Distrib Comput (1994) 7: 129–135

DISTRIBUTED COMPUTING

Decidability of the termination problem for completely specified protocols

Alain Finkel

Ecole Normale Supérieure de Cachan, L.I.F.A.C., 61 avenue du Président Wilson, F-94235 Cachan Cedex, France

Received January 1989 / Accepted October 1993

Alain Finkel is a Professor of Computer Science at the Ecole Normale Supérieure of Cachan. His research is concerned with the models of concurrency and the practical possibilities to verify and to validate distributed systems. He is also interested by the cognitive interfaces.

Summary. In this paper, we present a new class of protocols called completely specified protocols. Each protocol is represented as a system of Communicating Finite State Machines. The class of completely specified protocols is such that each message that can be received by a Finite State Machine, can also be received in every local state of the Finite State Machine. These protocols are important because they allow for modelling unbounded fifo channels and make it possible to decide the Termination Problem, that is whether the reachability tree is finite or not. An example of our techniques is given using a practical problem concerning link protocols.

Key words: Communicating Finite State Machines – Termination detection – Completely specified protocols – Higman's Lemma.

1 Introduction

We are interested in the *termination detection* of protocols. The Termination Problem (TP) is well-known, especially for distributed systems and rewriting system, and it is very important: for example, when election or agreement protocols are implemented, they must also terminate. In order to analyze the protocols, and particularly to decide the TP, we need a formal framework. We have chosen to model the protocols by using automata theoretical techniques. Determining behavior properties (termination, regular behavior) and detecting design error (unbounded channel, deadlock, unspecified reception) in protocols can be performed in the same way as finding the solution of some classical analysis problems on the corresponding automata ([5], [17], [38], [13]). In these models, a protocol terminates if its reachability tree is finite.

Protocols are often modeled as systems (or networks) of *finite state machines* that communicate exclusively by exchanging messages over unbounded, unidirectional, error-free fifo channels (a non-perfect channel can be modeled by an additional finite state machine consuming messages passing through it [37]). There are generally two fifo channels between each pair of machines in the system, one for sending messages and the other one for receiving. Each machine has a finite number of states and state transition rules. Each state transition rule is associated with the action of sending or receiving one message to or from one of the output or input channels of the machine. Communicating Finite State Machines (CFSM) systems are used for protocol modeling [34], analysis ([2], [5], [23], [28], [31], [38], [19], [13], [24]) and synthesis ([37], [17], [9], [29], [26]). Take for example the well-known alternating bit protocol ([2]), or the call establishment/clear protocol in X.25 ([9]), or the simplified OSI Transport protocol ([3]). Furthermore, several new protocol design systems have recently been developed using the CFSM model ([1]).

Nevertheless, there is a price to pay for choosing CFSM systems with unbounded fifo channels as a protocol modeling tool. In fact, there are no algorithms able to decide all of the reachability problems, even for CFSM classes with fifo channel alphabets containing at least two letters. CFSM systems with unbounded fifo channels have the power of Turing machines when there is at least one unbounded fifo channel with a message alphabet of at least two messages [5]. A similar result can be found in the framework of an equivalent model, called the fifo nets model [14]. Therefore, it can be proved that almost all reachability problems are undecidable.

Consequently, we must look for a tradeoff between flexibility and generality. For instance, algorithms for checking classical properties are available if we choose as our model CFSM whose fifo channel alphabets are of size

one (easily simulated by Petri nets) or systems of interacting CFSM using CSP-type input/output operations ("rendez-vous"). However, it will then be difficult if not impossible to simulate, for example, unbounded fifo channels or "the zero testing capacity" (i.e., the possibility to test whether a channel is empty or not). CFSM systems with bounded channels are equivalent to finite automata.

It is preferable to use restricted classes of these systems in which *the TP is decidable*. Typical restrictions involve limiting the allowable sequences of messages that a fifo channel can enable ([35], [31], [11], [28], [38], [8], [13], [14]). We now provide a short overview of the main results concerning the decidability of the TP in CFSM systems.

- The TP is *undecidable* for *systems of two* CFSM [5].
- The TP is *decidable* for *CFSM systems if all channel alphabets are of a single message type* or *if all channels are bounded*. These particular CFSM systems are equivalent to Petri nets, and the TP is decidable for Petri nets ([21], [6]). The TP also remains decidable for systems of two CFSM if one of the machines sends one, and only one type of message [31].
- The TP is *decidable* for systems of *monogeneous CFSM* (a CFSM is monogeneous if the input language of each fifo channel is included in a finite union of languages xy^*, where x and y are finite words) ([13]).
- The TP is *decidable* for systems of *linear CFSM* (a CFSM is linear if the input-language of each fifo channel is included in a language $a^*b^* \ldots z^*$, where $a, b, \ldots, z$ are letters) ([19], [8]).
- The TP is *decidable* for *CFSM systems which have a recognizable channel property* (a CFSM system has a recognizable channel property if the projection on every channel of the reachability set is a recognizable language) ([27], [28]).
- The TP is *decidable* for *well-ordered protocols* ([35]).

Other papers consider CFSM systems with bounded fifo channels ([32]) as a set of process equations in [38] or as "tree protocols" in [5]. Gouda introduced [16] the technique of closed covers for verifying progress for CFSM, but there is no automatic method for finding a closed cover.

Almost all previous TP decidability proofs have been based on the construction of a finite and a representative part of the reachability tree associated to the protocol. This notion has been formalized in the general framework of structured transition systems ([12]).

One of the main properties necessary for computing a finite and a representative part of the reachability tree is the existence of an *ordering* $\leqq$ (in fact, a *quasi-ordering*, it means a reflexive and transitive binary relation, is sufficient) on the reachability set. The second necessary property is the *monotonicity* of the transition system: if a transition t can be enabled from a state s to reach a state s_1, then t can be enabled from any state $s' \geqq s$ after a finite delay to reach a state $s'_1 \geqq s_1$. The finite degree of the reachability tree makes it possible to apply Koenig's lemma [22]. Thus, an *infinite path* does exist in an infinite reachability tree. When the ordering $\leqq$ is a *well-ordering* (an ordering for which an infinite increasing subsequence can be extracted from every infinite sequence), we may detect an infinite path in the reachability tree. Finally, the ordering $\leqq$ has to be *decidable* for states, that is to say given two states s and s', one may decide whether $s \leqq s'$ or not. This makes it possible to compute an algorithm which constructs the reachability tree until two comparable states s, s' are encountered on a same branch so that s' is reachable from s.

We define *completely specified protocols* as CFSM systems *in which* each machine of the protocol *can receive any message in any local state, and can stay in the same local state*. We will prove that the Termination Problem is decidable for completely specified protocols. Protocols using *non-perfect fifo channels*, are one example of completely specified protocols.

The paper is organized as follows: Section 2 gives the general overall definitions concerning CFSM used throughout the paper. Section 3 presents the main results, i.e. the decidability of the Termination Problem for completely specified protocols.

2 Specification model of protocols: communicating finite state machines

In this section, we provide a general range of definitions concerning CFSM.

Let X be an *alphabet* (i.e., a finite set) whose elements are called *letters* or *messages*. The concatenation operator "." allows for constructing words on X. A word x on X is a sequence of letters from X. The *empty word* is denoted by 1. X^* is the set of finite words on X (X^* contains the empty word) and X^+ is equal to $X^* - \{1\}$; we write $|x|$ for the length of x: we have $|x.x'| = |x| + |x'|$, for all words x, $x' \in X^*$, and $|1| = 0$. We denote by $|$ (said "is a subword of") the *ordering* (an ordering is a reflexive, anti-symmetrical and transitive relation) on words defined as follows: for two words $u, v \in A^*$, $u|v$ if the word v can be written

$v = w_1 . u_1 . w_2 . u_2 \ldots w_n . u_n . w_{n+1}$ where $u = u_1 . u_2 \ldots u_n$ and $w_1, w_2, \ldots, w_{n+1}$ are words of A^*. An ordering $\leqq$ on a set S is a *well-ordering* if one can always extract an increasing (for $\leqq$) infinite subsequence $\{s_{n_i}\}$ from every infinite sequence $\{s_n\}$ of elements, $s_n \in S$. We will often use the well-known Koenig's lemma [22]: an infinite tree with a finite degree (i.e., each node has a finite number of successors) has an infinite branch. Finally, we need the following result: if A is finite then the ordering $|$ is a well-ordering on A^* ([20]). This means that from any infinite sequence of finite words $w_1, w_2, \ldots, w_n, \ldots$ one can extract an infinite increasing subsequence $w_{i_1} | w_{i_2} | \ldots | w_{i_n} | \ldots$ such that $i_1 < i_2 < \ldots < i_n < \ldots$.

Definition. A *finite (infinite, respectively) state machine* is a quadruplet $M = (S, T, h, s_0)$ where S is the finite (infinite, respectively) set of states, T is a finite set of transitions, h is a partial transition function from $S \times T$ into S, and s_0 is the initial state. The function h is naturally extended from $S \times T^+$ into S as follows: for every sequence of transitions $x \in T^+$ and for every transition $t \in T$, $h(s, xt)$ is defined by $h(s, xt) = h(h(s, x), t)$. A *labelled finite state machine* is a finite state machine $M = (S, T, h, s_0, L)$ where (S, T, h, s_0) is a finite state machine and $L: T \to E$ is a labelling function from T into a set E. A *transition system* is a finite *or* an infinite state machine.

In a CFSM system, the finite state machines communicate exclusively by exchanging messages via connecting channels. There are generally two unidirectional fifo channels between each pair of machines in the system. Each state transition rule is associated with either sending or receiving one message to or from one of the output or the input channels of the machine.

Definition. A *protocol* P is a set of labelled finite state machines which communicate via a set of common fifo channels. Formally, $P = (M_1, \ldots, M_p, \ldots, c_{ij}, \ldots)$ where for every $i = 1, \ldots, p$ $M_i = (S_i, T_i, h_i, s_{0i}, L_i)$, and $L_i : T_i \to (\{-, +\} \times A \times \{1, \ldots, p\}) \cup \{1\}$ with $A = \{a_1, \ldots, a_n\}$, and for every $i, j = 1, \ldots, p$ $(i \neq j)$, c_{ij} is the fifo channel from the machine M_i to the machine M_j.

The set A is the *global alphabet of messages of protocol P.*

The meaning of $L_2(t) = (-, a_3, 7)$ is that transition t of machine M_2 is associated with the sending of message a_3 to machine M_7 via channel c_{27}.

The meaning of $L_1(t) = (+, a_5, 8)$ is that transition t of machine M_1 is associated with the receiving of message a_5 from machine M_8 via channel c_{81}.

The meaning of $L(t) = 1$ is that no message is sent or received.

Notations

$\text{SEND}_{ij} = \{a \in A;$ there is $t \in T_i / L_i(t) = (-, a, j)\}$.
SEND_{ij} is the set of messages that can be sent by M_i to M_j via channel c_{ij}.
$\text{RECEIVE}_{ij} = \{a \in A;$ there is $t \in T_j / L_j(t) = (+, a, i)\}$.
RECEIVE_{ij} is the set of messages that can be received by M_j from M_i via the channel c_{ij}.
We define the *alphabet* A_{ij} of channel c_{ij} as $A_{ij} = \text{SEND}_{ij} \cup \text{RECEIVE}_{ij}$ and we assume $A = \bigcup A_{ij}$.
When a protocol contains only two communicating finite state machines, we write $L_i(t) = +a$ instead of $L_i(t) = (+, a, j)$ and $L_i(t) = -a$ instead of $L_i(t) = (-, a, j)$ with $i \neq j$.

Figure 2.1. represents a protocol. The two machines are M_1 and M_2. Circles denote the (local) states of the machines. The transition labelled $-a$ indicates that the transition is associated with sending an "a" message to the output fifo channel of the machine. (channel destinations are not explicitly given here because there is only a single input and output channel for each machine) The label $+b$ (in machine M_2) indicates that the message "b" is to be received in state 1. The starting state for M_1 and M_2 is the state labelled 1. The machine M_1 can send a message "a", thus transiting from state 1 to state 2; after sending "c" then "d", machine M_2 will receive "a" and also transit from state 3 to state 1.

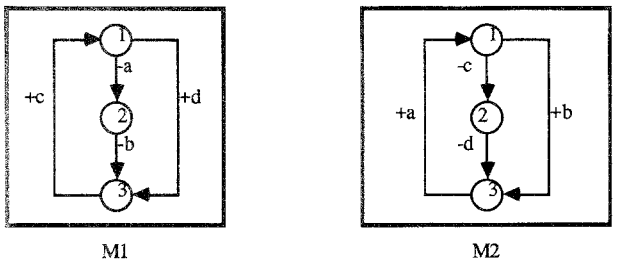

Fig. 2.1

This particular protocol can be analyzed because the two fifo channels are monogeneous ([13]). Most of the common reachability problems, such as the Termination Problem, the Reachability Problem and the Liveness Problem are decidable by using a finite coverability tree which is a finite representation of the reachability tree. Still, we must keep in mind that protocols are generally not analyzable ([5]).

We now define the global states of a protocol and their firing rules.

Definition. A *global state* s of a protocol $P = (M_1, \ldots, M_p, \ldots, c_{ij}, \ldots)$ is given by a vector $s = (s_1, \ldots, s_p, \ldots, w_{ij}, \ldots)$ where s_i is the current state of machine M_i and $w_{ij} \in A_{ij}^*$ is the content of channel c_{ij}. A transition $t \in T_i$ of a protocol $P = (M_1, \ldots, M_p, \ldots, c_{ij}, \ldots)$ is firable from the global state $s = (s_1, \ldots, s_p, \ldots, w_{ij}, \ldots)$ in the following two cases, and the global state s' is reached from s by firing t.

1. $L_i(t) = (-, a, k)$, $a \in A_{ik}$ for some $k \in \{1, \ldots, i-1, i+1, \ldots, p\}$ and $h_i(s_i, t)$ is defined.
The *new global state* s' is defined as follows:

$$s' = (s_1, \ldots, s_{i-1}, h_i(s_i, t), s_{i+1}, \ldots, s_p, \ldots, w_{ik}.a, \ldots, w_{ij}, \ldots).$$

2. $L_i(t) = (+, a, k)$, $a \in A_{ki}$ for some $k \in \{1, \ldots, i-1, i+1, \ldots, p\}$, $w_{ki} = a.w'_{ki}$, $w'_{ki} \in A_{ki}^*$ and $h_i(s_i, t)$ is defined.
The *new global state* s' is defined as follows:

$$s' = (s_1, \ldots, s_{i-1}, h_i(s_i, t), s_{i+1}, \ldots, s_p, \ldots, w'_{ki}, \ldots, w_{ij}, \ldots) \text{ with } w_{ki} = a.w'_{ki}.$$

These firing rules are naturally extended to finite sequences of transitions.

Remark. A protocol $P = (M_1, \ldots, M_p, \ldots, c_{ij}, \ldots)$ often has an *initial state* $s_0 = (s_{01}, \ldots, s_{0p}, \ldots, w_{0ij}, \ldots)$ where s_{0i} is the initial state of machine M_i and w_{0ij} is the initial content of channel c_{ij}. Moreover, the initial state often has empty channels ($w_{0ij} = 1$) but this is not a condition for obtaining our result.

Example. In Fig. 2.1., we reach the global state $(3, 1, ab, -)$ from the initial global state $(1, 1, -, -)$ by firing the sequence labelled by $(-a).(-b)$.

These firing rules correspond to channels with infinite capacities (potentially unbounded). When all the channels are bounded, the CFSM system is equivalent to a finite state machine and the TP is thus obviously decidable.

Definition. The *reachability set* $RS(P)$ of a protocol $P = (M_1, \ldots, M_p, \ldots, c_{ij}, \ldots)$ with an initial state $s_0 = (s_{01}, \ldots, s_{0p}, \ldots, w_{0ij}, \ldots)$ is the set of all states which are reachable from s_0.

A *reachability tree* $RT(P)$ defined from the firing rules is associated with a protocol P:
1) the root r is labelled by the initial state s_0,
2) a node n, labelled by s, has no successor if no transition $t \in \bigcup T_i$ is firable from s,

3) if n is a node, labelled by s, which does not satisfy condition 2) then for every transition $t \in \bigcup T_i$ such that t is firable from s and one reach a state s', we create a new node n' labelled by s', and a new arc labelled by t, from node n to node n'.

The *reachability graph* $RG(P)$ is obtained from the reachability tree by identifying nodes with the same label.

Definition. The reachability tree of a protocol P naturally defines an *associated transition system* $M(P) = (S, T, h, s_0)$ where $S = RS(P)$ is the reachability set of P, $T = \bigcup T_i$ is the union of all transitions sets, h is a partial transition function from $RS(P) \times \bigcup T_i$ into $RS(P)$ defined by: when a transition t is firable from a state s, $h(s, t)$ is equal to s' (defined by the firing rules), and $s_0 = (s_{01}, \ldots, s_{0p}, \ldots, w_{0ij}, \ldots)$ is the initial state of the protocol P.

Remark. The reachability graph of a protocol always has a finite degree because the function h goes from $S \times T$ into S, and because T is finite.

Definition. A protocol P *terminates* when its reachability tree is finite.

Example. The communicating finite state machines in Fig. 2.2 are obtained from those of Fig. 2.1; we have reduced each communicating finite state machine by identifying state "2" and state "3". This reduction preserves the Termination Property (i.e., P_1 terminates if P'_1 terminates). The reachability graph of the protocol P'_1 (Fig. 2.2) is described in Fig. 2.3.

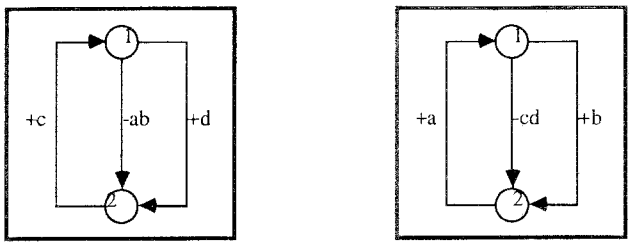

Fig. 2.2. Protocol $P'1$

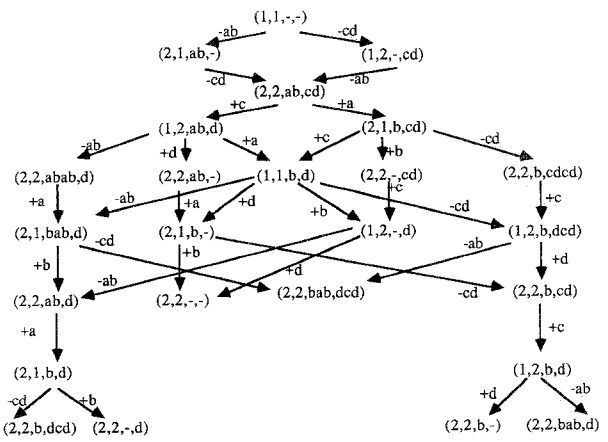

Fig. 2.3. The reachability graph of $P'1$

We see that protocol P'_1 (and P_1) terminates because its reachability graph is finite and contains no cycles.

3 Decidability of the termination problem

In the general framework of protocols, it is well known that the Termination Problem (TP) is undecidable (it is shown in [4, pp. 31–39] how to simulate a Turing machine by a protocol; and so, every non-trivial property is undecidable for protocols by Rice's Theorem). Our aim is to present a new class of protocols for which TP is decidable. This class of protocols, called completely specified protocols, is such that every finite machine of the protocol, in every local state, can receive every message, and it can stay in the same local state.

Protocols using non-perfect fifo channels (which allow loss of messages) are an example of this class.

Suppose now that fifo channels in the protocol P'_1 are not perfect and that they can lose messages. We can specify this new protocol by transforming the protocol P'_1 into a protocol named P_2 by adding, in each local state, a loop labelled by all possible receptions (Fig. 3.1). This new protocol P_2 models channels with loss of messages: for example, machine M_1 of protocol P_2 can consume the message c from its local state 1, and yet still remain in state 1.

Definition. A protocol $P = (M_1, \ldots, M_p, \ldots, c_{ij}, \ldots)$ is *completely specified* if for every $i, j = 1, \ldots, p$, $i \neq j$, for every local state $s \in S_i$ and for every message $s \in \mathrm{SEND}_{ji}$, there exists at least one transition $t \in T_i$ such that $h_i(s, t) = s$ and $L_i(t) = (+, a, j)$.

Example. Protocol P_2 is a completely specified protocol.

The model of "tree protocols" described in [5] is similar to ours, but it is limited to protocols with bounded channels.

It is easy to decide whether a protocol is completely specified or not.

Proposition 3.1. *The property for a protocol to be completely specified is decidable with a complexity in* $o(\sum_{i,j=1,\ldots,p,i\neq j}(|S_i| * |\mathrm{SEND}_{ji}| * |T_i|))$.

Proof. Omitted. □

Remark. If a completely specified protocol terminates, then all the channels are empty.

The main result of this paper is the following.

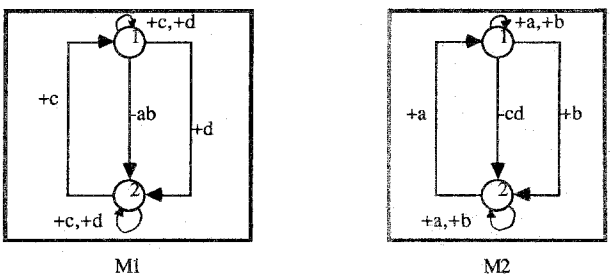

Fig. 3.1

Theorem 3.2. *The Termination Problem is decidable for completely specified protocols.*

The complete proof of this theorem needs five lemmas. Using an ordering $\leqq$ on the reachability set, we prove that there is an infinite computation if, and only if, there exist two states s, s' in the reachability tree such that s is an ancestor of s' and $s \leqq s'$.

First, let us define the following relation on the set S of states.

Definition. Let $s = (s_1, \ldots, s_p, \ldots, w_{ij}, \ldots)$ and $s' = (s'_1, \ldots, s'_p, \ldots, w'_{ij}, \ldots)$ be two global states of a protocol $P = (M_1, \ldots, M_p, \ldots, c_{ij}, \ldots)$. The relation on the set S of global states, denoted by $\leqq$, is defined as follows: $s \leqq s'$ if for every $i, j = 1, \ldots, p$, $s_i = s'_i$ and $w_{ij} | w'_{ij}$.

Lemma 3.3. *The relation $\leqq$ is a well-ordering.*

Proof. It can be easily verified that $\leqq$ is an ordering because the intersection of two orderings (the equality and subword relations) is still an ordering. Let us prove that $\leqq$ is a well-ordering. Since each set S_i of local states is finite, we can extract from any infinite sequence of global states $\{s_n\}$, $n \in \mathbb{N}$, an infinite subsequence, denoted by $\{s_m\}$, $m \in M \subseteq \mathbb{N}$, $s_m = (s_{m1}, \ldots, s_{mp}, \ldots, w_{mij}, \ldots)$, such that for every $i = 1, \ldots, p$ and for every $q, q' \in M$, $s_{qi} = s_{q'i}$. Moreover, by applying the Higman's lemma c times (let c be the number of channels c_{ij}), we find an infinite subsequence of $\{s_m\}$, denoted by $\{s_d\}$, $d \in D \subseteq M$, $s_d = (s_{d1}, \ldots, s_{dp}, \ldots, w_{dij}, \ldots)$, such that for every $i, j = 1, \ldots, p$ and for every $d, d' \in D$, such that $d \leqq d'$, we have $w_{dij} | w_{d'ij}$. Hence, we find an infinite increasing sequence $\{s_d\}$ of global states. Thus, the ordering $\leqq$ is a well-ordering. □

On the basis of the same notations introduced in Sect. 2, we will now prove the following result.

Lemma 3.4. *The transition system, $M(P) = (S, T, h, s_0)$, associated with a completely specified protocol P, is monotonous:*

$$(\forall s, s' \in S)(\forall x \in T^+)(s \leqq s' \Rightarrow (h(s, x) \text{ is defined } \Rightarrow \exists x' \in T^+, h(s, x) \leqq h(s', x') \text{ and } x | x')) .$$

Proof. We proceed by induction on the length of the word $x \in T^+$. Using the induction rule, it is sufficient to prove the following implication.

$$(\forall s, s' \in S)(\forall t \in T)(s \leqq s' \Rightarrow (h(s, t) \text{ is defined } \Rightarrow \exists y' \in T^*, h(s, t) \leqq h(s', y't)) .$$

Let us denote by $s = (s_1, \ldots, s_p, \ldots, w_{ij}, \ldots)$ and $s = (s'_1, \ldots, s'_p, \ldots, w'_{ij}, \ldots,)$ the two global states of the protocol $P = (M_1, \ldots, M_p, \ldots, c_{ij}, \ldots)$ such that $s \leqq s'$. As $s \leqq s'$, we deduce that $s_i = s'_i$ for every $i = 1, \ldots, p$. Let us suppose that $h(s, t)$ is defined. Let $r = h(s, t)$ be the new global state reached by firing the transition t. We distinguish the following two cases:

1. $L_i(t) = (-, a, k)$, $a \in A_{ik}$ for some $k \in \{1, \ldots, p\} - \{i\}$. We obtain:

$$r = (s_1, \ldots, s_{i-1}, h_i(s_i, t), s_{i+1}, \ldots, s_p, \ldots, w_{ik}.a, \ldots, w_{ij}, \ldots) .$$

The transition t is still firable from s' and we obviously reach a state $r' = h(s', t) \geqq r$.

2. $L_i(t) = (+, a, k)$, $a \in A_{ki}$ for some $k \in \{1, \ldots, p\} - \{i\}$ and the content of the fifo channel c_{ki} is $w_{ki} = a.v_{ki}$, $v_{ki} \in A^*_{ki}$ because $h(s, t)$ is defined. We obtain:

$$r = (s_1, \ldots, s_{i-1}, h_i(s_i, t), s_{i+1}, \ldots, s_p, \ldots, v_{ki}, \ldots, w_{ij}, \ldots) .$$

Let us note $w'_{ki} = u'_{ki}.a.v'_{ki}$ such that u'_{ki} does not contain any letter a. Since the protocol P can always receive messages, it is possible to consume the word u'_{ki} and stay in the same local state s_i. Hence, there exists a sequence of transitions $x \in T^*$ which empties the word u'_{ki}, and does nothing else, such that:

$$h(s', x) = (s_1, \ldots, s_p, \ldots, a.v'_{ki}, \ldots, w_{ij}, \ldots) .$$

The transition t is firable from $h(s', x)$ and we reach the state r':

$$r' = h(s', xt) = (s_1, \ldots, s_p, \ldots, v'_{ki}, \ldots, w_{ij}, \ldots) .$$

Prove that $v_{ki} | v'_{ki}$. Since $s \leqq s'$, we have $w_{ki} = a.v_{ki} | w'_{ki} = u'_{ki}.a.v'_{ki}$ Because u'_{ki} does not contain letter a, we deduce that $v_{ki} | v'_{ki}$.

And then we have $h(s, t) = r \leqq r' = h(s', x')$, $x' = xt$ and obviously $x | x'$. □

Remark. Given two words w and w', the relation $w | w'$ can be decided in time $o(|w'|)$; hence, given two words w and w', one needs no more than $o(|w| + |w'|)$ tests for deciding whether w and w' are comparable (i.e., $w | w'$ or $w' | w$).

In order to decide the TP, we need to be able to compare two states for the ordering $\leqq$. Let us define the length of a state $s = (s_1, \ldots, s_p, w_1, \ldots, w_q)$ by the following sum:

$$|s| = p + \sum_{i=1,\ldots,q} |w_i| .$$

Lemma 3.5. *The ordering $\leqq$ is decidable. Moreover, given two states s and s', one may decide whether they are comparable with a complexity in $o(|s| + |s'| - p)$.*

Proof. The equality and the subword orderings are obviously decidable, hence this is also the case for $\leqq$. For deciding whether $s \leqq s'$ or $s' \leqq s$, with $s = (s_1, \ldots, s_p, w_1, \ldots, w_q)$ and $s' = (s'_1, \ldots, s'_p, w'_1, \ldots, w'_q)$, one needs to make no more than p comparisons for the equality between integers (for the p local states of the p automaton) *plus* $\sum_{i=1,\ldots,q} |w_i| + |w'_i|$ comparisons between letters (for the q words in the q channels). □

Let us establish the following equivalence.

Lemma 3.6. *$RT(P)$ is infinite $\Leftrightarrow \exists s, s' \in RT(P)$ s' is reachable from s and $s \leqq s'$.*

Proof. That the right side implies the left side is a consequence of Lemma 3.4. For the converse, let us suppose that the reachability tree of P is infinite. Because the reachability tree has a finite degree, there exists an infinite branch, labelled by the infinite sequence $\{s_n\}$, issued from the root r, by Koenig's lemma. By Lemma 3.3, $\leqq$ is a well-ordering, hence there exist two states s_p and s_q such that: s_q is reachable from s_p, s_p is reachable from s_0 and $s_p \leqq s_q$. □

We can now prove Theorem 3.2.

Proof of Theorem 3.2. By definition, we have: P has an infinite computation $\Leftrightarrow$ $RT(P)$ is infinite.

By Lemma 3.6, we have $RT(P)$ is infinite $\Leftrightarrow \exists s$, $s' \in RT(P)$ s' is reachable from s and $s \leqq s'$. The following algorithm based on Lemma 3.6 decides whether $RT(P)$ is infinite: Construct $RT(P)$ breadth-first until either $RT(P)$ is complete (and then finite) or two global states s, s' are found such that s' is reachable from s and $s \leqq s'$. □

Let P be a protocol using perfect fifo channels. If now, the fifo channels used by P are not supposed to be perfect, then the new system named P' can be represented by a completely specified protocol constructed from P. To every protocol P, we associate a completely specified protocol denoted P_{CS}.

Definition. Let $P = (M_1, \ldots, M_p, \ldots, c_{ij}, \ldots)$ be a protocol. We denote by P_{CS} its *associated completely specified protocol* which is defined by $P_{CS} = (M'_1, \ldots, M'_p, \ldots, c_{ij}, \ldots)$ where, for every $i = 1, \ldots, p$, we have $M'_i = (S_i, T'_i, h'_i, s_{0i}, L_i)$, $T_i \subseteq T'_i$, and for every transition $t \in T_i$, we have, $h'_i(s, t) = h_i(s, t)$.

For every $i, j = 1, \ldots, p$, $i \neq j$, for every local state $s \in S_i$ and for every message $a \in \mathrm{SEND}_{ji}$, if there does not exist a transition $t \in T_i$ such that $h_i(s, t) = s$ and $L_i(t) = (+, a, j)$, then we create a new transition t' in T'_i, such that $h'_i(s, t') = s$ and $L_i(t') = (+, a, j)$.

Example. We can see that $P'_{1CS} = P_2$.

We can state a sufficient condition for P to terminate.

Proposition 3.7. *If P_{CS} terminates then P terminates.*

Proof. Obvious. □

The converse if false: consider $P'_{1CS} = P_2$. It does not terminate but P'_1 does.

4 Conclusion

We have found a new class of protocols, called completely specified protocols, for which the Termination Problem is decidable. The decidability proof is obtained by stopping the construction of the reachability tree when two comparable states are met. Our algorithm which decides TP terminates because of Higman Lemma.

The *practical interest* of completely specified protocols is to allow the modelling and the detection termination of a few link protocols such as the well-known HDLC or the alternating bit protocol. More precisely, every protocol using non-perfect fifo channels (allowing loss of messages) can be simulated by a completely specified protocol; moreover, it can be automatically constructed from the initial protocol using perfect fifo channels. The theoretical result obtained in his paper meets the empirical proof of protocols [10] in the following way: a practical method to verify the validity of protocols is to empty "old messages" in every fifo channel after a given timeout. This empirical method considers, in fact, that the protocol is completely specified, and we have proved here that the TP is then decidable.

Acknowledgements. I wish to thank Béatrice Bérard, Laure Petrucci and David Massart as well as the three anonymous referees for their careful reading of the different versions of this paper.

References

1. Aggarwal S, Gopinath B: Special issue on tools for computer communication systems. IEEE Trans Softw Eng 14(3) (1988)
2. Bochmann G: Finite state description of communication protocols. Comput Network (2): 361–372 (1978)
3. Bochmann G, Finkel A: Impact of queued interaction on protocol specification and verification. 2nd Int Symp on Interoperable Information Systems (ISIIS '88) Tokyo, Japan (1988).
4. Brand D, Zafiropulo P: On communicating finite-state machines. Research Report, RZ 1053, IBM Zurich Research Laboratory, pp 1–83 (1981)
5. Brand D, Zafiropulo P: On communicating finite-state machines, J ACM 30(2): 323–342 (1983)
6. Brauer W, Reisig W, Rozenberg G: Petri nets: Central models and their properties. Advances in Petri Nets 1986, Part 1, Bad Honnef, Lect Notes Comput Sci vol 254. Springer, Berlin Heidelberg New York 1986
7. Author deleted
8. Choquet A, Finkel A: Simulation of linear fifo nets by Petri nets having a structured set of terminal markings. 8th European Workshop on Applications and theory of Petri nets, Zaragoza, Spain (1987)
9. Chow C, Gouda M, Lam S: A discipline for constructing multiphase communication protocols. ACM Trans Comput Syst 3(4): 315–343 (1985)
10. Favreau J M: Personal communication (1988)
11. Finkel A: Structuration des systèmes de transitions: applications au contrôle du parallélisme par files fifo. Thèse d'Etat, University Paris 11 (1986)
12. Finkel A: A generalization of the procedure of Karp and Miller to well structured transition system. 14th ICALP Karlsruhe, RFA. Ottmann (ed:) LNCS 267: 499–508 (1987)
13. Finkel A: A new class of analyzable CFSM with unbounded fifo channels. 8th International Symposium on Protocol Specification, Testing, and Verification, Atlantic City, New Jersey, USA, IFIP WG6.188 (1988)
14. Finkel A, Rosier L: A survey on decidability results for classes of fifo nets. Advances in Petri Nets 1988. LNCS 340: 106–132 (1988)
15. Author deleted
16. Gouda M: To verify progress for Communicating Finite State Machines. IEEE Trans 10(6): 846–855 (1984)
17. Gouda M, Yu Y: Synthesis of Communicating Finite State Machines with guaranteed progress. IEEE Transactions on Communications 32(7) (1984)
18. Author deleted
19. Gouda M, Gurari E, Lai T, Rosier L: On deadlock detection in systems of communicating finite state machines. Comput Artif Intell 6(3): 209–228 (1987)
20. Higman G: Ordering by divisibility in abstract algebras. Proc Lond Math Soc 2 (1952)
21. Karp R, Miller R: Parallel program schemata. JCSS 4, 147–195 (1969)
22. Koenig D: Theorie der endlichen und unendlichen Graphen. Akademische Verlagsgesellschaft, Leipzig 1936
23. Lam S, Shankar U: Protocol verification via projections. IEEE Transact Softw Eng 10(4) (1984)
24. Lin F, Chu P, Liu M: Protocol verification using reachability analysis: the state space explosion problem and relief strategies. ACM SIGCOMM '87, Frontiers in Computer Communications Technology" Stowe, Vermont vol 17, no 5 (1987)
25. Author deleted
26. Miller R: The construction of self-synchronizing finite state protocols. Distrib Comput 2: 104–112 (1987)
27. Pachl J: Reachability problems for CFSMs. Research Report CS-82-12, University of Waterloo, Dept of Comput Sci (1982)

28. Pachl J: Protocol description and analysis based on a state transition model with channel expressions. Rudin H, West CH (eds) 7th Int Workshop on Protocol Specification, Testing, and Verification Montréal, Québec. IFIP 87, Elsevier Science Publishers B.V. (North Holland), pp 207–219
29. Ramamoorthy C, Yaw Y, Aggarwal R, Song J: Synthesis of two party error recoverable protocols. ACM-SIGCOMM '86 Symposium, Communications Architectures & Protocols, Stowe, Vermont (1986)
30. Author deleted
31. Rosier L, Yen H: Boundedness, empty channel detection, and synchronization for communicating finite automata. Theor Comput Sci 44: 69–105 (1986)
32. Rubin J, West CH: An improved protocol validation technique. Comput Networks 6: 65–73 (1982)
33. Author deleted
34. Sunshine C: Formal modelling of communication protocols. In: Schoemaker (ed) Computer networks and simulation 2. North Holland, 1982
35. Vuong ST, Cowan DD: Reachability analysis of protocols with fifo channels. ACM-SIGCOMM '83 Symposium Communications Architectures and Protocols. University of Texas at Austin, March 8–9. In: Computer Communication Review, vol 13, no 2 (1983)
36. Author deleted
37. Zafiropulo P, AL: Towards analyzing and synthesizing protocols. IEEE Trans Commun 28(4): 651–661 (1980)
38. Zhao Z, Bochmann G: Reduced reachability analysis of communication protocols: a new approach. 6th Int Workshop on Protocol Specification, Testing, and Verification Montréal, Québec. IFIP 7, North Holland (1986)