such that $q_1 \in \delta_1^*(q'_1, u_1)$, $q_2 \in \delta_2^*(q'_2, u_2)$, $x_1 = \mathrm{proj}_1(u_1)$ and $x_2 = \mathrm{proj}_2(u_2)$. Thus, from s' we can perform the following execution: $s' = (q'_1, q'_2; \varepsilon, \varepsilon) \xrightarrow{u_1} (q_1, q'_2; \mathrm{proj}_1(u_1), \varepsilon) \xrightarrow{u_2} (q_1, q_2; \mathrm{proj}_1(u_1), \mathrm{proj}_2(u_2)) = s$. Then we have $s_o \xrightarrow{*} s' \xrightarrow{*} s$. Therefore: $s \in RS(S)$.

$RS(S) \subseteq H(S)$. Straightforward from Corollary 21. □

Example 25 *Let us consider the communicating system S_1 in Fig. 1, the reader can verify that $RS(S_1) = H(S_1)$. In particular, the subset $(2, 2; \{a, b\}^*, \varepsilon)$, in Example 2, is obtained by taking $(q'_1, q'_2) = (q_1, q_2) = (2, 2)$ in Definition 22.*

From this lemma, we deduce the main result of this section.

Theorem 26 *The reachability set of a half-duplex system is* channel recognizable and computable *in time: $O\Big(|\Sigma^3 \times Q_1 \times Q_2|\big(|Q_1| + |Q_2|\big)\Big)$.*

PROOF. Recognizability and computability of $RS(S)$ are proved by the preceding Remark and Lemma. Let us now consider the complexity issue. From Definition 22, we only need to compute stable configurations of $RS_1(S)$ since from these configurations a simple run through sending actions of one of the machines gives the other reachable configurations. Computation of $RS_1(S)$ is done by a simple reachability search from the starting configuration s_o. Knowing that $|RS_1(S)| = |Q_1 \times Q_2 \times \Sigma^2|$ and that a reachable configuration has at most $|\Sigma \times Q_1| + |\Sigma \times Q_2|$ successors, we deduce the complexity. □

Corollary 27 *For half-duplex systems of two machines, the seven problems of interest are decidable.*

PROOF. Straightforward from Theorems 26 and 16. □

The effective channel-recognizable description of the reachability set of a half-duplex system of two machines brings us to the following definition.

Definition 28 *Let S be a system.*

- *the* symbolic reachability set *of S, $\mathtt{SRS}(S)$, is a set whose elements have two parts. The first one is a control state and the second one the restriction on the channels of the description of the reachability set for this control state.*
- *the* symbolic reachability graph *of S, $\mathtt{SRG}(S)$, is a graph whose nodes are labelled with elements of $\mathtt{SRS}(S)$ and such that there is an edge labelled with*

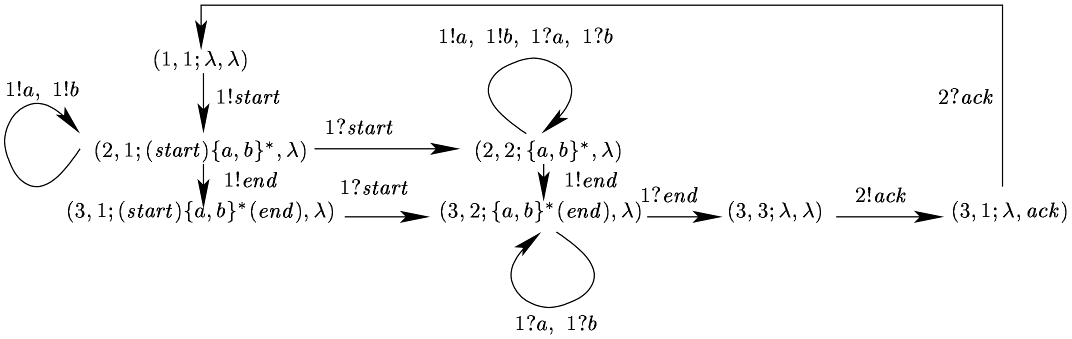


Fig. 3. The symbolic reachability graph of S_1

$t \in \delta$ from $(\vec{q}; \vec{X})$ to $(\vec{q}\,'; \vec{X}\,')$ if and only if every successor, by the execution of transition t, of elements of $(\vec{q}, \vec{X})$ are in $(\vec{q}\,', \vec{X}')$.

The symbolic reachability graph of a system is closely related to MCG the *minimal coverability graph* of a Petri net [Fin93], in the sense that each execution of a system S is a path in SRG(S) and each reachable configuration is *covered* by (i.e. is an element of) a node of this graph. However, the SRG contains more information since its nodes describe exactly the reachability set, which is not the case for the MCG of a Petri net.

Example 29 *The symbolic reachability graph of system S_1 is given in Fig. 3.*

3.2 Decidability of the half-duplex property

In the previous subsection, we gave a technique to describe the reachability set of half-duplex systems. But to be useful, we need to know on which systems these results can be applied. In other words, we have to decide whether a system S of two machines is half-duplex.

Verifying whether a communicating system is half-duplex requires to look at all reachable configurations (cf. Definition 8). But, a communicating system may have an infinite, and potentially non recursive, reachability set. So we have to reduce the domain of interest to a finite one. Then, we will be able to verify the half duplex property.

Lemma 30 *A system $S = (M_1, M_2)$ is not half-duplex if and only if there exist two transitions $(q_1, 1!a, q_1') \in \delta_1$ and $(q_2, 2!b, q_2') \in \delta_2$ such that the state $s = (q_1, q_2; \varepsilon, \varepsilon)$ belongs to $RS_1(S)$.*

PROOF. Assume S is not half-duplex, there exists a reachable configuration whose both channels are not empty. Let u be one of the smallest executions from s_o leading to such a reachable configuration $s_3 = (q_{1,3}, q_{2,3}; x_{1,3}, x_{2,3})$ with

$x_{1,3} \neq \varepsilon$ and $x_{2,3} \neq \varepsilon$. Obviously we have $|u| \geq 1$, then there exist $s_2 \in RS(S)$, $u' \in \delta^*$ and $t \in \delta$ such that $s_o \xrightarrow{u'} s_2 \xrightarrow{t} s_3$ with $u = u't$. Since the execution u from s_o is one of the smallest executions leading to a configuration both of whose channels are not empty then each intermediate configuration of u has at least one channel empty. In other words, u' is a half-duplex execution from s_o. Therefore, there exists, Lemma 20, an execution $s_o \xrightarrow{u_1} s_1 \xrightarrow{u_2} s_2 \xrightarrow{t} s_3$, with $s_k = (q_{1,k}, q_{2,k}; x_{1,k}, x_{2,k})$, such that:

- the execution $s_o \xrightarrow{u_1} s_1$ is 1-bounded and s_1 is a stable configuration.
- $\exists i \in \{1, 2\}$ such that $u_2 \in (\delta_i^!)^*$, $x_{i,2} = \mathrm{proj}_i(u_2)$ and $x_{j,2} = \varepsilon$ with $j = 3 - i$.

Since both channels of s_3 are not empty while channel j of s_2 is empty then $|u_2| \geq 1, t \in \delta_j^!$ and $q_{j,1} = q_{j,2}$. Therefore there exists a sending transition t' from $q_{i,1}$, and the sending transition t starts from $q_{j,1}$. Thus we can take $s = s_1$.

To summarize, if a system is not half-duplex, then one of its 1-bounded executions leads to a stable configuration from which each machine can execute a sending action. Furthermore, this condition is obviously sufficient for the system not to be half-duplex, since those transitions can be fired in sequence from this reachable configuration. □

Theorem 31 *The half-duplex property is decidable in time :*

$$O(|\Sigma^3 \times Q_1 \times Q_2|(|Q_1| + |Q_2|)) \ .$$

PROOF. Since $RS_1(S)$ is finite and computable, the property of Lemma 30 is checkable. The complexity calculation follows from the complexity of computing $RS_1(S)$ as discussed in Proof of Theorem 26. □

3.3 Connection of these results with Gouda et al.

In [GMY84], Gouda et al. searched for a simple class of communicating systems free of deadlock configurations and of unspecified reception configurations. They proposed the definition of systems made of two deterministic machines, with no mixed state and such that their communication is *compatible* (cf. next definition). They solved the boundedness problem for this class. A similar work has been done by Mountassir but for two symmetrical machines [Mou89] instead of two machines with a compatible communication. The class defined in [Mou89] appears to be a subclass of this defined in [GMY84]. We claim that these two classes are included in the half-duplex one of two machines; thus, all the problems of interest (Definition 13) are decidable. In particular, we can

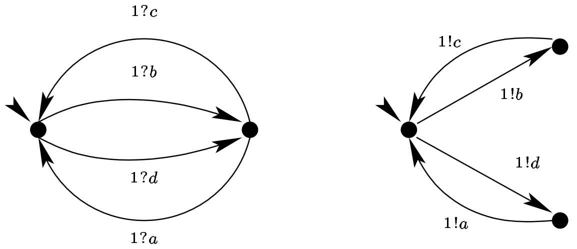


Fig. 4. A non-compatible half-duplex system

compute a channel-recognizable description of the reachability set and thus solve more problems than just the boundedness one.

First of all, let us recall the definition.

Definition 32 ([GMY84]) *A system $S = (M_1, M_2)$ is* compatible *if $L(M_1) = \Phi(L(M_2))$ with $L(M_i)$ the language of all sequences of actions that M_i can execute (i.e. the recognizable language accepted by M_i considered as a classical finite state automaton whose all states are final).*

As an example, S_1 in Fig. 1 belongs to the classes of [GMY84] and [Mou89]. However, as shown in Fig. 4, there exist half-duplex systems which are not compatible.

To prove the next theorem we need the two following lemma inspired from a part of the proof of [GMY84, Lemma 2].

Lemma 33 *Let $S = (M_1, M_2)$ be a system, w_1 be a local execution of M_1 ending at states $q_1 \in Q_1$, w_2 be a local execution of M_2 ending at states $q_2 \in Q_2$ such that $\ell(w_1) = \Phi(\ell(w_2))$. Then, there exists an execution u from s_o such that $u|_1 = w_1$, $u|_2 = w_2$ and $s_o \xrightarrow{u} s = (q_1, q_2; \varepsilon, \varepsilon)$.*

PROOF. For a sequence of transitions $v = t_1 \ldots t_n$, we note $v(i)$, with $1 \leq i \leq n$, the transition t_i. Since $\ell(w_1) = \Phi(\ell(w_2))$ then if $w_1(i)$ is a sending transition, then $w_2(i)$ is a receiving transition such that $\ell(w_1(i)) = \Phi(\ell(w_2(i)))$; and vice versa.

Therefore, we take:

$$u(2 * i) = \begin{cases} w_1(i) & \text{if } w_1(i) \text{ is a sending transition} \\ w_2(i) & \text{otherwise } (w_2(i) \text{ is a sending transition}) \end{cases}$$

$$u(2 * i + 1) = \begin{cases} w_1(i) & \text{if } w_1(i) \text{ is a receiving transition} \\ w_2(i) & \text{otherwise } (w_2(i) \text{ is a receiving transition}) \end{cases}$$

For each sending transition of w_1 corresponds a receiving transition of w_2; and vice versa. Then all sent messages are received and both channels are empty at the end of u. □

Lemma 34 *Let $S = (M_1, M_2)$ be a compatible system without mixed local state such that M_1 and M_2 are deterministic and let $s_o \xrightarrow{u} s$ be an execution of S such that $s = (q_1, q_2; \varepsilon, \varepsilon)$. Then $\ell(u|_1) = \Phi(\ell(u|_2))$*

PROOF. Since S is compatible, there exists w_1 a local execution of M_1 such that $\ell(w_1) = \Phi(\ell(u|_2))$. From Lemma 33, there exists an execution v from s_o such that $v|_1 = w_1$ and $v|_2 = u|_2$. Let us show that $u|_1 = w_1$. If it is not the case, let $t_{u|_1}$ and t_{w_1} be the first different transitions in $u|_1$ and w_1 respectively. Transitions $t_{u|_1}$ and t_{w_1} start from the same local state in M_1. Since there is no mixed state, they are either the ith receiving transition or the ith sending transition in their respective paths $u|_1$ and w_1. Furthermore, $t_{u|_1}$ and t_{w_1} correspond to the same ith receiving transition or the ith sending transition in $u|_2$. So, they have identical labels; which contradicts the fact that M_1 is deterministic. Therefore, $u|_1 = w_1$ and thus $\ell(u|_1) = \Phi(\ell(u|_2))$. □

Theorem 35 *A compatible system $S = (M_1, M_2)$ without mixed local state, and such that M_1 and M_2 are deterministic, is half-duplex.*

PROOF. By contradiction, suppose S is not half-duplex. Then, from Lemma 30, there exists a 1-bounded execution u from s_o leading to a stable configuration $s = (q_1, q_2; \varepsilon, \varepsilon)$ such that there exist $a_1, a_2 \in \Sigma$ with $\delta_1(q_1, 1!a_1) \neq \emptyset$ and $\delta_2(q_2, 2!a_2) \neq \emptyset$.

From Lemma 34, we have $\Phi(\ell(u|_2)) = \ell(u|_1)$. Since S is compatible, there exists in M_1 a local execution w_1 starting from q_{o_1} such that $\ell(w_1) = \Phi(\ell(u|_2).2!a_2)$ $= \ell(u|_1).2?a_2$. Since M_1 is deterministic, q_1 has an output transition labeled by the receiving action $2?a_2$. But by hypothesis, q_1 has also an output transition labeled by the sending action $1!a_1$, which contradicts the absence of mixed nodes. Thus S is half-duplex. □

4 Generalization and Regular Model Checking

In the preceding sections we showed how several properties can be verified on half-duplex systems of two machines. In this section, we look for a generalization of these nice results for systems of any number of machines and channels. One way is to consider a generalization of the half-duplex notion. Another way is to seek for semi-algorithms which can be applied on any system and will halt

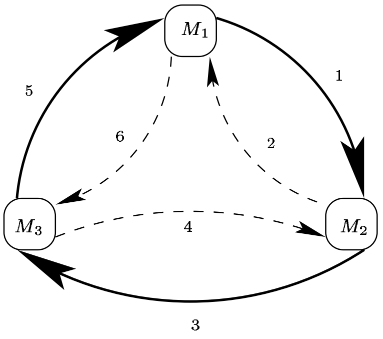


Fig. 5. Simulation of a Turing machine by a half-duplex system

successfully on, at least, half-duplex systems of two machines. This last way belongs to the field of *regular model checking* [BG96,BW98,ABJM99,DLS01].

4.1 Half-duplex Systems of many Machines

There are two immediate generalizations of the half-duplex notion for systems of more than two machines: (a) the *natural generalization* and (b) the *restricted generalization.*

4.1.1 The natural generalization

The *natural generalization* is to define as half-duplex all systems in which each pair of machines linked by two channels, one in each direction, has a half-duplex communication. Unfortunately, this class allows the simulation of a Turing machine with such a system of three communicating machines. This is done by the following construction. Let us consider the simulation of a Turing machines in [BZ83] or [FM97]. It involves two machines which do not have a half-duplex communication since each machine can send messages while its input channel is not empty. Let us add a third machine which acts only as a repeater. The system in Fig. 5 illustrates this. We obtain a half-duplex system which still simulates a Turing machine (channels 2, 4 and 6 are unused!). Thus we have.

Theorem 36 *The natural generalization of half-duplex systems with three machines is able to simulate Turing Machines.*

So this generalization is not analyzable and we cannot adopt the approach used in the previous sections.

4.1.2 The restricted generalization

The *restricted generalization* is to consider systems in which each reachable configuration has at most one non-empty channel. This kind of communication is often used in radio communications where transceivers of several machines use the same frequency (at any moment only one machine is allowed to send messages). Recall that systems which contains a machine allowed to receive messages from a channel its has sent messages to are able to simulate Turing machines (the content of the channel simulates the content of the tape). Assume we do not allow this last case, then the hypothesis of the restricted generalization is sufficient to describe the reachability set of such systems by using techniques similar to those explained in the previous sections. The idea is the same : since every configuration has at most one non empty channel, then the content of that channel is a regular language.

4.2 Regular model checking

Regular model checking is an approach to compute the reachability sets of infinite state systems. One represents, symbolically, sets of states by regular languages and one develops *meta-transitions* [BW98] which can compute, in one step, infinite sets of successors. This amounts to compute $R'(X)$ for a given channel-recognizable subset X and a given relation R' representing a subpart of $post^*$ the transitive and reflexive closure of the transitive relation *post* of the system.

We saw in the previous sections that the reachability set of half-duplex systems of two machines is channel-recognizable. Now we use the key element of the construction of this reachability set to propose the kind of meta-transitions needed to make regular model checking succeed on half-duplex systems of two machines. As we will see, the semi-algorithms proposed will also succeed on a version of the alternating bit protocol which is not half-duplex with respect to Definition 8.

To simplify the notations, we consider, in the remainder of this section, communicating systems made of a single CFSM. This is not really a restriction as for each communicating system of many CFSMs we can associate, as follows, a system made of a single CFSM.

Definition 37 *Let $S = (M_1, \ldots, M_n)$ be a system of CFSMs with $M_i = (Q_i, q_{o_i}, \Sigma, \delta_i)$. The* associated CFSM *of S is $M = (Q, q_o, \Sigma, \delta)$ such that: $Q = Q_1 \times \cdots \times Q_n$, $q_o = (q_{o_1}, \ldots, q_{o_n})$ and δ is the least relation verifying*

$$\Big((q_1, \ldots, q_c, \ldots, q_n), \pi, (q_1, \ldots, q'_c, \ldots, q_n)\Big) \in \delta \textit{ if } \exists c \in \{1, \ldots n\} \ (q_c, \pi, q'_c) \in \delta_c$$

Now, we define *post* the transition relation between sets of states.

Definition 38 *Let $M = (Q, q_o, \Sigma, \delta)$ be a CFSM on p channels. post is the relation between subsets of the state space of M, $Q \times (\Sigma^*)^p$, such that:*

$$post(X) = \bigcup_{t=(q,\pi,q')\in\delta} \{(q', \vec{x'}) \,\big|\, \exists (q, \vec{x}) \in X,\ (q, \vec{x}) \xrightarrow{t} (q', \vec{x'})\}$$

As usual, we note post the transitive and reflexive closure of post. Remark that $RS(M) = post^*(\{s_o\})$.*

Lemma 39 *Let M be a CFSM on p channels and X be a channel-recognizable set of configurations of M. Then $post(X)$ is a calculable channel-recognizable set of configurations of M.*

PROOF. From Theorem 11, since X is channel-recognizable, it is a finite union of sets of the form $\{q\} \times X_1 \times \ldots \times X_p$ with $q \in Q$ and X_i recognizable sets of Σ^*. Since δ is a finite set, it remains to show that $t(X)$ is effectively recognizable with $X = \{q\} \times X_1 \times \ldots \times X_p$ and $t(X) = \{(q', \vec{x'}) \,\big|\, \exists (q, \vec{x}) \in X,\ (q, \vec{x}) \xrightarrow{t} (q', \vec{x'})\}$. There is two cases :

$t = (q, c!a, q')$. Then $t(X) = \{q'\} \times X_1 \times \ldots \times X_c.a \times \ldots \times X_p$.
$t = (q, c?a, q')$. Then $t(X) = \{q'\} \times X_1 \times \ldots \times a^{-1}X_c \times \ldots \times X_p$ with $a^{-1}X_c = \{y \,\big|\, ay \in X_c\}$.

In both cases, the result is a computable channel-recognizable set. □

The reachability set of a half-duplex systems S of two machines is included in the set $H(S)$ of Definition 22. Therefore we propose a set of meta-transitions which allows to compute $H(S)$ in a few steps.

Definition 40 *Let $M = (Q, q_o, \Sigma, \delta)$ be a CFSM on p channels, $q, q' \in Q$ and $1 \leq c \leq p$. We define:*

- *$M^{c!}_{q,q'} = (Q, \{q\}, \{1, \ldots, p\} \times \{!, ?\} \times \Sigma, \delta', \{q'\})$ the finite state automaton with initial state q, single final state q' and $\delta' = \{(q_1, \pi, q_2) \in \delta \,\big|\, \exists a \in \Sigma, \pi = c!a\}$. The language of $M^{c!}_{q,q'}$ is the set of all sequences, from q to q', made of sending actions on channel c.*
- *$post_{c!}$ the relation between subsets of the state space of M, $Q \times (\Sigma^*)^p$, such that:*

$$post_{c!}(X) = \{(q'; \vec{x'}) \,\big|\, \exists (q; \vec{x}) \in X\ \exists q' \in Q\ \exists u \in L(M^{c!}_{q,q'}),\ (q; \vec{x}) \xrightarrow{u} (q', \vec{x'})\}$$

$post_{c!}$ is the meta-transition (part of $post^*$) which consists of applying only sequences of sending actions on channel c.

As stated in the following lemma, the effect of a meta-transition $post^*$ on a channel-recognizable set is computable.

Lemma 41 *Let M be a CFSM on p channels and X be a channel-recognizable set of configurations of M. Then all $post_{c!}(X)$, for $1 \leq c \leq p$, are effective recognizable sets of configurations of M.*

PROOF. From Theorem 11, since X is channel-recognizable, it is a finite union of sets of the form $\{q\} \times X_1 \times \ldots \times X_p$ with $q \in Q$ and X_i recognizable sets of Σ^*. Therefore, it remains to show that $post_{c!}(\{q\} \times X_1 \times \ldots \times X_p)$ is effectively recognizable; which is a direct consequence of the fact that:

$$post_{c!}(\{q\} \times X_1 \times \ldots \times X_c \times \ldots \times X_p) = \bigcup_{q' \in Q} \{q'\} \times X_1 \times \ldots \times X_{c-1} \times X_c.\mathrm{proj}_c(L(M^{c!}_{q,q'})) \times X_{c+1} \times \ldots \times X_p$$

□

In Definition 22, $H(S)$ which appeared to be the channel-recognizable reachability set of a half-duplex system S of two machines was defined. A generalisation of $H(S)$ for systems of more than two machines is computed by Procedure 1.

```
Procedure 1
Require: $M = (Q, q_o, \Sigma, \delta)$
  $X = RS_1(M)$
  $X' = X \cup post_{p!}(\ldots post_{2!}(post_{1!}(X))\ldots)$
  return $X'$
```

In [CF97], *quasi-stable* systems have been defined in order to generalize for systems of any number of machines the good properties of half-duplex systems of two machines. This was an ad hoc definition of systems for which each reachable state can be reached by an execution during which the size of each channel evolves as depicted in Fig. 2.2 and such that the reachability set remains channel-recognizable.

Proposition 42 *The algorithm of Procedure 1 computes the reachability set of half-duplex systems of two machines and the reachability set of quasi-stable systems.*

PROOF. A straightforward consequence of the fact that quasi-stable systems include half-duplex systems of two machines [CF97, Remark 25] and of [CF97, Lemma 26]. □

Procedure 2 Semi-algorithm with meta-transitions $post_{c!}$

```
Require: $M = (Q, q_o, \Sigma, \delta)$
  $X' = \{s_o\}$
  repeat
    $X = X'$
    $X' = X \cup post(X) \cup post_{1!}(X) \cup \ldots \cup post_{p!}(X)$
  until $X' \subseteq X$
  return $X'$
```

In the context of regular model checking, the meta-transitions $post_{c!}$ will be preferably used as stated in Procedure 2. Instead of making a composition of the different $post_{c!}$, we make a union of the applications of these meta-transitions which corresponds better to the generic scheme of semi-algorithms. This Procedure is linked with Procedure 1 as follows.

Proposition 43 *If the reachability set of a system is computed by Procedure 1 then it is computed by Procedure 2.*

PROOF.

Let M be a communicating system made of a single machine and whose reachability set is computed by Procedure 1. $RS_1(M)$ is the set of all configurations reachable by executions whose each intermediate configuration has at most one messages in every channel. Consider Procedure 2. From the presence of $X \cup post(X)$ in the expression of X', in less than $|Q \times \Sigma^p|$ iterations, X will contain $RS_1(M)$. From that point, $post_{p!}(\ldots post_{2!}(post_{1!}(X))\ldots)$ is computed in less than p other steps thanks to the presence of $post_{c!}(X)$ in the expression of X'. □

The following corollary is a immediate consequence of the two preceding propositions.

Corollary 44 *Let S be a half-duplex system of two machines. Then, the semi-algorithm of Procedure 2 halts on S and returns its reachability set.*

Therefore we have a semi-algorithm, Procedure 2, which can be applied on any communicating system and which gives the exact reachability set of half-duplex systems of two machines.

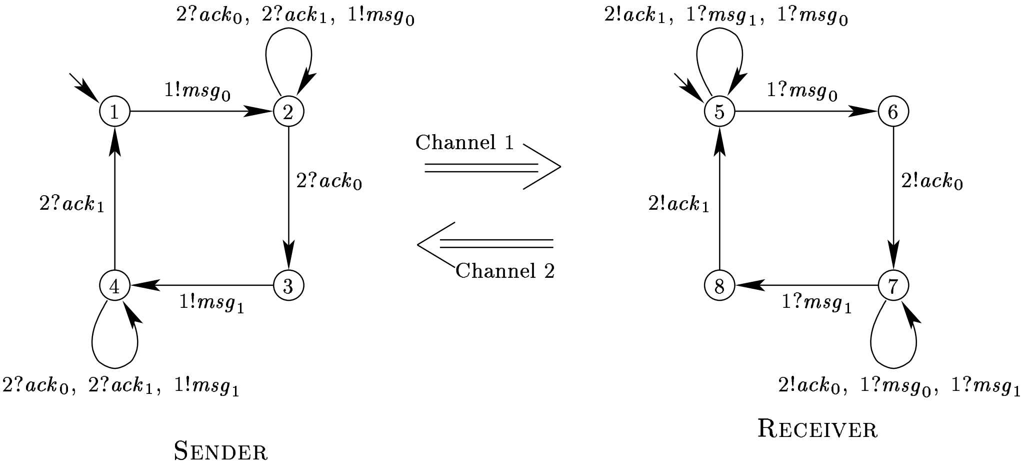


Fig. 6. A modelisation of the alternating bit protocol

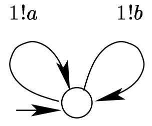


Fig. 7. A simple system not taken into account by meta-transitions in the literature

In the next example, the semi-algorithm of Procedure 2 is successfully applied on a system which is not half-duplex, with respect to Definition 8, and whose reachability set is not computable by the algorithm of Procedure 1.

Example 45 *Let us consider the communicating system in Fig. 6. It simulates the well-known alternating bit protocol [BRW69]. The sender emits its first message with the alternating bit set to 0 (msg_0), then waits for an acknowledgement of that message. At this point, losses (simulated by the loop* $(5, 1?msg_0, 5)$*) may occur. Consequently, it may be necessary to send again the message several times (accomplished by the loop* $(2, 1!msg_0, 2)$*). When the first message has been received and acknowledged, the next message the sender will send will get the alternating bit set to 1 (msg_1) and so on. The other loops suppress from the channels the duplicated messages, simulate losses of acknowledgements or simulate the reemissions of acknowledgements.*

The semi-algorithm 1 applied on the alternating bit protocol in Fig. 6 halts and returns the following reachability set :

$$\begin{array}{lll}
\{1\} \times \{5\} \times \mathit{msg_1}^* \times \mathit{ack_1}^* & \cup\ \{2\} \times \{5\} \times (\mathit{msg_1}^*.\mathit{msg_0}^+ \cup \mathit{msg_0}^*) \times \mathit{ack_1}^* & \cup \\
\{2\} \times \{6\} \times \mathit{msg_0}^* \times \mathit{ack_1}^* & \cup\ \{2\} \times \{7\} \times \mathit{msg_0}^* \times (\mathit{ack_1}^*.\mathit{ack_0}^+ \cup \mathit{ack_0}^*) & \cup \\
\{3\} \times \{7\} \times \mathit{msg_0}^* \times \mathit{ack_0}^* & \cup\ \{4\} \times \{7\} \times (\mathit{msg_0}^*.\mathit{msg_1}^+ \cup \mathit{msg_1}^*) \times \mathit{ack_0}^* & \cup \\
\{4\} \times \{8\} \times \mathit{msg_1}^* \times \mathit{ack_0}^* & \cup\ \{4\} \times \{5\} \times \mathit{msg_1}^* \times (\mathit{ack_0}^*.\mathit{ack_1}^+ \cup \mathit{ack_1}^*) &
\end{array}$$

Meta-transitions like $post_{c!}$ are quite natural. This is all the more surprising that they have not been considered before. In the literature, meta-transitions which have been considered are essentially loops [BG96,BH97]. Therefore, the semi-algorithms build upon them cannot compute the reachability set of systems as simple as the one depicted in Fig. 7.

5 Model-checking of Half-Duplex Systems against PLTL

Since we can compute a recognizable representation of their reachability sets, several interesting properties are decidable on half-duplex systems. Unfortunately, we show in this section a negative result which excludes the verification of a large class of properties useful to check: those expressed in PLTL or in CTL. The *Propositional Linear Temporal Logic (PLTL)* and *Computational Tree Logic* [Eme90] are classical logics for dealing with the behavior of systems.

The set of formulas of PLTL is the least set built from a set *Prop* of atomic propositions, and closed under the application of the boolean connectives, the unary temporal operator $\bigcirc$ (*next*, also noted X), and the binary operator $\mathcal{U}$ (*until*). A PLTL formula is interpreted over the sequence of configurations of an execution. The key elements of the semantic is as follows (a more complete presentation can be found in [Eme90], or in [AJ96a] for a similar use of PLTL on CFSMs).

First of all, a set of atomic propositions, $f(s) \subseteq \mathit{Prop}$, is associated with each configuration s such that $f(s)$ depends only on the control state of s.

- a sequence of states $s_1, s_2, s_3, \ldots$ satisfies a proposition $p \in \mathit{Prop}$ if $p \in f(s_1)$.
- a sequence of states $s_1, s_2, s_3, \ldots$ satisfies a formula $\bigcirc\varphi$ if the sequence $s_2, s_3, \ldots$ satisfies φ.
- a sequence of states $s_1, s_2, s_3, \ldots$ satisfies a formula $\varphi_1 \mathcal{U} \varphi_2$ if and only if there exists $i \geq 1$ such that the sequence $s_i, s_{i+1}, \ldots$ satisfies φ_2 and for all $j < i$, the sequence $s_j, s_{j+1}, \ldots$ satisfies φ_1.
- the boolean connectives are interpreted as usual.
- a communicating system satisfies a PLTL formula if all of its executions from the initial configuration satisfy that formula.

We also use the following abbreviations:

- $\Box\varphi$, "always φ", for $\varphi \mathcal{U}$ *false*
- $\Diamond\varphi$, "eventually φ", for *true* $\mathcal{U}\,\varphi$.

To show the undecidability of PLTL on half-duplex systems, we use the fact that it is undecidable to know whether a Turing machine features any infinite computation. In [FM97] it is shown that any Turing machine T can be simulated by a daisy communicating machine, M_T, like the one in Fig. 8, such that

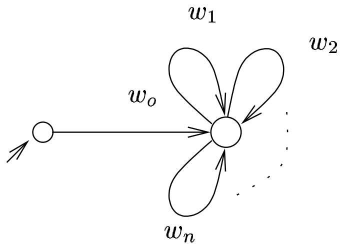


Fig. 8. M_T, a daisy CFSM simulating a Turing machine T

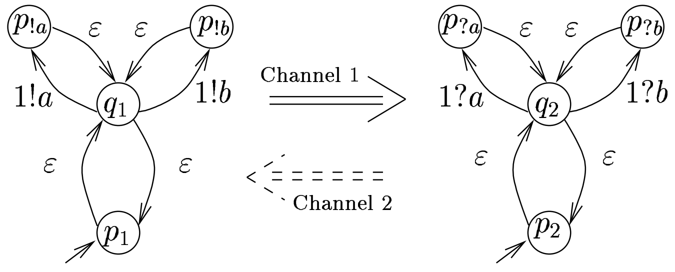


Fig. 9. The half-duplex system S_H when $\Sigma = \{a, b\}$

T has an infinite computation if and only if M_T has also an infinite execution. The communicating machine M_T acts on a single channel and is composed of an initialization sequence w_o of communicating actions and of n elementary cycles w_i of other sequences of communicating actions.

Now we present a half-duplex system of two machines S_H and a PLTL formula Φ_{M_T} such that the executions of S_H which satisfy Φ_{M_T} are similar to the infinite executions of M_T.

The system S_H is composed by a sender which can send everything in the first channel and a receiver which can receive everything from that channel. The second channel is not used, thus S_H is trivially half-duplex. Each local state of the two machines is labeled by a proposition that will be used to observe the behavior of S_H by the formula Φ_{M_T}. The convention is that a configuration of S_H is labeled by the union of labels of its local control states. As an example, the global state $s = (p_{!a}, q_2, abb, \varepsilon)$ is labelled by $f(s) = \{p_{!a}, q_2\}$.

More precisely, let $M_T = (Q, q_o, \Sigma, \delta)$ be the daisy communicating machine, as given in [FM97], simulating a Turing machine T. Then, $S_H = (M_1, M_2)$ with $M_i = (Q_i, p_i, \Sigma, \delta_i)$ be such that:

- $Q_1 = \{p_1, q_1\} \cup \{p_{!a} \,\big|\, a \in \Sigma\}$, $Q_2 = \{p_2, q_2\} \cup \{p_{?a} \,\big|\, a \in \Sigma\}$,
- $\delta_1 = \{(p_1, \varepsilon, q_1), (q_1, \varepsilon, p_1)\} \cup \{(q_1, 1!a, p_{!a}), (p_{!a}, \varepsilon, q_1) \,\big|\, a \in \Sigma\}$,
  $\delta_2 = \{(p_2, \varepsilon, q_2), (q_2, \varepsilon, 2_1)\} \cup \{(q_2, 1?a, p_{?a}), (p_{?a}, \varepsilon, q_2) \,\big|\, a \in \Sigma\}$

Figure 9 depicts the case where $\Sigma = \{a, b\}$.

To construct Φ_{M_T} from M_T depicted in Fig. 9, first, we associate a formula φ_i with each communicating sequences w_i as follows:

Let $w_i = x_{i,1}x_{i,2}\dots x_{i,m_i}$ with $x_{i,j} \in \{1!a,\ 1?a \,\big|\, a \in \Sigma\}$ then

$$\begin{aligned}\varphi_i = \ & (p_1 \wedge p_2) \wedge \bigcirc q_1 \wedge \bigcirc\bigcirc q_2 \wedge \\ & \textstyle\bigwedge_{j=1}^{m_i} \bigcirc^{2j+1} P_{i,j} \wedge \bigcirc^{2j+2} P'_{i,j} \wedge \\ & \bigcirc^{2(m_i+1)+2} p_1 \wedge \bigcirc^{2(m_i+1)+2} p_2\end{aligned}$$

with $P_{i,j}$ and $P'_{i,j}$ such that:

- if $x_{i,j} = 1!a$ then $P_{i,j} = p_{!a}$ and $P'_{i,j} = q_1$.
- if $x_{i,j} = 1?a$ then $P_{i,j} = p_{?a}$ and $P'_{i,j} = q_2$.

Then we take Φ_{M_T} such that :

$$\Phi_{M_T} = \varphi_0 \wedge \Box\Diamond(p_1 \wedge p_2) \wedge \Box\Big(p_1 \wedge p_2 \ \rightarrow \bigvee_{i=1}^{n} \varphi_i\Big)$$

The next propositions express immediate consequences of the definitions of φ_i and Φ_{M_T}.

Proposition 46 *A sequence of configurations $s_1, s_2, \dots$ of S_H satisfies φ_i if and only if the control state of s_1 is (p_1, p_2), the control state of $s_{2(m_i)+2}$ is also (p_1, p_2), and a prefix of this sequence corresponds to the execution of $w_i = x_{i,1}x_{i,2}\dots x_{i,m_i}$.*

Proposition 47 *Let M_T be a daisy communicating machine, S_H and Φ_{M_T} be its corresponding system and formula. Then M_T presents an infinite execution if and only if an execution of S_H satisfies Φ_{M_T}.*

PROOF. Straightforward from the two equivalent following facts:

- An infinite execution of M_T begins with the communicating sequence w_o and continues with an infinite succession of communicating sequences w_i.
- An execution satisfying Φ_{M_T} begins with the communicating sequence w_o, passes infinitely often by the control state (p_1, p_2) and each time it reaches this control state (p_1, p_2) it executes one of the communicating sequences w_i immediately followed by a configuration whose control state is (p_1, p_2). □

Note that, in the preceding sections, we have not consider systems with ε transitions, however the different results still hold (in the proof of Lemma 20 it suffices to add the case where t is an ε transition). So we have.

Theorem 48 *PLTL is undecidable on half-duplex systems.*

PROOF. From the preceding propositions, we have the equivalence: the communicating system S_H satisfies $\neg\Phi_{M_T}$ (which mean by definition that none of its executions satisfy Φ_{M_T}) if and only if there is no infinite execution in M_T. And from [FM97] it is undecidable to know whether there exists an infinite execution in M_T. □

A similar, though slightly more complicated, work can be done for CTL.

6 Conclusion

In this paper, we have studied half-duplex communication for systems of communicating finite state machines. We have shown that half-duplex systems with two machines have effective channel-recognizable reachability sets. This allows us to solve several verification problems (the reachability problem, the boundedness problem, etc.) on this kind of systems. Unfortunately, half-duplex systems with more than two machines happen to be Turing powerful. Furthermore, everything is not decidable on half-duplex systems with two machines. We have shown that the classical temporal logics PLTL and CTL (and thus CTL*) are undecidable on these systems.

Compared to general systems with recognizable channels [Pac82] and lossy channel systems [AJ96b,AJ96a], our decision procedures are effective and efficient (polynomial time). Moreover, though lossy channel systems have a channel-recognizable reachability set, this set is not computable [CFP96], in contrast here with the case of half-duplex systems of two machines. Moreover, the reachability problem is decidable for lossy channel systems but it is not primitive recursive [Sch02] while it may be solved in polynomial time for half-duplex systems.

By the way, it may be interesting to combine these two hypotheses: loss of messages and half-duplex communication. Indeed, with a slight modification of the algorithms given here (in order to simulate the losses), we are still able to compute the reachability set of such systems. The reason is that losses do not disrupt the half-duplex hypothesis. Our work can also be extend to do parametric verification of the half-duplex part of systems : instead of computing the reachability set of systems from a single initial state with channels empty, we can compute the set of states reachable by half-duplex executions from a given channel-recognizable set of states.

We have also situated this work in the field of regular model checking. From the technical part of our results, we have extracted the kind of meta-transitions which allows the generic semi-algorithm to successfully compute the reachability set of, at least, half-duplex systems of two machines. This kind of meta-transitions, $post_{c!}$, enrich the other kind of meta-transitions used in semi-algorithms dedicated to communicating systems. An interesting perspective is to confront this approach on systems with counters and lossy channels such as the BRP protocol [ACABJ04].

Acknowledgements

We thank the anonymous referees for their helpful suggestions.

References

[ABJM99] Parosh Aziz Abdulla, Ahmed Bouajjani, Bengt Jonsson, and Nilsson Marcus. Handling global conditions in parameterized system verification. In *CAV'99*, number 1633 in LNCS, pages 134–150, 1999.

[ABS01] A. Annichini, Ahmed Bouajjani, and M. Sighireanu. TReX : A Tool for Reachability Analysis of Complex Systems. In *Proceedings of the 13th International Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *Lecture Notes in Computer Science*, Paris (France), July 2001. Springer.

[ACABJ04] P. A. Abdulla, A. Collomb-Annichini, A. Bouajjani, and B. Jonsson. Using forward reachability analysis for verification of lossy channel systems. *Formal Methods in System Design*, 25(1):39–65, July 2004. Kluwer Academic Pub.

[AJ96a] Parosh Aziz Abdulla and Bengt Jonsson. Undecidable verification problems for programs with unreliable channels. *Information and Computation*, 130(1):71–90, 10 October 1996.

[AJ96b] Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 15 June 1996.

[Ber79] J. Berstel. *Transductions and Context-Free Languages.* B.G. Teubner Stuttgart, 1979.

[BFLP03] Sébastien Bardin, Alain Finkel, Jérôme Leroux, and Laure Petrucci. FAST: Fast Acceleration of Symbolic Transition systems. In Warren A. Hunt, Jr and Fabio Somenzi, editors, *Proceedings of the 15th International Conference on Computer Aided Verification (CAV'03)*, volume 2725 of *Lecture Notes in Computer Science*, pages 118–121, Boulder, Colorado, USA, July 2003. Springer.

[BG96] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. In *Proc. of* 8^{th} *CAV (August), USA*, volume 1102, pages 1–12. LNCS, 1996.

[BH97] A. Bouajjani and P. Habermehl. Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations. In *Proceedings of the 24th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume LNCS 1256, pages 560–570, July 1997.

[BRW69] K.A. Bartlette, Scantlebury R.A., and P.T. Wilkinson. A note on reliable full-duplex transmissions over half-duplex links. *Communications of the ACM*, 12(5), May 1969.

[BW94] Bernard Boigelot and Pierre Wolper. Symbolic verification with periodic sets. In *Proc. 6th International Conference on Computer-Aided Verification*, volume 818 of *LNCS*, pages 55–67, Stanford, june 1994. Springer-Verlag.

[BW98] Bernard Boigelot and Pierre Wolper. Verifying systems with infinite but regular state spaces. In *CAV'98*, volume 1427 of *LNCS*, pages 88–97, June 1998.

[BZ83] Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. *JACM*, 30(2):323–342, April 1983.

[CF97] Gérard Cécé and Alain Finkel. Programs with quasi-stable channels are effectively recognizable. In *Proc. of CAV'97, LNCS 1254, Israel*, pages 304–315, June 1997.

[CFP96] Gérard Cécé, Alain Finkel, and Iyer S. Purushothaman. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31, 1996.

[DLS01] Dennis Dams, Yassine Lakhnech, and Martin Steffen. Iterating transducers. In *Conference on Computer Aided Verification*, volume 2102, pages 286–297, 2001.

[Eme90] E.A. Emerson. *Handbook of Theoretical Computer Science*, chapter 16, pages 996–1072. Elsevier Science Publishers, 1990.

[Fin90] Alain Finkel. Reduction and covering of infinite reachability trees. *Information and Computation*, 89(2):144–170, 1990.

[Fin93] Alain Finkel. The minimal coverability graph algorithm. In *Advances in Petri Nets*, volume LNCS 674, pages 210–243. Springer-Verlag, 1993.

[Fin94] Alain Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, 7:129–135, 1994.

[FL02] Alain Finkel and Jérôme Leroux. How to compose Presburger-accelerations: Applications to broadcast protocol s. In Manindra Agrawal and Anil Seth, editors, *Proceedings of the 22nd Conference on*

Fundations of Software T echnology and Theoretical Computer Science (FSTTCS'02), volume 2556 of *Lecture Notes in Computer Science*, pages 145–156, Kanpur, India, December 2002. Springer.

[FM97] Alain Finkel and Pierre McKenzie. Verifying identical communicating processes is undecidable. *Theoretical Computer Science*, 174:217–230, 1997.

[FPIS03] Alain Finkel, S. Purushothaman Iyer, and Grégoire Sutre. Well-abstracted transition systems: Application to FIFO automata. *Information and Computation*, 181(1):1–31, February 2003.

[GMY84] M. G. Gouda, E. G. Manning, and Y. T. Yu. On the progress of communication between two finite state machines. *Information and Control*, 63:200–216, 1984.

[Hol91] Gerard J. Holzmann. *Design and Validation of Computer Protocols.* Prentice-Hall, 1991.

[JJ93] Thierry Jéron and Claude Jard. Testing for unboundedness of fifo channels. *Theoretical Computer Science*, 113:93–117, 1993.

[las] The Liège Automata-based Symbolic Handler (LASH). Available at: http://www.montefiore.ulg.ac.be/~boigelot/research/lash/.

[Mou89] Hassan Mountassir. Sur la progression de la communication entre processus communicants. étude d'une classe particulière de machines déterministes. Technical report, UFR des Sciences et techniques, Laboratoire d'Informatique; 16 route de Gray, 25030 Besançon FRANCE, September 1989.

[Pac82] Jan K. Pachl. Reachability problems for communicating finit state machines. Research Report CS-82-12, Department of Computer Science University of Waterloo, Waterloo, Ontario, Canada, May 1982.

[Pac87] Jan K. Pachl. Protocol description and analysis based on a state transition model with channel expressions. In *Proc. of Protocol Specification, Testing and Verification, VII*, 1987.

[Sch02] Philippe Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5):251–261, September 2002.

[Tur93] Kenneth J. Turner, editor. *Using formal description techniques. A introduction to Estelle, Lotos and SDL.* Wiley, John & Sons Ltd., Baffins Lane, Chichester West Sussex PO19 1UD, England, 1993.

[VC83] S.T. Vuong and D.D. Cowan. Reachability analysis of protocols with fifo channels. In *ACM SIGCOMM*, Austin, Texas, 1983.