Programs with Quasi-Stable Channels are Effectively Recognizable

(Extended Abstract)

Gérard Cécé Alain Finkel

LSV, CNRS URA 2236; ENS de Cachan, 61 av. du Pdt. Wilson
94235 Cachan Cedex, FRANCE.
{Gerard.CECE, Alain.FINKEL}@lsv.ens-cachan.fr

Abstract. We consider the analysis of infinite half-duplex systems which consists of finite state machines that communicate over unbounded channels. The property half-duplex for two machines and two channels (one in each direction) says that each reachable state has at least one channel empty.
The contributions of this paper are (a) to give a finite description of the reachability set of such systems, which happens to be effectively recognizable; this description allows us to solve classical verification problems such as: whether a given state is reachable, whether there exist deadlock states, whether the reachability set is finite and whether a specified action is useless; (b) to propose an extension of these results for a new class, systems with quasi-stable channels, which includes systems with similar behavior but which implies more than two machines.

1 Introduction

There are several models which can be used to study distributed algorithms and communicating protocols. The model we use in this paper is that of systems of communicating finite states machines (CFSMs). It consists of programs represented by finite state machines which communicate over unbounded channels. This model is widely used both for verification and validation in languages such ESTELLE, SDL and LOTOS [15]. It has also its own theoretical interest as shown by the literature it has generated in the last few years [1,2,6,9] .

However, this model has the power of Turing machines since it's possible to simulate them by a system of two communicating finite state machines [5,10], thus verification is undecidable for non-trivial problems. This limitation motivates the study of classes of systems for which verification algorithms are possible [1,2,6,8,9,12,14].

The class introduced by Pachl [14], systems with a recognizable reachability set, is interesting since it covers a wide number of protocols and since the reachability problem is solved for it. However, the need to list recognizable sets makes this algorithm inefficient! Moreover, it's not possible to decide whether a given system has a recognizable reachability set, which is the starting point for using Pachl's result.

The term half-duplex is used to qualify communication between two machines which don't send messages simultaneously. We show in this paper that many verification problems are solvable for systems of two machines using this kind of communication. More precisely, we give a recognizable description of their reachability sets and show that the property of being half-duplex for systems with two machines and two channels is decidable. These results are quite surprising since one might think that the half-duplex property doesn't really reduce the power of the model as each machines can transfer data each one in turn. But this hypothesis is precisely too restrictive to simulate a Turing machine since it forbids the possibility of storing the content of the tape in the channels: when an automaton sends messages, since it is not able to receive information from the other automaton, its possible future states are fixed and of finite number. The information it sends must thus be recognizable and we use this fact to describe the reachability set.

The remaining of the paper is as follows. In section 2, we define the model and the problems we want to verify on it. In section 3, we show that half-duplex systems with two machines and two channels are verifiable, we give a recognizable representation of their reachability sets and show that this result is useable since we can also decide whether any system with two machines and two channels is half-duplex. Then in section 4, we suggest a generalization of the half-duplex property, quasi-stable systems. Section 5 is the conclusion.

For lack of space, we mostly give only the ideas of proofs. The complete presentation can be found in the technical report [7].

2 System of Communicating Finite State Machines

2.1 Preliminaries and properties

Let us consider the communicating protocol of Fig. 1. It involves two machines: a sender and a receiver. When the sender has a message to transmit, it first advices the receiver by sending a starting symbol *start*. Then, it sends the main message over an alphabet of two letters, $\{a, b\}$. When this is complete, it advises the receiver by sending an *end* symbol and then waits for an acknowledgment, *ack*. The receiver has a symmetrical behavior. Formally we have:

Definition 1. A *Communicating Finite State Machine* (CFSM) is a finite transition system given by a 4-tuple $M = (Q, q_0, \Sigma, \delta)$ where : Q is a finite set of *states*, $q_0 \in Q$ is the *initial state*, Σ is a finite *alphabet*, and $\delta \subseteq Q \times (\{+,-\} \times \Sigma \times \mathbb{N}) \times Q$ is a finite set of *transitions*. We also consider δ as an application from $Q \times (\{+,-\} \times \Sigma \times \mathbb{N})$ to 2^Q.

Note that the alphabet of M in the usual transition system sense is $(\{+,-\} \times \Sigma \times \mathbb{N})$ rather than Σ, i.e. M sees $(+, a, i)$ and $(-, a, i)$ as single symbols, which we henceforth write as $+a$ and $-a$ respectively when there is no ambiguity about the identity of the channel. Intuitively, $(-, a, i)$ denotes the emission of a in channel i, and $(+, a, i)$ denotes the reception of a from channel i. Furthermore, we do

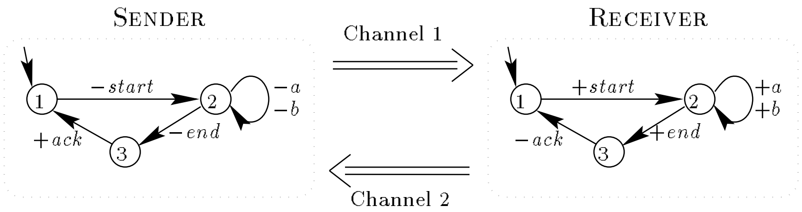


Fig. 1. The protocol S_1

not allow a machine to send and receive messages in/from the same channel, which could then be used as an auxiliary memory. We will need to consider transitions of δ which correspond only to sending actions (receiving actions) whence $\delta^- = \delta \cap Q \times (\{-\} \times \Sigma \times \mathbb{N}) \times Q$, $\delta^+ = \delta \cap Q \times (\{+\} \times \Sigma \times \mathbb{N}) \times Q$. We want now to group communicating machines together in order to deal with systems. Formally, we have:

Definition 2. A *system S of n CFSMs* is a n-tuple of CFSMs $S = (M_1, \ldots, M_n)$ with $M_i = (Q_i, q_{0i}, \Sigma, \delta_i)$ such that there is at most one machine which sends messages to a given channel. And conversely, there is at most one machine which receives messages from a given channel.

In the remainder of the paper and if not state otherwise, S will refer to a system $S = (M_1, \ldots, M_n)$ of CFSMs such that $M_i = (Q_i, q_{0i}, \Sigma, \delta_i)$; we also define $Q = \cup_{i=1,\ldots,n} Q_i$ and $\delta = \cup_{i=1,\ldots,n} \delta_i$ as the set of all transitions of the system. Let p be the number of channels used in S, we can rename the channels in $\{1, \ldots, p\}$. Then a *global state* (abbreviated to a *state*) of S is a $(n+p)-$tuple $s = (\mathbf{q}; \mathbf{x})$ with $\mathbf{q} = (q_1, \ldots, q_n)$, $q_i \in Q_i$ and $\mathbf{x} = (x_1, \ldots, x_p)$, $x_i \in \Sigma^*$. We note λ the empty word of Σ^* and $\mathrm{G}(S)$ the *set of all global states* of S. An element $\mathbf{q} \in Q_1 \times \ldots \times Q_n$ is a *control state* and an element $q \in Q_i$ for $i \in \{1, \ldots, n\}$ is a *local state* (of machine i).

The operational semantics associated with a system of CFSMs is defined by the firing of a transition which changes the current global state in one step.

Definition 3. Let S be a system. A state $s' = (q'_1, \ldots, q'_n; x'_1, \ldots, x'_p)$ is *reachable* from another state $s = (q_1, \ldots, q_n; x_1, \ldots, x_p)$ *by the firing of the transition* t, written $s \to s'$, or redundantly $s \xrightarrow{t} s'$, if one of the following two cases hold :

1. $t = (q_i, (-, a, j), q'_i) \in \delta_i$ such that:
   (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. $t = (q_i, (+, a, j), q'_i) \in \delta_i$ such that:
   (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 the machine M_i in channel j. Condition 2. describes the reception of message a by the machine M_i from channel j.

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 we have $s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \ldots \xrightarrow{t_m} s_{m+1}$. The *initial state* is $s_0 = (\mathbf{q_0}; \boldsymbol{\lambda})$ with $\mathbf{q_0} = (q_{01}, \ldots, q_{0n})$, and $\boldsymbol{\lambda} = (\lambda, \ldots, \lambda)$.
A state s is said *reachable* if $s_0 \xrightarrow{*} s$ and the *reachability set* of S is the set $\mathrm{RS}(S) = \{s \in \mathrm{G}(S) | s_0 \xrightarrow{*} s\}$ of all reachable states.

Example 4. $\mathrm{RS}(S_1) = \{(1,1;\lambda,\lambda),\ (2,1;(start)\{a,b\}^*,\lambda),\ (3,1;(start)\{a,b\}^*(end),\lambda),\ (2,2;\{a,b\}^*,\lambda), (3,2;\{a,b\}^*(end),\lambda),\ (3,3;\lambda,\lambda),\ (3,1;\lambda,(ack))\ \}$.

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 $RT(S)$ with root labelled s_0 such that a node labelled s has a child labelled s' and the arc (s, s') is labelled by transition t iff $s \xrightarrow{t} s'$.
- the *reachability graph* of S is the labelled graph $\mathrm{RG}(S)$ whose nodes are $\mathrm{RS}(S)$ labelled by their corresponding states in $\mathrm{RS}(S)$. A node labelled s has a successor labelled s' and the arc (s, s') is labelled by transition t iff $s \xrightarrow{t} s'$.

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 and then be switched. A consequence is that the reachability graph of the system is finite since channels are bounded in function of the duration of each sending period and of the transmission's rate. We will use the term half-duplex with a less restrictive sense, without notion of elapsed time.

Definition 5. A system $S = (M_1, M_2)$ of two machines with two channels (one in each direction) is *half-duplex* if each reachable state has at least one channel empty.

Example 6. System S_1 of Fig. 1 is half-duplex.

2.2 Properties and decidability for systems of CFSMs

Usually, the verification of systems requires to solve the following kinds of problems: *reachability problems* which are to determine whether a given global state or local state or transition is reachable or executable; *boundedness problems* which are to determine whether the system requires unbounded channels and which ones.

The following theorem states how difficult verification of systems is.

Theorem 7 (Brand & Zafiropulo [5], Finkel & McKenzie [10]). *Systems of CFSMs have the power of Turing's machines. Hence, general verification is undecidable.*

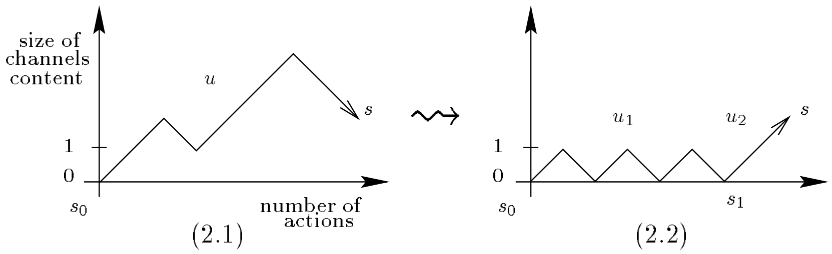


Fig. 2. Transformation of executions

In [13,14], Pachl shows that reachability problems are decidable under the sole assumption that the reachability set is recognizable. Remember that the notion of recognizable set is the natural extension (see [3]) for set of tuple of words of the classical notion of recognizable or regular languages. But we've seen in the introduction that these results are not useable in practice. Furthermore, *knowing that a reachability set is 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 the case of *lossy channel systems* [6]. On the other hand, having an effective recognizable description of the reachability set makes all of the problems just stated, easy to solve [7]. So we have:

Theorem 8. *For a system S of n CFSMs whose reachability set is recognizable and from which we have an effective description, the problems of interest are decidable.*

3 Half-duplex systems with two machines are verifiable

3.1 A Symbolic Reachability Graph

In this subsection and in the two following ones, we restrict ourselves to systems $S = (M_1, M_2)$ with two machines and two channels, the first one from M_1 to M_2 and the second one in the opposite direction. First of all, we show that each reachable state of such half-duplex systems is reached by a particular execution which we use to give a recognizable description of the reachability set.

Example 9. Let's consider the half-duplex system S_1 of Fig. 1. The sequence of actions $u = (-start)(-a)\ (+start)(-b)(-b)(-a)(+a)(+b)$ leads to the state $s = (2, 2; ba, \lambda)$. A representation of u is given by Fig. (2.1). But as shown by Fig.(2.2), u can be reordered in an execution split in two part, $u_1 = (-start)(+start)\ (-a)(+a)(-b)(+b)$ and $u_2 = (-b)(-a)$, such that u_1 is "bounded" by 1 and leads to a stable state s_1, and u_2 is exclusively made by sending messages in the same channel. We will show that this transformation is always available for such half-duplex systems. Hence, we can compute the reachability set in two steps. The first one computes the finite set of all stable states reached by a "1-bounded" execution. For each of these states, the second step gives all states reached by carrying out sending actions in a given channel. Because this last step uses only one machine, we obtain a recognizable set. What follows proves and formalizes this informal description.

We first need to give a more precise description of executions and states reached by "bounded" executions.

Definition 10. Let S be a system.

- an *execution* is a sequence of the form $s_1, t_1, s_2, t_2, \ldots, t_{m-1}, s_m$ such that $s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \ldots \xrightarrow{t_{m-1}} s_m$. Given a starting state s we also consider a sequence of transitions $t_1 t_2 \ldots t_m$ as an *execution* whenever there exists a sequence of states $(s_k)_{k=1,\ldots,m}$ such that $s, t_1, s_1, \ldots, t_m, s_m$ is an execution. If not set otherwise, the starting state will supposed to be the initial one s_0. Given a sequence of transitions representing an execution u, we note $u|_i$ the projection of this sequence on transitions of machines i. Thus, the projection $u|_i$ is the local execution induced by u on machine i.
- an execution $u = s_1, t_1, s_2, \ldots, t_{m-1}, s_m$ of S is *k-bounded* if the sizes of each channel's content of all intermediate states s_i visited by u is less than k.
- the *k-reachability set* of S with $k \in \mathbb{N}$ is the largest subset $\mathrm{RS}_k(S)$ of $\mathrm{RS}(S)$ in which, each state s is reached by a k-bounded execution from s_0.

Remark 11. Given a system S, for every integer $k \in \mathbb{N}$ the set $\mathrm{RS}_k(S)$ is finite and computable.

From an execution, we want to extract the sequence of messages involving a given channel j into an execution; so we define for each j, the morphism $\mathrm{proj}_j : \delta^* \to \Sigma^*$ such that $\mathrm{proj}_j((q,(-,a,j),q')) = \mathrm{proj}_j((q,(+,a,j),q')) = a$ and for $k \neq j$, $\mathrm{proj}_j((q,(-,a,k),q')) = \mathrm{proj}_j((q,(+,a,k),q')) = \lambda$.

Lemma 12. *Let $S = (M_1, M_2)$ be a half-duplex system. For every reachable state $s = (q_1, q_2; x_1, x_2)$ of S, there exists an execution $u_1 u_2$ and a state s_1 such that the three following conditions hold:*

1. $s_0 \xrightarrow{u_1} s_1 \xrightarrow{u_2} s$,
2. *u_1 is 1-bounded, s_1 is a stable state, and,*
3. *$\exists i \in \{1,2\}$ such that $u_2 \in (\delta_i^-)^*$, $x_i = \mathrm{proj}_i(u_2)$ and $x_{3-i} = \lambda$.*

Proof. By induction on the length of an execution u ending at s. There are two cases depending whether the last action of u is a receiving or a sending one.

Definition 13. From a given system, $S = (M_1, M_2)$, we distinguish the following subset of global states:

$$\mathrm{H}(S) = \bigcup_{\substack{(q_1', q_2'; \lambda, \lambda) \in \mathrm{RS}_1(S), \\ (q_1, q_2) \in Q_1 \times Q_2}} \{q_1\} \times \{q_2\} \times L_1(q_1', q_1) \times L_2(q_2', q_2)$$

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

Remark 14. $\mathrm{H}(S)$ is recognizable as a finite union of products of recognizable languages [3]. Secondly, the description of $\mathrm{H}(S)$ is effective since those of the different $L_i(q'_i, q_i)$ are.

Lemma 15. *The reachability set* $\mathrm{RS}(S)$ *of a half-duplex system* S *is equal to* $\mathrm{H}(S)$.

Proof. The set $\mathrm{H}(S)$ describes states reached by a specific execution, hence $\mathrm{H}(S) \subseteq \mathrm{RS}(S)$. The converse part, $\mathrm{RS}(S) \subseteq \mathrm{H}(S)$, is straightforward from Lemma 12 ∎

Example 16. Let's consider system S_1, the reader can verify that $\mathrm{RS}(S_1) = \mathrm{H}(S_1)$. In particular, the subset $(2, 2; \{a, b\}^*, \lambda)$ is obtained by taking $(q'_1, q'_2) = (q_1, q_2) = (2, 2)$ in Definition 13.

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

Theorem 17. *The reachability set of a half-duplex system is* recognizable and computable *in time:*
$O(|(\Sigma + 1)^3 \times Q_1 \times Q_2|(|Q_1| + |Q_2|))$.

Proof. The recognizability and computability of $\mathrm{RS}(S)$ is proved by the preceding Remark and Lemma. Let us now consider the complexity. From Definition 13, we only need to compute stable states of $\mathrm{RS}_1(S)$ since from these states a simple run through sending actions of one of the machine gives other states. Computing of $\mathrm{RS}_1(S)$ is done by a simple reachability search from the starting state s_0. Knowing that $|\mathrm{RS}_1(S)| = |Q_1 \times Q_2 \times (\Sigma + 1)^2|$ and that a reachable state has at most $|\Sigma \times Q_1| + |\Sigma \times Q_2|$ successors, we deduce the complexity. ∎

Corollary 18. *For half-duplex systems, the problems of interest are decidable.*

Proof. Straightforward from Theorems 17 and 8. ∎

Having a recognizable description of the reachability set of a given half-duplex system S allows us the following definition.

Definition 19. 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 recognizable description of the reachability set on this control state.
- the *symbolic reachability graph* of S, $\mathtt{SRG}(S)$, is a graph whose nodes are labelled by elements of $\mathtt{SRS}(S)$ and such that there is an edge labelled by $t \in \delta$ from $(\mathbf{q}; \mathbf{X})$ to $(\mathbf{q}'; \mathbf{X}')$ if and only if every successor, by the execution of transition t, of elements of $(\mathbf{q}, \mathbf{X})$ are in $(\mathbf{q}', \mathbf{X}')$.

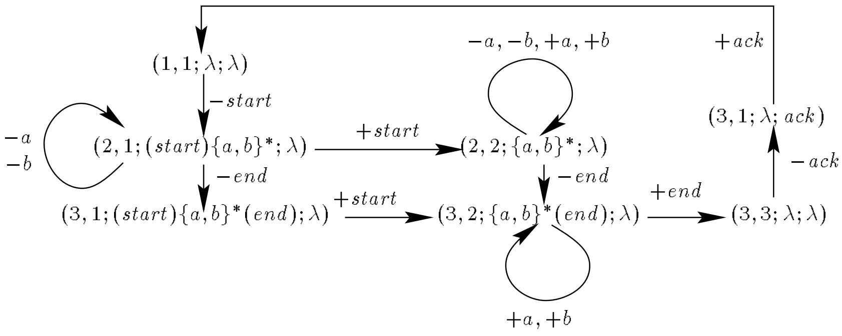


Fig. 3. The symbolic reachability graph of S_1

The symbolic reachability graph of a system is closely related to the minimal coverability graph (MCG) of a Petri net, in the sense that every execution of a system S is a path in $\texttt{SRG}(S)$ and every reachable state 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 minimal coverability graph of a Petri net.

Example 20. 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 mean to describe the reachability set of a half-duplex system. But to be useful, we need to know on which systems these results may be applied. In other words, we have to decide whether a system of two machines S is half-duplex.

Obviously, since a system may have an infinite reachability set, we have to reduce the domain of interest to a finite one. Then, we will be able to verify the half duplex property.

Lemma 21. *A system $S = (M_1, M_2)$ is not half-duplex if and only if there exist $s = (q_1, q_2; \lambda, \lambda) \in \mathrm{RS}_1(S)$ and $a_1, a_2 \in \Sigma$ such that $\delta_1(q_1, (-, a_1, 1)) \neq \emptyset$ and $\delta_2(q_2, (-, a_2, 2)) \neq \emptyset$.*

Proof. The if part is straightforward. For the only if part, we consider one of the smallest execution u leading to a state which contradicts the half-duplex property. Then the state we're looking for is the stable one obtained by using lemma 12 on u truncated of its last action.

■

Theorem 22. *The half-duplex property is decidable in time:*
$O(|(\Sigma+1)^3 \times Q_1 \times Q_2|(|Q_1| + |Q_2|))$.

Proof. Since $\mathrm{RS}_1(S)$ is finite and computable, the property of Lemma 21 is checkable. The complexity comes from that of Theorem 17 ■

4 Generalization for more than two machines

In the preceding section we showed how half-duplex systems with two machines and two channels can be verified. In this section, we look for a generalization of these nice results for systems with more than two machines and more than two channels. One way is to consider a generalization of the half-duplex notion. We see two possibilities: (a) the *natural generalization* and (b) the *restricted generalization*. We will see that neither of them satisfies our requirements. This will lead us to define a new class of systems whose restriction for systems of two machines and two channels corresponds to this of half-duplex.
First of all, let's consider the two generalizations.

(a) The *natural generalization* is to define as half-duplex all systems in which each pair of machines linked by two channels have a half-duplex communication. Unfortunately, this class allows the simulation of a Turing machine, with such a system of three machines, by the following construction. Let's consider the simulation of a Turing machines in [5]. It involves two machines which don't have half-duplex communication since each machine can send messages while its input channel is not empty. Let's add a third machine which will act only as a repeater of a channel . Then, we have three machines and three channels forming a cycle. Let's add three dummy channels in the opposite direction. We obtain a half-duplex system which still simulates a Turing machine. Thus we have.

Theorem 23. *Half-duplex systems with three machines are able to simulate Turing Machines.*

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

(b) The *restricted generalization* is to consider systems in which each reachable state has at most one non-empty channel. This kind of communication is often used in radio communications where transceivers use the same frequency (at a given moment only one machine is allowed to send messages). And this condition is sufficient to describe the reachability set of such systems by use of the same techniques explained in the previous sections. However, consider now two independent subsystems satisfying this generalization. We would have liked the whole system to satisfy it too. But this is not the case since two channels (respectively in each subsystem) can be simultaneously non-empty.

Another possibility is to start from the good property of half-duplex system with two machines and two channels saying this of Lemma 12. This leads us to the following Definition 24. In what follows we note j^- the unique machine allowed to send messages in channel j, $T_j^- = \delta \cap Q \times (\{-\} \times \Sigma \times \{j\}) \times Q$ (resp. $T_j^+ = \delta \cap Q \times (\{+\} \times \Sigma \times \{j\}) \times Q$) the set of sending (resp. receiving) transitions in (resp. from) channel j, $T_j = T_j^- \cup T_j^+$ and $ш$ the shuffle operator of two words e.g. $x \,ш\, y = \{y_1 x_1 y_2 \ldots y_n x_n y_{n+1} | x = x_1 \ldots x_n,\ y = y_1 \ldots y_{n+1}\}$.

Definition 24. Let S be a given system with p channels.

– An execution u is said to be *quasi-stable* if:

(a) $u \in v'_1 v''_1 \sqcup\!\sqcup \ldots \sqcup\!\sqcup v'_p v''_p$ with $v'_j \in (T_j^- T_j^+)^*$ and $v''_j \in (T_j^-)^*$. The sequence v'_j is called the *quasi-stable part* of channel j and the sequence v''_j its *increasing part.* Fig.(2.2) is also an illustration of the change in size of the content of a given channel during a quasi-stable execution.

(b) For each decomposition $u = u_1 t u_2$ with t a transition belonging to the increasing part of channel j then $u_2|_{j^-} \in (T_j^-)^*$. In other words, the machine which sends messages during the increasing part of a channel j, doesn't execute any action involving other channels during its future actions.

– A system is said to be *quasi-stable*, or to have quasi-stable channels, if every reachable state can be reached by a quasi-stable execution.

Remark 25. The reader can verify using Lemma 12 that half-duplex systems with two machines and two channels are quasi-stable systems.

Quasi-stable systems can be characterized by a more effective property which will allow us to effectively compute their reachability sets.

Lemma 26. *Let $s = (\mathbf{q}; x_1, \ldots, x_p)$ be a state of a system S with n machines and p channels. The state s is reachable by a quasi-stable execution if and only if s can be reached by an execution u such that:*

1. $u = u_0 u_1 u_2 \ldots u_p$,
2. u_0 *is 1-bounded and leads to a stable state,*
3. $\forall j, k \in \{1, \ldots, p\}, j \neq k, (u_j \neq \lambda$ *and* $u_k \neq \lambda \Longrightarrow j^- \neq k^-)$,
4. $\forall j \in \{1, \ldots, p\} \;\; x_j = \mathrm{proj}_j(u_j)$.

Proof. The if part is straightforward. For the only if part, we show, by the independence of the actions involved that a quasi-stable execution can be reordered in an execution satisfying the requirements.

Definition 27. From a given system S with n machines and p channels, we distinguish the following subset of global states:

$$\mathcal{Q}(S) = \bigcup_{\substack{(\mathbf{q}';\ \boldsymbol{\lambda}) \in \mathrm{RS}_1(S) \text{ and} \\ J \subseteq \{1,\ldots,p\} \text{ such that} \\ \forall j,k \in J,\ j \neq k \Rightarrow j^- \neq k^-}} \left\{ (\mathbf{q};\mathbf{x}) \,\middle|\, \begin{array}{l} j \in J \Rightarrow x_j \in L(j, q'_{j^-}, q_{j^-}) \\ j \notin J \Rightarrow x_j = \lambda \\ \forall i \notin \{j^- | j \in J\}, q'_i = q_i \end{array} \right\}$$

With $L(j, q'_{j^-}, q_{j^-})$ the regular language recognized by the machine M_{j^-} considered as an finite automaton and such that: q'_{j^-} is the initial state; q_{j^-} is the unique final state; each edge labeled by $(-, a, j)$ in M_{j^-} is labeled by the corresponding message (i.e. a); other edges are removed.

Since $\mathcal{Q}(S)$ is a finite union of products of regular languages, the set $\mathcal{Q}(S)$ is recognizable [3]. Furthermore, its description is computable.

Lemma 28. *For any quasi-stable system S,* $\mathrm{RS}(S) = \mathcal{Q}(S)$.

Proof. Lemma 26 says that a reachable state s is reachable from a stable state s' of $\mathrm{RS}_1(S)$ such that from s' to s there is an execution with each machine sending messages in at most one channel. The definition of $\mathcal{Q}(S)$ describes exactly all states reached by this procedure.

Theorem 29. *The reachability set of a quasi-stable system $S = (M_1, \ldots, M_n)$ with $M_i = (Q_i, q_{0i}, \Sigma, \delta_i)$ and p channels is* recognizable and computable *in time:* $O((\prod_{i=1}^{n} |Q_i|)|(\Sigma + 1)|^{n+1}(\sum_{i=1}^{p} |Q_i|))$.

Proof. Similar to that of Theorem 17.
From this theorem, we deduce that.

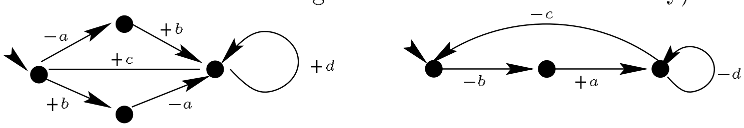
Corollary 30. *For a quasi-stable system, the problems of interest are decidable.*

The property of quasi-stability is decidable but needs a different approach than that of Lemma 21.

Theorem 31. *The quasi-stable property for a system S is decidable.*

Proof. From Definition 27 and Lemma 26, the problem of showing that a system S is quasi-stable amounts to that of showing that $\mathrm{RS}(S) = \mathcal{Q}(S)$. Because $\mathcal{Q}(S) \subseteq \mathrm{RS}(S)$, we have only to verify whether $\mathcal{Q}(S)$ is a fixpoint of the successor function. This is possible by the recognizability of $\mathcal{Q}(S)$.

Example 32. Here is an example of a quasi-stable system which is not half-duplex (the two machines can send a message to each other simultaneously):



5 Conclusion

In this paper, we have shown that half-duplex systems with two machines have a effectively recognizable reachability set. Which allows us to define a symbolic reachability set and graph for this class of systems. However, half-duplex systems with more than two machines happen to be Turing powerful. This has led us to define a new class: quasi-stable systems, whose reachability set stays effectively recognizable for systems of any number of machines and channels. The restriction of this class for systems of two machines includes the half-duplex one.

Compared to general systems with recognizable channels [13] and lossy channel systems [1,2], our results are effective. We have applied these results to a class, systems with compatible communication, defined by Gouda et al. [11]. Only the boundedness problem was solved for it and we proved that this class is included in the half-duplex one with two machines and two channels. Hence reachability problems are also solvable.

Symbolic reachability sets and graphs are classical notions to represent infinite transition systems. Boigelot and Godefroid [4] used a symbolic reachability graph which approximates by recognizable sets the reachability set of any given system. Nodes of their graph are labelled by recognizable sets and edges by transitions or cycle of transitions of the underlying system. The approach we've used in this paper has led us to consider a similar construction but with recognizable languages as symbolic transitions, which is strictly more general. We intend to investigate the decidability of temporal logic such as LTL for quasi-stable systems.

References

1. P. Abdulla and B. Jonsson. Verifying Programs with Unreliable Channels. In *Proc.* 8^{th} *Annual IEEE Symposium of Logic in Computer Science*, 1993.
2. P. Abdulla and B. Jonsson. Undecidable Verification Problems for Programs with Unreliable Channels. in *Proc. of ICALP*, vol. 820 of LNCS pp.316- 1994.
3. J. Berstel. Transductions and Context-Free Languages. *B.G. Teubner Stuttgart*, 1979.
4. B. Boigelot and P. Godefroid. Symbolic Verification of Communication Protocols with Infinite State Spaces Using QDDs. In *Proc. of* 8^{th} *CAV (August), USA* LNCS 1102,pp.1–12, 1996.
5. Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. *JACM*, 30(2):323–342, 1983.
6. G. Cécé, A. Finkel and S. Purushothaman Iyer. Unreliable Channels Are Easier to Verify Than Perfect Channels. In *Information and Computation*, vol.124, No 1,20-31, 1996
7. G. Cécé and A. Finkel Programs with Quasi-Stable Channels are Effectively Recognizable. Technical Report, LSV, ENS de Cachan, 1997 (available via the authors)
   grenade
8. A. Finkel. Reduction and covering of Infinite Reachability Trees. *Information and Computation*, vol.89, n° 2, pp. 144–170, 1990.
9. A. Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, 7:129–135, 1994.
10. A. Finkel and P. McKenzie. Verifying identical communicating Processes is undecidable. *to appear in TCS* 1997
11. M. G. Gouda, E. G. Manning and Y. T. Yu. On the Progress of Communication between Two Finite State Machines. *Information and Control,* 63:217–225, 1984.
12. T. Jéron and C. Jard. Testing for unboundedness of fifo channels. *Theoretical Computer Science* 113, pp.93-117, 1993.
13. J. K. Pachl. Reachability Problems for Communicating Finite State Machines *Research Report CS-82-12,* University of Waterloo, Dpt. of Computer Science, 1982.
14. J. K. Pachl. Protocol description and analysis based on a state transition model with channel expressions. In *Proc. of Protocol Specification, Testing and Verification, VII*, May 1987.
15. Kenneth J. Turner. Using Formal Description Techniques; an introduction to Estelle, Lotos and SDL. ©*John Wiley & Son Ltd.*, 1993.