SHARPENING BOUNDS ON THE TIME BETWEEN EVENTS IN MAXIMALLY PARALLEL SYSTEMS

George S. Avrunin

CMPSCI Technical Report 92–69
September 1992

Department of Computer Science
University of Massachusetts
Amherst, Massachusetts 01003

This research was partially supported by Office of Naval Research Grant N00014-89-J-1064 and National Science Foundation Grant CCR-9106645.

SHARPENING BOUNDS ON THE TIME BETWEEN EVENTS IN MAXIMALLY PARALLEL SYSTEMS

GEORGE S. AVRUNIN

1. Introduction

A recent paper [3] describes a method for obtaining bounds on the time that can elapse between two given events in an execution of a concurrent software system running on a single processor under arbitrary scheduling. The technique involves generating linear inequalities expressing conditions that must be satisfied by all executions of such a system and using integer programming methods to find appropriate solutions to the inequalities. Corbett [4, 5] has extended this approach to obtain upper bounds on the time between events in executions of multi-processor concurrent systems in which each process proceeds unless forced to wait to communicate with another, the case of *maximal parallelism*, and processes communicate by synchronous message passing. Corbett's method does not strictly enforce the maximal parallelism assumption, however, and may thus give poor (though valid) bounds in some cases. In this paper, I show how to modify Corbett's method to obtain sharper bounds.

2. Corbett's method

I will begin by briefly reviewing Corbett's method, and will assume that the reader is familiar with the uniprocessor method of [3]. A system of concurrent processes communicating by synchronous message-passing over named channels is modeled by a collection of deterministic finite state automata (DFAs), possibly together with a collection of recursive *restriction languages*. A string over the union of the alphabets of the DFAs corresponds to a trace of an execution of the concurrent system if its projection on the alphabet of each DFA lies in the language accepted by the automaton, and its projection on the alphabet of each restriction language lies in that language. (See [1] or [2] for a more complete description of this formalism.) I will adopt the convention of [3] and assume that each symbol representing a communication event belongs to the alphabet of exactly two DFAs and that each symbol representing an internal computation belongs to the alphabet of exactly one DFA. This can easily be achieved by encoding process and channel names in the symbols.

This research was partially supported by Office of Naval Research Grant N00014-89-J-1064 and National Science Foundation Grant CCR-9106645.

Corbett's technique can be viewed as an extension of the standard method of critical path analysis to the case in which the underlying partial order varies, due to the nondeterminacy of concurrent computation. Essentially, he generates a system of inequalities representing necessary conditions satisfied by each execution of the system, a separate system of inequalities finding a critical path, and additional inequalities constraining the critical path to lie along some execution.

Corbett first generates a set of flow equations from the process DFAs, with implicit flow of 1 into the start state of each DFA and a flow of 1 out of one of the accepting states of each DFA. Further inequalities are added, as in [1], to express the fact that the same communication event occurs simultaneously in different processes or conditions imposed by the restriction languages. These inequalities, the *execution part* of Corbett's system of inequalities, express necessary conditions for the existence of a finite trace of the concurrent system.

He then adds arcs connecting states in different DFAs. For each pair of syntactically matching communication events in different DFAs, he adds a pair of *cross arcs* connecting the state in one DFA preceding the communication with the state in the other DFA following it. Introducing new variables at the start and accepting states of the DFAs, he generates a system of flow equations for this augmented graph with total flow in of one through the new variables at start states and total flow out of one through the new variables at accepting states. These inequalities, the *critical path part* of Corbett's system, find a *wait path* through the augmented graph, beginning at the start state of some process DFA and ending at an accepting state of some, possibly different, process DFA.

Corbett then introduces inequalities for each arc in the process DFAs and for each cross arc in the augmented graph. For each arc in the process DFAs, he requires that the value of the variable associated with that arc in the critical path part be no greater than the value of the variable associated with that arc in the execution part. For each cross arc, he generates a pair of inequalities requiring that the value of the variable associated with that arc be no greater than the values of the variables associated with the corresponding communication events in the process DFAs in the first system of inequalities. Finally, he adds inequalities bounding the sum of the variables associated with communication or cross arcs in the augmented graph that represent a single communication by variables corresponding to the associated communication in the first set of inequalities. Corbett calls these inequalities the *bounding part* of the system of inequalities.

This technique is illustrated with the simple system shown in Figure 1. In this example, two customer processes, Process 1 and Process 3, each may use a resource represented by Process 2. The events c and d are internal to the second customer process, while events a and b represent use of the resource by the first and second customers, respectively. The cross arcs introduced by Corbett's method are shown with dotted lines. States and arcs are numbered for reference.

The system of inequalities generated for this example is shown in Figure 2. The variable x_i is associated with arc i in the execution part, while y_i corresponds to arc i in the critical path part. The variables b_1, b_3, and b_6 are the new variables

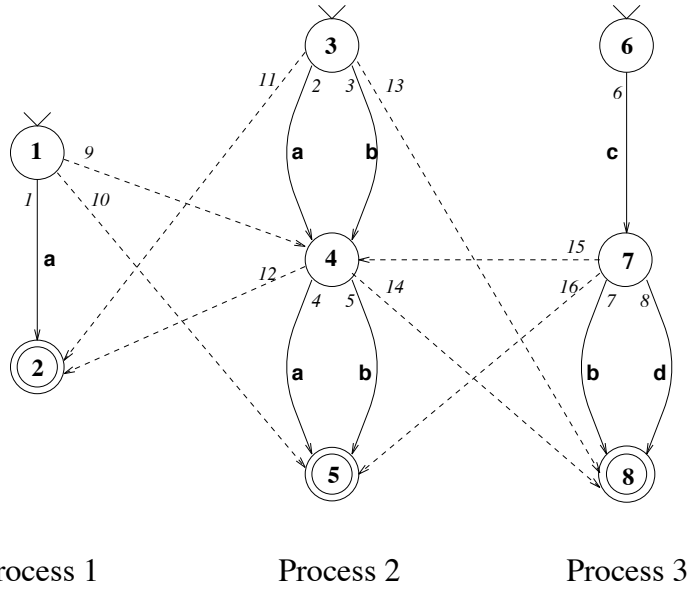

FIGURE 1. Simple resource contention example

introduced at the start states of the process DFAs, and e_2, e_5, and e_8 are the variables introduced at the accepting states.

Suppose we are given the duration of each possible event in the system. If we set d_i to be the duration of the event labeling arc i in the wait graph (cross arcs are labeled with the corresponding communication event), the maximum value of the function $\sum_i d_i y_i$ over solutions to the full system of inequalities is an upper bound on the duration of an execution of the system. As described in [5], this technique can easily be extended to find an upper bound on the time between two specific events by generating an inequality system whose execution part finds segments of an execution beginning and ending with the given events, as in [3]. Additional b_i and e_i variables would allow the path found by the critical path portion of the inequality system to begin in any state in which a process could be after the first given event and end in any state in which a process could be after the second given event.

The bound obtained in this way may not be sharp. The inequalities represent necessary conditions that must be satisfied by any execution but, in general, there will be solutions that do not correspond to executions. For instance, cycles in the wait graph arising from cycles in the process DFAs or repeated communications may lead to cyclic flows in the solution that are not part of an actual critical path. Furthermore, the inequalities do not strictly enforce the maximal parallelism assumption and allow solutions corresponding to executions in which processes that could proceed do not.

This failure to enforce the maximal parallelism assumption can be seen in the example of Figures 1 and 2. Suppose that each event in the system has duration 1. The upper bound on the duration of an execution obtained from the system of

Execution part:

$$x_1 = 1$$
$$x_2 + x_3 = 1$$
$$x_2 + x_3 - x_4 - x_5 = 0$$
$$x_4 + x_5 = 1$$
$$x_6 = 1$$
$$x_6 - x_7 - x_8 = 0$$
$$x_7 + x_8 = 1$$
$$x_1 - x_2 - x_4 = 0$$
$$x_7 - x_3 - x_5 = 0$$

Critical path part:

$$b_1 - y_1 - y_9 - y_{10} = 0$$
$$y_1 + y_{11} + y_{12} - e_2 = 0$$
$$b_3 - y_2 - y_3 - y_{11} - y_{13} = 0$$
$$y_2 + y_3 + y_9 + y_{15} - y_4 - y_5 - y_{12} - y_{14} = 0$$
$$y_4 + y_5 + y_{10} + y_{16} - e_5 = 0$$
$$b_6 - y_6 = 0$$
$$y_6 - y_7 - y_8 - y_{15} - y_{16} = 0$$
$$y_7 + y_8 + y_{13} + y_{14} - e_7 8 = 0$$
$$b_1 + b_3 + b_6 = 1$$

Bounding part:

$$y_1 \le x_1 \quad y_7 \le x_7 \quad y_{11} \le x_1 \quad y_{14} \le x_5$$
$$y_2 \le x_2 \quad y_8 \le x_8 \quad y_{11} \le x_2 \quad y_{14} \le x_7$$
$$y_3 \le x_3 \quad y_9 \le x_1 \quad y_{12} \le x_1 \quad y_{15} \le x_3$$
$$y_4 \le x_4 \quad y_9 \le x_2 \quad y_{12} \le x_4 \quad y_{15} \le x_7$$
$$y_5 \le x_5 \quad y_{10} \le x_1 \quad y_{13} \le x_3 \quad y_{16} \le x_5$$
$$y_6 \le x_6 \quad y_{10} \le x_4 \quad y_{13} \le x_7 \quad y_{16} \le x_7$$

$$y_1 + y_2 + y_4 + y_9 + y_{10} + y_{11} + y_{12} \le x_1$$
$$y_3 + y_5 + y_7 + y_{13} + y_{14} + y_{15} + y_{16} \le x_7$$

FIGURE 2. Corbett's inequalities for system in Figure 1

inequalities is 3 and it occurs with the solution in which y_6, y_{15}, and y_4 are 1 and the other y_i are 0. This corresponds to an execution in which Process 3 performs the internal computation represented by c, then uses the resource (as represented by the communication event b, and finally Process 1 uses the resource (as represented by the communication event a). In this execution, the process representing the resource waits to communicate with Process 3 first, even though it could communicate with Process 1 immediately. This violates the assumption that processes proceed with their computations unless blocked.

3. Sharpening the bounds

In this section I describe a method that sharpens the bound obtained by Corbett's method in cases like the example described in the previous section by excluding solutions to the inequalities that correspond to executions in which processes wait unnecessarily. The method is limited to systems in which the process DFAs contain no cycles. I first describe the basic ideas, and then give the details.

3.1. Basic ideas. The upper bound found by Corbett's method for the system of Figure 1 corresponds to an execution in which Process 2 waits in state 3 to communicate with Process 3, although it could communicate with Process 1 immediately. In order to exclude solutions to the system of inequalities corresponding to such executions, I need to be able to determine when a process is waiting for a communication in a particular state and then add additional inequalities that prevent such waiting when the process can proceed.

Suppose that Process 2 enters state 3 at time s_3 and leaves it at time t_3. Then it is waiting in state 3 during the interval from s_3 to t_3. Let s_1 and t_1 be the corresponding times for state 1. To ensure that Process 2 does not wait in state 3 when it could communicate with Process 1, I need to ensure that either Process 2 enters state 3 after Process 1 leaves state 1 or Process 1 enters state 1 after Process 2 leaves state 3. Equivalently, I want to avoid cases in which the open intervals (s_1, t_1) and (s_3, t_3) overlap. (I allow the cases in which one process's arrival in a state occurs at the same instant as the other's departure from the corresponding state. Thus, for example, Process 1 could enter state 1 at the same time as Process 2 leaves state 3.)

This can be achieved with the quadratic inequality

$$(s_1 - t_3)(s_3 - t_1) \leq 0 \tag{1}$$

In general, however, integer programming with nonlinear constraints is much more difficult than when all constraints are linear. If I impose an upper bound B on the times at which the processes could enter or leave the states, I can achieve the same results using linear inequalities and additional variables, as described below. A safe upper bound can be calculated easily for acyclic DFAs, for instance, by summing the durations of all possible events.

Note that it is impossible for both factors of the left side of (1) to be positive. If $s_1 - t_3$ is positive, Process 1 entered state 1 after Process 2 left state 3. It follows

that Process 2 entered state 3 before Process 1 left state 1, so that $s_3 - t_1$ is negative. The case I need to exclude is the one in which both factors are negative. I therefore introduce new variables that indicate when each of the factors is negative, and use these variables to enforce (1). Let $w_{1,3}$ and $w_{3,1}$, be 0-1 variables. The inequalities

$$t_3 - s_1 \leq B w_{1,3} \qquad s_1 - t_3 < (B+1)(1 - w_{1,3}) \tag{2}$$

$$t_1 - s_3 \leq B w_{3,1} \qquad s_3 - t_1 < (B+1)(1 - w_{3,1}) \tag{3}$$

force $w_{i,j}$ to be 1 exactly when $s_i - t_j$ is negative. The linear inequality

$$w_{1,3} + w_{3,1} \leq 1 \tag{4}$$

then has the same effect as the quadratic inequality (1).

Of course, I have to make sure that the times at which states are entered and exited are consistent, given the durations of events and the synchronous nature of communication. The first set of consistency requirements simply requires that, if a process changes from state i to state j along an arc labeled by an event α with duration d_α, it enter state j d_α units of time after it leaves state i. The consistency requirements reflecting the synchronization between processes due to communication are somewhat more complicated.

Again consider the system of Figure 1, and suppose that Process 1 communicates with Process 2 during an execution. Then the variable x_1 in the execution part of the system of inequalities of Figure 2 must take the value 1 in the corresponding solution to those inequalities, and exactly one of x_2 and x_4 must also be 1. If x_2 is 1, the communication with Process 2 occurred when that process was in state 3, and it must be the case that Process 1 left state 1 at the same time as Process 2 left state 3 and that Process 1 entered state 2 at the same time as Process 2 entered state 4. If x_4 is 1, the communication occurred when Process 2 was in state 4. Then Process 1 must have left state 1 at the same time as Process 2 left state 4 and Process 1 must have entered state 2 at the same time as Process 2 entered state 5.

Let $z_{1,3}$ and $z_{1,4}$ be 0-1 variables. The inequalities

$$z_{1,3} \leq x_1 \tag{5}$$

$$z_{1,3} \leq x_2 \tag{6}$$

$$z_{1,4} \leq x_1 \tag{7}$$

$$z_{1,4} \leq x_4 \tag{8}$$

allow $z_{1,3}$ to be 1 only if the communication between Process 1 and Process 2 occurs when the processes are in states 1 and 3, respectively, and allow $z_{1,4}$ to be 1 only if the communication occurs when the processes are in states 1 and 4. The equation

$$z_{1,3} + z_{1,4} = x_1 \tag{9}$$

forces one of these two variables to be 1 if the communication occurs at all, so $z_{1,3}$ is 1 if and only if the communication occurs with Process 1 in state 1 and Process 2 in state 3, while $z_{1,4}$ is 1 if and only if the communication occurs with Process 1 in state 1 and Process 2 in state 4.

I can now ensure that the processes leave the appropriate states at the same time. The quadratic inequalities

$$z_{1,3}(t_1 - t_3) = 0 \tag{10}$$

$$z_{1,4}(t_1 - t_4) = 0 \tag{11}$$

enforce these restrictions, where t_1 and t_3 are as before and t_4 is the time that Process 2 leaves state 4. Again assuming that the times are bounded above by B, I can achieve the same result with the linear inequalities

$$t_1 - t_3 \leq B(1 - z_{1,3}) \tag{12}$$

$$t_3 - t_1 \leq B(1 - z_{1,3}) \tag{13}$$

$$t_1 - t_4 \leq B(1 - z_{1,4}) \tag{14}$$

$$t_4 - t_1 \leq B(1 - z_{1,4}) \tag{15}$$

Finally, I need to restrict the critical path variables. I want to be sure that one of the cross arcs corresponding to a communication can be part of the critical path only when that communication actually occurs in the execution. I can achieve this using the inequalities

$$y_9 + y_{11} \leq z_{1,3} \tag{16}$$

$$y_{10} + y_{12} \leq z_{1,4} \tag{17}$$

Inequalities of these types can be used to eliminate solutions corresponding to executions in which processes wait when they should proceed, although a slightly more general form is required to ensure that restrictions on waiting are imposed only for states that are actually reached in a particular execution and to deal with the fact that a process should not wait in a state in which it can perform an internal computation. These complications are described in the next subsection.

3.2. The method. Consider a maximally parallel concurrent system modeled by a collection of DFAs, as described in Section 1, and assume that there are no cycles in the DFAs. In this subsection, I describe the generation of a collection of inequalities that can be used to sharpen Corbett's bounds by excluding solutions to his inequalities that correspond to executions in which processes wait unnecessarily or to certain cyclic flows in the wait graph.

For each state i in a process DFA, let s_i be a variable representing the time at which the process enters the state and let B be an upper bound on the permitted values of the s_i. For each DFA, generate an equation

$$s_i = 0 \tag{18}$$

setting the entry time of the start state of that DFA to 0.

For each communication event a, let $M(a)$ be the set of unordered pairs $\{i, j\}$ such that

- i and j are states in different processes,
- there is an arc with label a in the DFA containing i from i to a state i', and

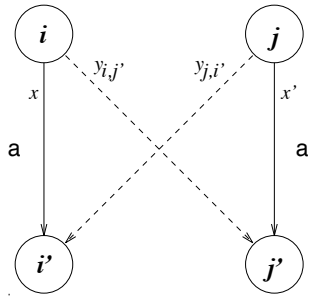

FIGURE 3. A pair of states $\{i, j\}$ in $M(a)$

- there is an arc with label a in the DFA containing j from j to a state j'.

The set $M(a)$ consists of the pairs of states in which processes can be just before an a communication occurs. The fact that the DFAs are deterministic implies that, for each $\{i, j\} \in M(a)$, the states i' and j' are uniquely determined, and the cross arcs introduced by Corbett run from i to j' and from j to i'. Let $M(a)_i = \{j | \{i, j\} \in M(a)\}$.

For each communication event a and each pair $\{i, j\} \in M(a)$, let $z_{\{i,j\},a}$ be a 0-1 variable. Let x and x' be the transition variables in the execution part of Corbett's system of inequalities associated with the arcs i-i' and j-j', respectively, as illustrated in Figure 3. Generate the inequalities

$$z_{\{i,j\},a} \leq x \tag{19}$$

$$z_{\{i,j\},a} \leq x' \tag{20}$$

For each communication event a and each state i with an outgoing arc labeled a from i, generate the equation

$$\sum_{j \in M(a)_i} z_{\{i,j\},a} - x = 0 \tag{21}$$

where x is the transition associated with the outgoing a-arc from i in the execution part. (Note that when $M(a)_i$ is a singleton, (21) makes an inequality of form (19) redundant.) These inequalities imply that $z_{\{i,j\},a}$ is 1 exactly when an a communication occurs between processes in states i and j, and I should impose the condition that states i and j are exited at the same time, or, equivalently, that states i' and j' are entered at the same time. I have

$$s_{i'} - s_{j'} \leq B(1 - z_{\{i,j\},a}) \tag{22}$$

$$s_{j'} - s_{i'} \leq B(1 - z_{\{i,j\},a}) \tag{23}$$

I must also bound the associated critical path variables. I have

$$y_{i,j'} + y_{j,i'} \leq z_{\{i,j\},a} \tag{24}$$

where $y_{r,s}$ is the variable associated with the cross arc running from state r to state s.

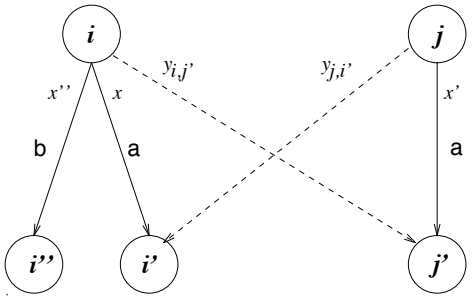

FIGURE 4. A pair of states $\{i, j\}$ in $M(a)$ with an internal event enabled in state i

Let i be any state with an outgoing arc labeled by an internal event. The maximal parallelism assumption implies that the process should not wait in state i. For each arc incident from i, generate the pair of inequalities

$$s_{i'} - s_i - d \leq (B - d)(1 - x) \tag{25}$$

$$s_i + d - s_{i'} \leq (B + d)(1 - x) \tag{26}$$

where the arc is incident to state i', x is the transition variable in the execution part associated with the arc, and d is the duration of the event labeling the arc. These inequalities imply that the entry time for any state reached from i in a single transition is greater than the entry time for i by exactly the duration of the event labeling the transition.

Now let i be a state such that all outgoing arcs are labeled by communication events. Let t_i be a variable representing the time at which state i is left. Generate the inequality

$$s_i - t_i \leq 0 \tag{27}$$

This inequality says that such a state is entered before it is left. For each arc incident from i, generate the inequalities

$$s_{i'} - t_i - d \leq (B - d)(1 - x) \tag{28}$$

$$t_i + d - s_{i'} \leq (B + d)(1 - x) \tag{29}$$

where the arc is incident to state i', x is the transition variable in the execution part associated with the arc, and d is the duration of the event labeling the arc. These inequalities imply that the entry time for any state reached from i in a single transition is greater than the time at which i is left by exactly the duration of the event labeling the transition. These inequalities refer to the time at which the process leaves state i, unlike (25) and (26), because the process may be forced to wait in state i until a communication is possible.

Consider a pair of states $\{i, j\} \in M(a)$ such that the process containing state i can execute an internal computation b in state i, as illustrated in Figure 4. Equations (25) and (26) force the process containing state i to leave that state

immediately after entering. Therefore, if communication between the processes containing states i and j occurs with the processes in those states, i.e., if the variable $z_{\{i,j\},a}$ is 1, we have $s_{i'} = s_i + d_a$, where d_a is the duration of the communication event a. Then equations (22) and (23) force $s_{j'} = s_{i'}$, so the process containing state j leaves that state as soon as the other process enters state i. In this case, neither process waits unnecessarily for the other.

To ensure that processes do not wait unnecessarily for communication in states for which all outgoing arcs are labeled by communication events, we need additional inequalities. For each unordered pair $\{i, j\}$ such that $\{i, j\} \in M(a)$, for some a, and all arcs leaving states i and j are labeled by communication events, let $w_{i,j}$, and $w_{j,i}$ be 0-1 variables. Generate the inequalities

$$t_j - s_i \leq B w_{i,j} \tag{30}$$

$$s_i - t_j < (B+1)(1 - w_{i,j}) \tag{31}$$

$$t_i - s_j \leq B w_{j,i} \tag{32}$$

$$s_j - t_i < (B+1)(1 - w_{j,i}) \tag{33}$$

To ensure that neither process waits for communication after the other is ready, we want to assert that $w_{i,j}$ and $w_{j,i}$ are not both 1. We should enforce this restriction, however, only if states i and j are actually entered during the execution.

We can do this with the inequality

$$w_{i,j} + w_{j,i} \leq 3 - In(i) - In(j) \tag{34}$$

where $In(k)$ is defined to be 1 if k is the start state of a process DFA and the sum of the transition variables associated with arcs incident to k otherwise. Since $w_{i,j}$ and $w_{j,i}$ are 0-1 variables, this inequality has no effect unless $In(i)$ and $In(j)$ are both 1, that is, unless the processes enter both states i and j during the execution.

Remark. As noted in Section 3.1, the numbers of variables and inequalities can be reduced by using nonlinear inequalities. The presence of nonlinear constraints, however, greatly increases the difficulty of solving the resulting optimization problems.

In particular, (25) and (26) can be replaced by the single constraint

$$(s_{i'} - s_i - d)x = 0 \tag{35}$$

Similarly, (28) and (29) can be replaced by

$$(s_{i'} - t_i - d)x = 0 \tag{36}$$

Furthermore, (30)–(34) can be replaced by the single inequality

$$(s_i - u_j)(s_j - u_i)In(i)In(j) \leq 0 \tag{37}$$

where u_k is replaced by t_k or s_k as before, and the $w_{i,j}$ eliminated.

As with Corbett's original technique, this approach can be extended to find an upper bound on the time between two specific events. It is, however, necessary to set the times at the start of the interval to some appropriate single value S. This can be achieved with inequalities of the form

$$s_i - S \leq B(1 - b_i) \tag{38}$$

$$S - s_i \leq B(1 - b_i) \tag{39}$$

at each state in which a process could be at the start of the timed interval.

4. Applying the method to the example

Applying the this method to the example of Figure 1 results in 58 additional linear inequalities involving the 30 variables introduced by Corbett's method and 19 additional variables. The additional inequalities are listed in Appendix A. The full system of inequalities, combining Corbett's inequalities and those added by the new method, thus has 102 inequalities in 49 variables.

With B set to 100 and the duration of each event set to 1, the IMINOS integer programming package [2] requires .6 seconds to solve this system (.5 seconds user time and .1 seconds system time) on a DECstation 5000/125, and reports an upper bound of 2 for the execution time, with the critical path variables y_6 and y_{16} equal to 1 and the remaining critical path variables equal to 0. In this solution, states 2, 4, and 7 are entered at time 1, and states 5 and 8 are entered at time 2. The variables x_1, x_2, and $z_{\{1,3\},a}$ are 1, indicating that the a communication occurred when Processes 1 and 2 were in states 1 and 3, respectively. The variables x_5, x_7, and $z_{\{4,7\},b}$ are 1, indicating that the b communication occurred when Processes 2 and 3 were in states 4 and 7, respectively. Thus, in the first unit of time, Processes 1 and 2 communicated on the a channel, while Process 3 executed the internal event c. During the second unit of time, Processes 2 and 3 communicated on the b channel. This solution does satisfy the hypothesis of maximal parallelism.

5. Conclusion

I have described a method for generating additional inequalities in order to sharpen the upper bounds obtained by Corbett on the time between events in maximally parallel systems with synchronous communication, and illustrated the method with a small example. For this example, the system of inequalities generated by the method involves roughly 130% more inequalities and about 63% more variables than Corbett's original method. The cost of applying the new method, in terms of the size of the integer programming problem that must be solved, may therefore be fairly high.

For this preliminary report, the system of inequalities was generated by hand. It will, however, be a fairly straightforward task to modify the inequality generator component of the constrained expression toolset [2, 6] to automate this process. Once these modifications are complete, it will be possible to carry out experiments

with larger and more complex systems. This will permit a much more complete evaluation of the feasibility and practical utility of the method described here.

Acknowledgement

I am grateful to Ugo Buy and Laura Dillon for their comments on earlier versions of this paper.

Appendix A. Linear inequalities for the example of Figure 1

Of type (18):

$$\begin{aligned} s_1 &= 0 \\ s_3 &= 0 \\ s_6 &= 0 \end{aligned}$$

Of types (19) and (20):

$$\begin{aligned} z_{\{1,3\},a} &\leq x_1 \\ z_{\{1,3\},a} &\leq x_2 \\ z_{\{1,4\},a} &\leq x_1 \\ z_{\{1,4\},a} &\leq x_4 \\ z_{\{3,7\},b} &\leq x_3 \\ z_{\{3,7\},b} &\leq x_7 \\ z_{\{4,7\},b} &\leq x_5 \\ z_{\{4,7\},b} &\leq x_7 \end{aligned}$$

Of type (21):

$$\begin{aligned} z_{\{1,3\},a} + z_{\{1,4\},a} - x_1 &= 0 \\ z_{\{1,3\},a} - x_2 &= 0 \\ z_{\{3,7\},b} - x_3 &= 0 \\ z_{\{1,4\},a} - x_4 &= 0 \\ z_{\{4,7\},b} - x_5 &= 0 \\ z_{\{3,7\},b} + z_{\{4,7\},b} - x_7 &= 0 \end{aligned}$$

Of types (22) and (23):

$$\begin{aligned}
s_2 - s_4 &\leq B(1 - z_{\{1,3\},a}) \\
s_4 - s_2 &\leq B(1 - z_{\{1,3\},a}) \\
s_2 - s_5 &\leq B(1 - z_{\{1,4\},a}) \\
s_5 - s_2 &\leq B(1 - z_{\{1,4\},a}) \\
s_4 - s_8 &\leq B(1 - z_{\{3,7\},b}) \\
s_8 - s_4 &\leq B(1 - z_{\{3,7\},b}) \\
s_5 - s_8 &\leq B(1 - z_{\{4,7\},b}) \\
s_8 - s_5 &\leq B(1 - z_{\{4,7\},b})
\end{aligned}$$

Of type (24):

$$\begin{aligned}
y_9 + y_{11} &\leq z_{\{1,3\},a} \\
y_{10} + y_{12} &\leq z_{\{1,4\},a} \\
y_{13} + y_{15} &\leq z_{\{3,7\},b} \\
y_{14} + y_{16} &\leq z_{\{4,7\},b}
\end{aligned}$$

Of types (25) and (26):

$$\begin{aligned}
s_7 - s_6 - d_c &\leq (B - d_c)(1 - x_6) \\
s_6 + d_c - s_7 &\leq (B + d_c)(1 - x_6) \\
s_8 - s_7 - d_b &\leq (B - d_b)(1 - x_7) \\
s_7 + d_b - s_8 &\leq (B + d_b)(1 - x_7) \\
s_8 - s_7 - d_d &\leq (B - d_d)(1 - x_8) \\
s_7 + d_d - s_8 &\leq (B + d_d)(1 - x_8)
\end{aligned}$$

Of type (27):

$$\begin{aligned}
s_1 - t_1 &\leq 0 \\
s_3 - t_3 &\leq 0 \\
s_4 - t_4 &\leq 0
\end{aligned}$$

Of types (28) and (29):

$$
\begin{aligned}
s_2 - t_1 - d_a &\le (B - d_a)(1 - x_1) \\
t_1 - s_2 - d_a &\le (B - d_a)(1 - x_1) \\
s_4 - t_3 - d_a &\le (B - d_a)(1 - x_2) \\
t_3 + d_a - s_4 &\le (B + d_a)(1 - x_2) \\
s_4 - t_3 - d_b &\le (B - d_b)(1 - x_3) \\
t_3 + d_b - s_4 &\le (B + d_b)(1 - x_3) \\
s_5 - t_4 - d_a &\le (B - d_a)(1 - x_4) \\
t_4 + d_a - s_5 &\le (B + d_a)(1 - x_4) \\
s_5 - t_4 - d_b &\le (B - d_b)(1 - x_5) \\
t_4 + d_b - s_5 &\le (B + d_b)(1 - x_5)
\end{aligned}
$$

Of types (30)–(33):

$$
\begin{aligned}
t_3 - s_1 &\le B w_{1,3} \\
s_1 - t_3 &< (B + 1)(1 - w_{1,3}) \\
t_1 - s_3 &\le B w_{3,1} \\
s_3 - t_1 &< (B + 1)(1 - w_{3,1}) \\
t_4 - s_1 &\le B w_{1,4} \\
s_1 - t_4 &< (B + 1)(1 - w_{1,4}) \\
t_1 - s_4 &\le B w_{4,1} \\
s_4 - t_1 &< (B + 1)(1 - w_{4,1})
\end{aligned}
$$

Of type (34):

$$
\begin{aligned}
w_{1,3} + w_{3,1} &\le 1 \\
w_{1,4} + w_{4,1} &\le 2 - x_2 - x_3
\end{aligned}
$$

References

1. G. S. Avrunin, U. A. Buy, and J. C. Corbett. Integer programming in the analysis of concurrent systems. In K. G. Larsen and A. Skou, editors, *Computer Aided Verification, 3rd International Workshop Proceedings*, volume 575 of *Lecture Notes in Computer Science*, pages 92–102, Aalborg, Denmark, July 1991. Springer-Verlag.
2. G. S. Avrunin, U. A. Buy, J. C. Corbett, L. K. Dillon, and J. C. Wileden. Automated analysis of concurrent systems with the constrained expression toolset. *IEEE Trans. Softw. Eng.*, 17(11):1204–1222, Nov. 1991.
3. G. S. Avrunin, J. C. Corbett, L. K. Dillon, and J. C. Wileden. A method for deriving bounds on the time between events in concurrent systems. Constrained Expression Memorandum 91-3, Department of Computer and Information Science, University of Massachusetts, Amherst, 1991.

4. J. C. Corbett. Constrained expression analysis of multi-processor real-time systems (extended abstract). Constrained Expression Memorandum 91-2, Department of Computer and Information Science, University of Massachusetts, Amherst, 1991. Distributed at the ONR Fourth Annual Review and Workshop on the Foundations of Real-Time Computing, October 31–November 1, 1991, Washington, D.C.
5. J. C. Corbett. *Automated Formal Analysis Methods for Concurrent and Real-Time Software*. PhD thesis, University of Massachusetts at Amherst, 1992.
6. J. C. Corbett. New and improved inequality generation in the constrained expression toolset. Constrained Expression Memorandum 92-1, Department of Computer and Information Science, University of Massachusetts, Amherst, 1992.

Department of Mathematics and Statistics, University of Massachusetts at Amherst, Amherst, Massachusetts 01003
E-mail address: avrunin@math.umass.edu