A Note on Communicating Machines with Identical Symmetrical and Dual processes using Rewriting Systems.

H. MOUNTASSIR

Laboratoire d'Informatique, UFR des Sciences et Techniques
16, route de Gray 25030 Besançon, FRANCE
Email : Mountass@comte.univ-fcomte. fr
Fax : (33) 81.66.64.50

Abstract - This paper states the following result. Replacing a pair of actions (**rcv** x; **send** y) with another pair of actions (**send** y; **rcv** x) in a machine M cannot introduce deadlocks or unspecified receptions when this machine communicates with another machine N. This result is utilized in identifying for any given machine M a large family of machines { N1, N2, ...} such that the communication between M and any Ni is free from deadlocks and unspecified receptions.
Processes exchanged messages through FIFO channels and modelled with finite states machines labelled with actions of receiving and transmitting messages.
Some special classes are discussed with guaranteed progress:
1) Let u be a sequence on a finite alphabet of messages, we propose to construct pairs of machines < M, N1 > with rewrite rules, symmetrical machines < M, N2 > and identical machines < M, M >. In those systems u indicates a cycle of M.
Transformations are defined by commuting pairs (rcv x; send y) into (send y; rcv x). M, N1 and N2 are completely specified by the number of states, the number of cycles and the communication is shown bounded.
2) Let M be a deterministic machine without mixed states, we propose to construct pairs of machines < M, N>. N is obtained from the dual of M using some transformations. The number of states and the number of transitions are the same of the original machine M.
Other systems can be derived from <M,N> by combination. Their number is evaluated by a combinatorial formula. The communication is shown bounded or not according as the original machine contains or not a cycle of receptions or transmissions.

Key words - Rewriting systems, Protocol of Communication, Combinatorial formula, Semi-commutation, Communicating Machines over FIFO channels.

I- Introduction

The model of communicating finite state machines is an abstraction of sequential processes which communicate exclusively by exchanging messages [1, 2, 3, 8,9].

This is the usual approach used to model formally communicating protocols and this important application motivates our work. Our aim is to obtain non-blocking protocols, i.e. protocols that cannot enter into a deadlock state or unspecified reception state. More specially, this paper tackles the following problem- for a given finite automaton M find a finite automaton N assuring non-blocking communication with M. We do not attack the problem in its full generality, we consider rather special cases. Nevertheless, we think that the problem even in those special case is interesting enough to deserve our attention. Moreover, the approach adopted here use a class of semi-commutations which allows to model communication between processes over unbounded FIFO channels is considered.
To this end, the set of sequences that can be obtained from a sequence is given by replacing actions (rcv x; send y) with pair of actions (send y; rcv x). A combinatorial formula is defined to compute the total length. It is interesting by itself in connection with rewriting systems and some specials protocols [10, 11].

Many formal techniques concerning the validation of communicating machines have been proposed and shown promising results. We distinguish three major problems :

. The analysis problem for communicating machines can be defined as follows : give two or more communicating machines, can they reach a state after which one of them cannot progress any further ? The algorithm generates all possible sequences of sending and receiving operations executed by each machine. It is required to decide whether or not their communication is bounded, free of deadlocks and unspecified receptions.
If the channels of the given machines have finite capacities then problem is clearly decidable [1, 2].
. The synthesis problem can be defined as : give one or more machines, it is required to synthesize other machines such that the communication between them is bounded, free of deadlocks and unspecified receptions [8, 9]. In general, we define a set of rules to correct the communicating machines.
. The design problem for communicating machines can be defined as follows : give one machine can another

0-8186-3670-X/93 $03.00 © 1993 IEEE

machine be constructed such that the two machines are guaranteed to progress indefinitely during their communication [3, 4].
The final paper is organized as follows :
In section II, we present the general model of communicating machines. Section III introduces basic definitions on rewriting systems and their connection with communicating systems. In sections IV, V and VI we propose a simple constructing technique of systems with derived, identical and symmetrical processes. The obtained machines are guaranteed to progress indefinitely. Finally, applications on deterministic machines are also given to illustrate our purpose.

II- The Model

The model which consists in representing protocols as two processes exchanging messages over unidirectional FIFO channels has been proved useful in defining and studying many communication protocols. Process traces are represented by finite automata. Among the properties suitable for such systems are the absence of deadlocks and unspecified receptions i.e. where the two processes are waiting for messages from each other while their input channels are empty or contains messages not concerned. Those problems have been shown undecidable. There exists many techniques based on the construction of the reachability graph of the all reachable states.
Consider a communicating system <M,N,C1,C2> where C2 (respect. C1) is a FIFO channel that allows M (respect. N) to send messages to N (respect M).
We denote by A the finite alphabet of actions of exchanged messages between M and N. M and N are defined by automata. Transitions are labelled with actions of sending and receiving messages. States are specified with integers in {0, 1, ...}.

Definitions :
. A global state is said a deadlock if the channels are empty and the two processes are waiting for receiving messages.
. A global state is said an unspecified reception if one of the two processes to progress needs to receive a message and it isn't in its channel.
. A blocking global state is an unspecified reception or a deadlock state.
. We say that the communication is bounded if the sequences of messages in their channels are bounded by an integer, otherwise it is unbounded.
. The communication between machines progresses indefinitely if no blocking states exist during their communication.

III- Preliminaries and Rewriting systems

For convenience, we denote a pair (rcv x ; send y) by (x, y^-). Let R be the set of semi-commutation rules defined by $R = \{(x,y^-)$ such that $x\,y^- \longrightarrow y^-\,x\}$ where x indicates an action of receiving message x and y^- an action of sending message y.
We say that a word u may be rewritten into a word w if one of the following conditions is satisfied :
i) $u = u_1\,x\,y^-\,u_2$, $w = u_1\,y^-\,x\,u_2$ with $x\,y^- \longrightarrow y^-\,x$.
ii) there exists v such that u may be rewritten in v and v may be rewritten in w. We denote that $u \longrightarrow w$.

Example :
Consider $u = a\,b\,a^-\,c\,b^-\,c^-$ and $R=\{(a,a^-), (b,a^-),(a,b^-), (b,b^-), (c,b^-), (a,c^-), (b,c^-), (c,c^-)\}$.
The set of obtainable sequences from u is :
$R(u) = \{ a\,b\,a^-\,c\,b^-\,c^- , a\,a^-\,b\,c\,b^-\,c^- , a^-\,a\,b\,c\,b^-\,c^- , a\,b\,a^-\,b^-\,c\,c^- , a\,a^-\,b\,b^-\,c\,c^- , a\,a^-\,b\,b^-\,c\,c^- , a^-\,a\,b^-\,b\,c\,c^- , a^-\,b^-\,a\,b\,c\,c^- , a\,b\,a^-\,b^-\,c^-\,c , a\,a^-\,b\,b^-\,c^-\,c , a\,a^-\,b^-\,b\,c^-\,c , a\,a^-\,b^-\,c^-\,b\,c , a^-\,a\,b\,b^-\,c^-\,c , a^-\,a\,b^-\,b\,c^-\,c , a^-\,a\,b^-\,c^-\,b\,c , a\,b^-\,a\,b\,c^-\,c , a^-\,b^-\,a\,c^-\,b\,c , a^-\,b^-\,c^-\,a\,b\,c \}$.
The total length is 19.
. Two sequences u and v of M and N are said to communicate perfectly if they can be executed starting from and coming back to empty channels.
One can verify that the two sequences $u=a^-\,b\,c^-\,d$ of M and $v=b^-\,d^-\,a\,c$ of N communicate perfectly. The behaviour u which consists in order of sending a message a and receiving b, and sending message c and receiving d. The system <u,v> communicate without errors and cannot enter into a deadlock state or unspecified reception state.
A computation of the two sequences being defined by the following steps : u sends a message a , thus v sends in the order messages b and d. u receives the message b, sends c and receives from its channel d. v receives respectively the messages a and c. All transmitted messages are received by the two special processes u and v and their channels are empty.

We denote by f the morphism defined from $A^* \longrightarrow A^*$ by $f(m)=m^-$, $f(m^-)=m$ and $f(E)=E$ where E indicates the empty sequence. f associates any sequence with its opposite.
Using those definitions we briefly recall the central results of communicating words in [6].
. Two sequences u and v in A^* communicate perfectly over two FIFO channels if and only if $f(u) \longrightarrow v$.
For example for $u= a^-\,b\,c^-\,d$ and $v=b^-\,d^-\,a\,c$, we then have :
$f(u)= a\,b^-\,c\,d^- \longrightarrow b^-\,a\,c\,d^- \longrightarrow b^-\,a\,d^-\,c \longrightarrow b^-\,d^-\,a\,c = v$ by applying rules on $R = \{(a,b^-), (c,d^-), (a,d^-)\}$.

. u and v have the same length.
. If u and v communicate perfectly then the set of

sequences R(u) communicate perfectly with the set of sequences R(v).

. In particular we have <u , f(u)> communicate perfectly.

. u —> v implies that $f(v)$ —> $f(u)$.

Definition :
Let u be a sequence in A*. Let $k_1, k_2, ..., k_n$ be n integers such that $1 \leq n$ and $1 \leq k_1 < k_2 < ... < k_n$.
We denote the total length by the quantity $Z_n[k_1, k_2, ..., k_n]$ where k_i specifies the respective positions of sending actions in u.
The total number of obtainable sequencess from u by rules of type $x\, y^-$ —>y^-x where x indicates a reception and y^- a transmission is given by $Z_n[k_1, k_2, ..., k_n]$.

Assume that :

$$Z_n[k_1, k_2, ..., k_n] = \sum_{\alpha_1=1}^{k_1} \sum_{\alpha_2=1+\alpha_1}^{k_2} \cdots \sum_{\alpha_n=1+\alpha_{n-1}} \quad (1)$$

In [7], we establish that the number is expressed exactly as a determinant of a matrix :

Theorem :
$Z_n[k_1, ..., k_n]$ = Determinant of $\{C_{k_j-j+1}^{i-j+1}\}_{1\leq i,j\leq n}$.
The binomial coefficient $C_i^j = 0$ if i > j . i and j indicate respectively lines and columns of matrix.

Consider an alphabet of actions with receiving and sending messages A = {a, b, c, d, a^-, b^-, c^-, d^- }.
For u = a b a^- c b^- c^- we obtain $Z_3[3, 5, 6]$= 19.
The quantity Z_n computes the cardinal of R(u). In particular, the number of communicating sequences with u is given by the cardinal of R(f(u)).

IV- Rewriting Machines

In this section, we discuss how to construct a pair of machines which ensure that the communication between them progresses indefinitely.
The processes have distincts automaton with cycles passing through in their initial states.
From the above results expressed in section III, we deduce the next results :

. If < u, v> communicate perfectly then the following systems < u , R(v)> , <R(u), R(v)>, < R(u), R(f(u)) >, < u , R(f(u))> communicate also.

Algorithm of the Construction :
A smple technique is considered to construct M. N1 is given by the same steeps using v instead of u.
Assume that u such that < u , v > communicate perfectly.
From u which represent a cycle of machine M. It is defined by u* passing from and come back to its initial state 0.
Let u be a sequence such that $u = u_1\, x\, y^-\, u_2$, where x and y^- indicate respectively transitions (q_{i-1}, x, q_i) and (q_i, y^-, q_{i+1}). To rewrite u into $w = u_1\, y^-\, x\, u_2$ where u_1 and u_2 are the same. We then have an intermediate state denoted by q'i and two transitions denoted respectively by (q_{i-1}, y^-, q'_i) and (q'_i, x, q_{i+1}). We repeat this opeartion until no pairs of actions can be exchanged. The total number ob obtainable sequence from u is finite and also the algorithm terminates.

Theorem : The communication of < M , N1> progresses indefinitely. Thus, it is bounded.

. The number of states
Let u in A* with n sending actions and k_i their respective positions in u. The total number of states in M is computed by :
length(u)+ $\sum_{1\leq i\leq n} k_i$ -i.
We use v to compute he number of states in N1.

. The number of Cycles composed M passing in their initial states is evaluated by the number of R(u) and it is equal to $Z_n[k_1, k_2, ..., k_n]$ where k_i specify positions of sending actions in u. Cycles of N1 are obtained from a sequence v .

. Capacities of Channels are obtained for maximal behaviours where in order transmitting n messages. This calculus is defined by the last obtainable sequences to complete the final machines M and N1. We then have 2n and 2* (length(u)-n) messages in their respective channels.

Application
To illustrate our ideas, we consider an example with
u = a^- b^- d c^- e and v = d^- a b e^- c.

. The number of cycles composed M is $Z_3[1,2,4] = 2$, the number of cycles in N1 is $Z_2[1,24] = 3$ and are defined by the following sets :
. R (u) = { a^- b^- d c^- e , a^- b^- c^- d e }.
. R (v) = { d^- a b e^- c , d^- a e^- b c , d^- e^- a b c }.

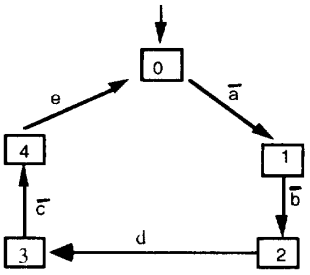
. The number of states of the final machine M is computed by 5 + 1-1 + 2-2 + 4-3=6.
The number of states in N1 is 5 + 1-1 + 4-2 =7.

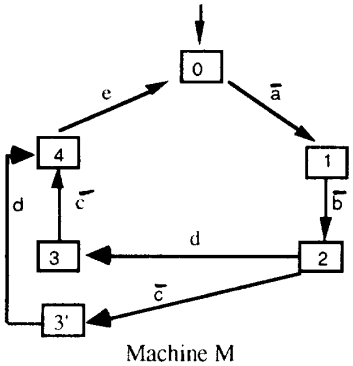
. The capacities of the two channels are 6 and 4 and

obtained by the following sequences < $a^- b^- c^- d\ e\ a^- b^- c^- d\ e$, $d^- e^-\ a\ b\ cd^- e^-\ a\ b\ c$> where each of them is executed by M and N1.

Let k_1 be the first position of sending message in u. This message moves at k_1-1 and creates an intermediate state denoted by 1'. It removes until reached the initial state 0. We repeat the same operation of letter in k_2 position until reaches some sending message. The construction terminates with the last message in position k_n .

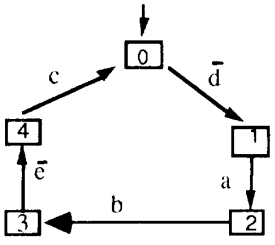
Final machines are obtained by the following transformations. At each steep we create an intermediate state between tow given transitions.
From u we obtain M :

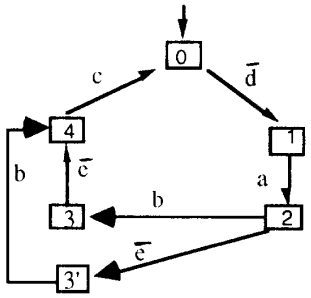


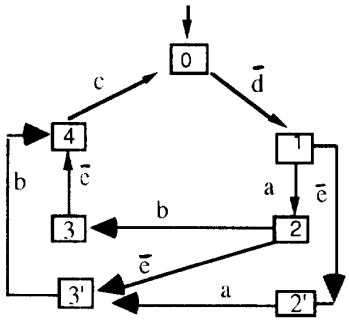


Machine M

From v we obtain N1 :







Machine N1

V- Symmetrical Machines

In this section, we discuss how to construct a pair of symmetrical processes such that the communication between them progresses indefinitely.
The processes have the same traces of sendings and receiving actions. Cycles are passing through their initial states.

Definition :
u and s are said symmetrics if and only have the same traces on receiving and sending actions.

Remark: If s exits and communicate perfectly with u then s can be obtained from $f(u)$ and have even length.

For example :
$u = a^- b^- d\ c^- e\ f$ and its opposite $v = a\ b\ d^- c\ e^- f^-$ are not symmetrics but u and $s = d^- e^-\ a\ f^-\ b\ c$ are symmetrics.

Theorem : The communication of < M , N2 > progresses indefinitely and it is bounded.

M is obtained from u and N2 from s. In this case the number of states and transitions are evaluated by the same construction like section IV.

Application
. The number of cycles composed M and N2 are given by

$Z_3[1, 2, 4] = 2$ and are defined by the following sets :

R (u) = { $a^- b^- d c^- e f$, $a^- b^- c^- d e f$ }.

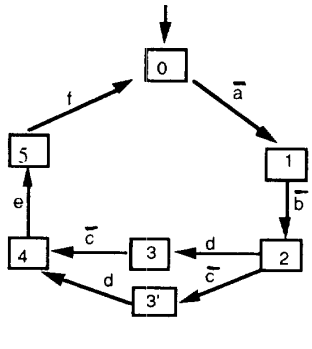
R (v) = { $d^- e^- a f^- b c$, $d^- e^- f^- a b c$ }

. The number of states of the final machine M is computed by $n(3-n)/2 + \sum_{1 \le i \le n} k_i$. It is equal to 7 for M and N2.

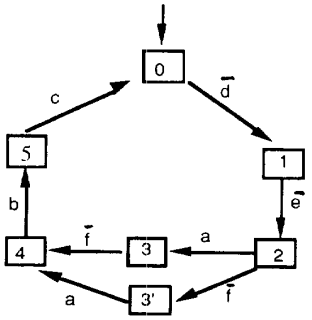
. The capacity of channel is 6, and obtained by the following behaviours <$a^- b^- c^- d e f$, $a^- b^- c^- d e f$, $d^- e^- f^-$. a b c $d^- e^- f^-$ a bc> where each of them is executed by M and N2.

Let k_1 be the first position of sending message in u. This message moves at k_1-1 and creates an intermediate state denoted by 1'. It removes until reached the initial state 0. We repeat the same operation of letter with k_2 position until reaches some sending message. The construction terminates with the last message in position k_n .

M and N2 are:



Machine M



Machine N2

VI- Identical Machines

This section discusses a construction of a pair of identical processes with garanteed progress.

The processes have the same automata with cycles passing through their initial states.

Remark: The process M contains u and necessary we have need that <u , u> communicate perfectly.

We deduce from the above the following results :

. The system <u,u> communicate perfectly if $f(u) \rightarrow u$.

. If < u, u> communicate perfectly then u has even length.

A method to construct M is similar. Assume that u such that < u , u > communicate perfectly.

From $f(u)^*$ which represent the first machine M passing from and come back to its initial state 0.

Let v be a behaviour such that $v = v_1 x y^- v_2$, firstly we have $v = f(u)$ where x and y^- indicate respectively transitions (q_{i-1}, x, q_i) and (q_i, y^-, q_{i+1}). Cycles can be obtained by derivation.

Theorem : The communication of < M , M > progresses indefinitely. Thus, it is bounded.

.An application

To illustrate our ideas we consider an example with u = $a^- b^- a c^- b c$ and its opposite $f(u) = a b a^- c b^- c^-$.

. The number of cycles composed M is $Z_3[3, 5, 6] = 19$ and are defined by the following set :

R ($f(u)$) = { $a b a^- c b^- c^-$, $a a^- b c b^- c^-$, $a^- a b c b^- c^-$, $a a^- b^- c c^-$, $a a^- b b^- c c^-$, $a a^- b^- b c c^-$, $a^- a b b^- c c^-$, $a^- a b^- b c c^-$, $a^- b^- a b c c^-$, $a b a^- b^- c^- c$, $a a^- b b^- c^- c$, $a a^- b^- b c^- c$, $a a^- b^- c^- b c$, $a^- a b b^- c^- c$, $a^- a b^- b c^- c$, $a^- a b^- c^- b c$, $a b^- a b c^- c$, $a^- b^- a c^- b c$, $a^- b^- c^- a b c$ }.

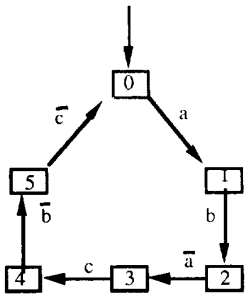
. The number of states of the final machine M is computed by $n(3-n)/2 + \sum_{1 \le i \le n} k_i$. We then have 3((3-3)/2 + 3 + 5 + 6=14.

. The capacity of channel is 6, and obtained by the following behaviours < $a^- b^- c^- a b c$, $a^- b^- c^- a b c$, $a^- b^- c^-$. a b c $a^- b^- c^-$ a b c> where each of them is executed by M.

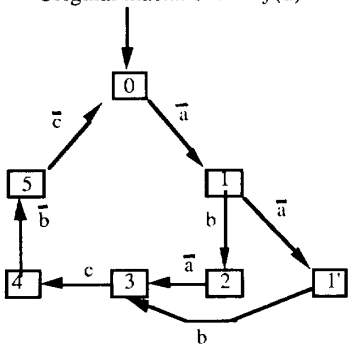
Let k_1 be the first position of sending message in $f(u)$. This message moves at k_1-1 and creates an intermediate state denoted by 1'. It removes until reached the initial state 0.

We repeat the same operation of letter with k_2 position until reaches some sending message. The construction terminates with the last message in position k_n .

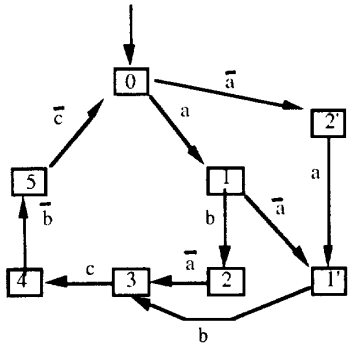
M is obtained by the following steeps :



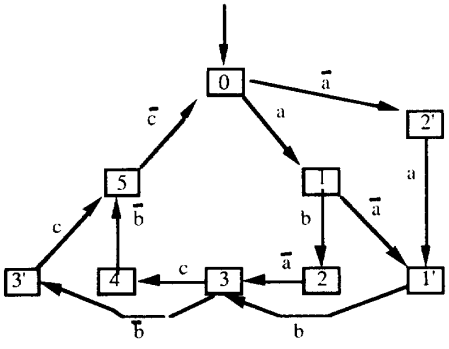
Original machine M =f(u)*



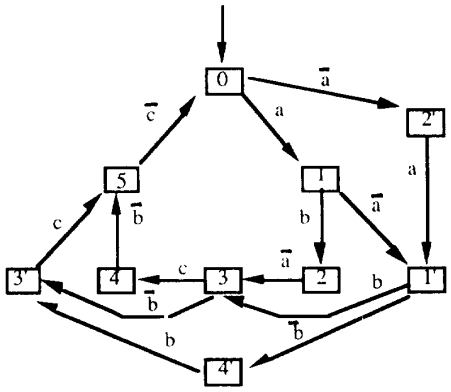
We rewrite b a⁻ —> a⁻ b , thus we create an intermediate state denoted 1'.



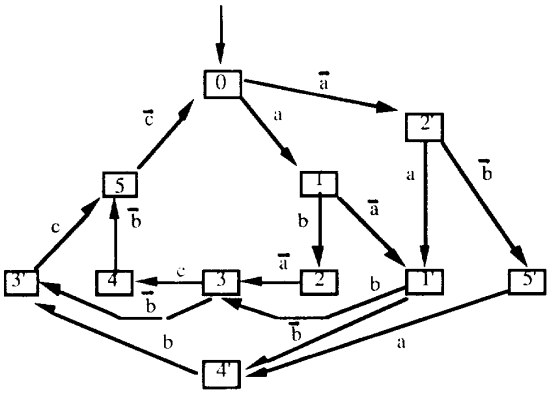
We rewrite a a⁻ —> a⁻ a, we then create an other state with number 2'.



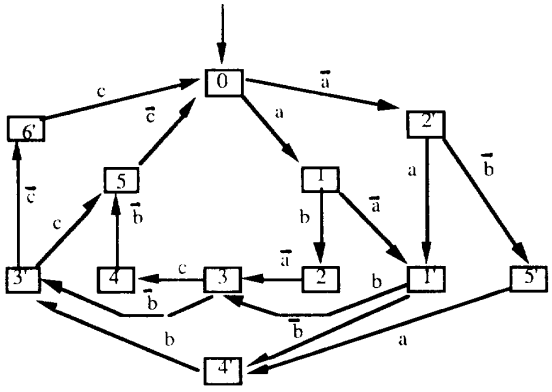
We use c b⁻ —> b⁻ c, we then create a state 3'.



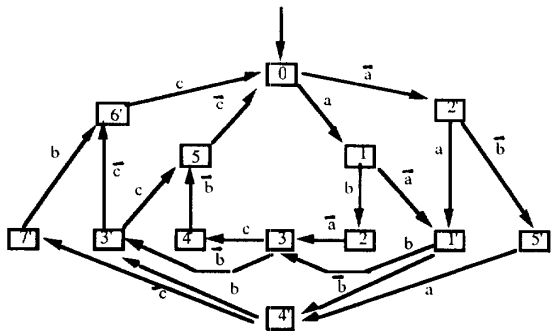
We rewrite b b⁻ —> b⁻ b, we then create an intermediate state with 4'.



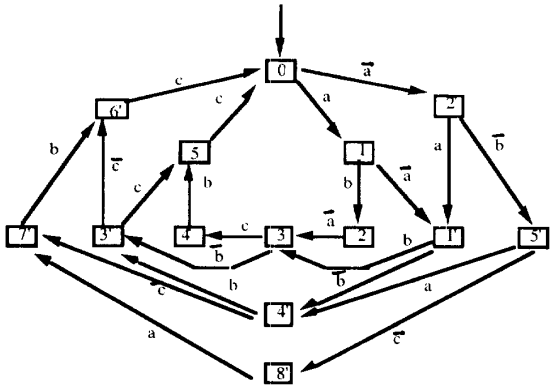
We use a b⁻ —> b⁻ a to create a state number 5'.



We use c c⁻ —> c⁻ c, we create a state 6'.



We use b c^- —> c^- b to create a state 7'.



The final machine M

Finally with a c^- —> c^- a, we create the last state denoted 8'. The communication of <M,M> progresses indefinitely.

VII- Deterministic Machines

In this section we propose to construct communicating machines from a given one. For convenience, we suppose that each state has at least an output transition.

Definition :

. A deterministic machine is defined as a machine which each state has no two output transitions with the same labels.

. We consider that the machine M has only transitions labelled with sending or receiving actions.

. We said the dual machine f(M) of M by replacing each action of sending (respectively receiving) message in M by its opposite.

R(M) denoted the machine obtained from M by rewriting some behaviours of M which we define furtherly the rules .

From each state denoted by an integer i, we denote u_{ij} the behaviours from i and defined by concatenation of all transitions prefixed by the first letters.

We apply the two rules :

. Let i be the state, if we have only one transition denoted u_{i1} then we rewrite u_{i1} globally,

. Let i be a state with r transitions prefixed by the first letters t_{i1}, t_{i2}, ... t_{ir} the behaviours are defined by $t_{i1}.u_{i1}$, $t_{i2}.u_{i2}$,.... $t_{ir}.u_{ir}$, in this case we rewrite only u_{i1}, u_{i2},.... u_{ir} .

We have shown the following result in [5] :

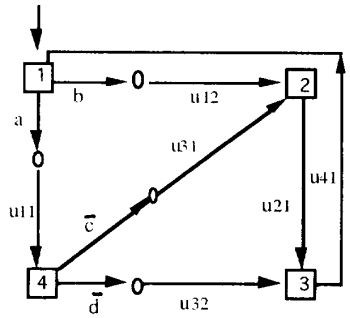
Theorem : If M is deterministic and without mixed states then the communication between machines <M, f(M)>, <M, R(f(M))>, <R(M), R(f(M))> progresses indefinitely.

If M contains a cycle of sending or receiving actions then the communication is unbounded, otherwise is bounded.

Application

The main idea is to construct pairs of machines from a given one without developing the reachability tree.

Let M be the original machine satisfied the above three conditions :



The original machine M is deterministic and without mixed states.

For example we define the following sequences :

u_{11} = b^- c, u_{12} = c^- d e, u_{21}= -a^- b c^- d, u_{31} = a^- b^- c, u_{32} = a^- b, u_{41} = a^- b^- c d

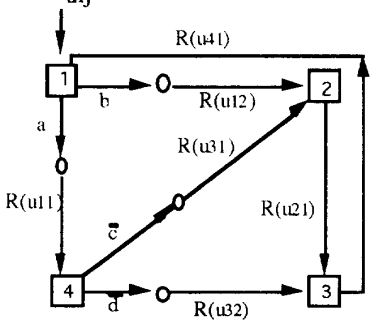
. The dual machines $M_{f(ij)}$ are obtained by rewriting $f(u_{ij})$, their number are defined by $Z_{f(uij)}$.

We have $Z_{f(u11)}$ = Z_1[2] = 2 , $Z_{f(u12)}$= Z_2[2,3] = 3 , $Z_{f(u21)}$ = Z_2[2,4] = 5, $Z_{f(u31)}$ = Z_1[3]=3 , $Z_{f(u\ 32)}$ = Z_1[2] =2 et $Z_{f(u41)}$= Z_1[3,4] = 6.

The total number is the product of $Z_{f(uij)}$ for each word indicated by $\pi_{i,j} Z_{f(uij)}$. In this case we obtain 1080 systems < M , R($M_{f(uij)}$)> .

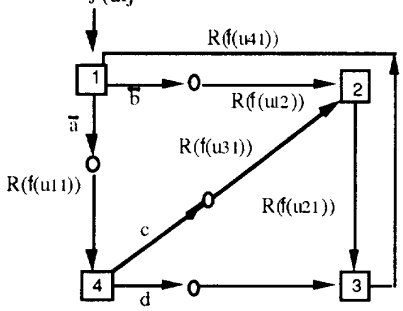
To generate other systems, we rewrite u_{ij} (their number is given by Z_{uij}) of M, we obtain machines M_{uij} :

$Z_{u11} = Z_1[1] = 1$, $Z_{u12} = Z_1[1] = 1$, $Z_{u21} = Z_2[1,3] = 2$, $Z_{u31} = Z_2[1,2] = 1$,
$Z_{u32} = Z_1[1] = 1$ et $Z_{u41} = Z_2[1,2] = 2$.
The total number is $\pi_{i,j} Z_{uij} * Z_{f(uij)}$. In this case we have 4320 systems.
Machines $R(M_{uij})$ are defined as :



Machines obtained from M

. Machines $R(M_{f(uij)})$ are defined as :



Machines obtained from f(M)

Finally, from M we construct a whole family < M, $R(M_{f(uij)})$>, < $R(M_{uij})$, $R(M_{f(uij)})$ > and their communication progresses indefinitely.

General remark

For each pair <u,v> we can construct multitude systems with a combination of systems mentionned above. Sometimes we have choosen to rewrite sequences u, v or f(u) in the goal to obtain maximal machines with an important number of cycles. It is clear that any restriction to machines with subsets preserve the same properties on progress and boundedness.

Conclusions

In this paper, we attempt to use another approach to validate the communication between machines without developing the reachability graph. Rules are defined by a set of semi-commutations. The main idea is to replace a pair (rcv x ; send y) with another pair (send y ; rcv x) where x and y are messages. We have shown that we can construct pairs of machines from a given pair sequences <u , v > whether their communication will progress indefinitely. Others systems can be easily derived. The final machines M and N are completely specified with the number of states, the composed cycles and the communication is shown bounded or not according as the original machine contains a cycle of sending actions. Others systems can be easily derived from the obtained systems. Their total number is enumerated by a formula based on the positions of some actions. The extension of this technique to systems with more than two processes is still an open question.

References

[1] G. V. BOCHMANN, " Finite state description of communication protocols", Comput. Networks 2, 361-371, 1978.

[2] D. BRAND and P. ZAFIROPULO, " On communicating finite-state machines", J. Assoc. Comput. Mach. vol 30, N° 2, 323-342, 1983.

[3] M. GOUDA and C. CHANG, " A technique for proving liveness of communicating finite state machines with examples ", 3 th ACM symp. on Principles of Distributed Computing, august, 1984.

[4] M. GOUDA and Y. YU, " Synthesis of communicating machines with guaranted progress ", IEEE trans. Comm. COM-32, N° 7, 846-855, 1984.

[5] H. MOUNTASSIR , "Sur la Progression de la Communication entre Processus Communicants", Rap. de Recherche N° 12, 1989.

[6] R. CASTANET, H. MOUNTASSIR et R. SIJELMASSI, "Mots communicants et leurs applications". Computers and Computing, Grenoble, December 1985..

[7] H. MOUNTASSIR , "A combinatorial Formula to compute chains and its connection with communicating systems", Rap. de Recherche, soumis à bublication.

[8] P. SIDHU , " Rules of Synthesizing correct communication protocols", ACM Comp. Com. Rev. , Vol 12, N°1, 1982.

[9] ZAFIROPULO et al., " Towards analyzing and synthesizing protocols ", IEEE trans. Comm. COM-28, N° 4, 651-661, 1980.

[10] S. HADDAD et al., " A Reduced state graph for symmetrical protocol", Rap. MASI, September 1992.

[11] P. SIDHU, " Authentification Protocols for general communication channels", Proc. 6th Conf. Comp. Net., Mineapolis, 1981.