Online Companion for

"On the Complexity of Verifying Structural Properties of Discrete Event Simulation Models"

Operations Research
Vol. 47, No. 3, May-June 1999

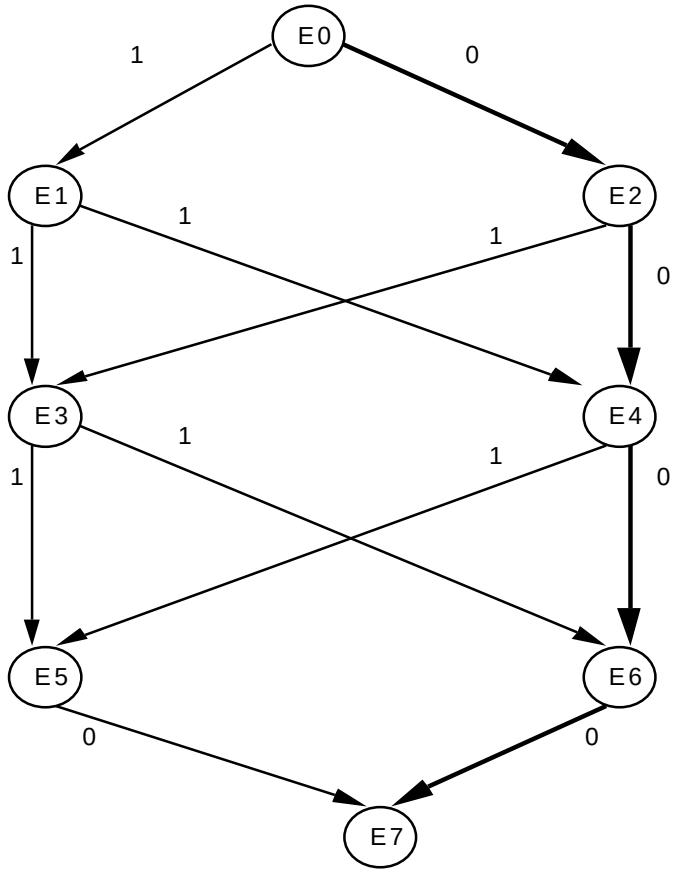
Sheldon H. Jacobson
Virginia Polytechnic Institute and State University

Enver Yücesan
INSEAD
France

An example with three variables and four clauses is presented to illustrate the construction used in the proof of Theorem 1. Let U = {u_1, u_2, u_3} be the set with three Boolean variables and let V = {V_1, V_2, V_3, V_4} be the set of four clauses such that $V_1 = (u_1, u_2, u_3)$, $V_2 = (u_1, u_2, \bar{u}_3)$, $V_3 = (u_1, \bar{u}_2, u_3)$, and $V_4 = (\bar{u}_1, u_2, u_3)$.

Suppose the time delays, a^i_j, are set as follows: $a^1_2 = a^2_4 = a^3_4 = 0$; $a^i_j = 1$ for all other i and j. An event graph model is depicted in Figure A1. For ease of presentation, we have omitted all canceling edges in the figure. These edges would emanate from vertex E_7 into vertices E_1, E_2, ..., E_6, with no time delays. With these input variables, the event graph model assigns values of one to all of the Boolean variables: $u_1 = u_2 = u_3 = 1$. Therefore all four clauses are satisfied, resulting in S=4 with M_1=4. Hence, 3-SAT is solved for the input parameter values, a^i_j, i=1,2,3; j=1,2,3,4, resulting in the event sequence $E_0E_2E_4E_6E_7 \Rightarrow S = 4$.

Figure A1. Simulation Model for 3-SAT



APPENDIX 2: PROOF OF THEOREM 2

A polynomial Turing reduction from ACCESSIBILITY to ORDERING is constructed. The discrete event simulation model specification associated with ORDERING is the same as that associated with ACCESSIBILITY, with the following modifications:

Define two new events and two new states.

E'_1: If STATE = S, then STATE $\leftarrow S_1$

If STATE $\neq$ S, leave STATE unchanged.

E'_2: If STATE = S, then STATE $\leftarrow S_2$

If STATE $\neq$ S, leave STATE unchanged.

The events E'_1 and E'_2 are scheduled with non-negative time delays t_1 and t_2, respectively, which allow these events to be executed anywhere in an event sequence. Each event of the model specification for ACCESSIBILITY is also defined for the model specification for ORDERING, with the additional condition that, if STATE=S_1 (S_2), then STATE remains at S_1 (S_2). This ensures that STATE remains unchanged, no matter which event is executed after these states are reached. Therefore once state S_1 or S_2 is reached, the value for STATE does not change. Finally, let $M_2 = M_1 + 2$ and $D_2 = D_1 + 2$.

Suppose ORDERING is solved by the sequence of events $E_1, E_2,...,E_m$. Both E'_1 and E'_2 can be scheduled and executed at most once. If $E'_1 = E_i$ for i=1,2,...,m-2, and $E_0E_1E_2...E_{i-1} \Rightarrow$ S, then S_2 can never be reached. On the other hand, if $E'_1=E_i$ for i=1,2,...,m-2, and $E_0E_1E_2...E_{i-1} \not\Rightarrow$ S, then S_1 can never be reached. Therefore, E'_1 must be either E_{m-1} or E_m. Similarly, E'_2 must be either E_{m-1} or E_m. Since E_0, E_1, E_2,..., E_m solves ORDERING, then $E'_1 = E_{m-1}$ and $E'_2 = E_m$, which implies $E_0E_1E_2...E_{m-2} \Rightarrow$ S. Therefore, E_0, E_1, E_2,..., E_{m-2} solves ACCESSIBILITY.

Conversely, suppose that there does not exist any sequence of events of length at most M_2 that solves ORDERING. Recall that states S_1 and S_2 can only be accessed by the execution of events E'_1 and E'_2, respectively, in state S. Therefore, since states S_1 and S_2 cannot be accessed in the particular instance of ORDERING, then S cannot be accessed in the arbitrary instance of ACCESSIBILITY. This means that there does not exist any sequence of events of length at most M_1 that solves the arbitrary instance of ACCESSIBILITY.

This reduction can be performed in time polynomial in n, where n is the size of the instance of ACCESSIBILITY, since it results in O(1) additional states and events, and O(n) event modifications. □

APPENDIX 3: PROOF OF THEOREM 3

A polynomial Turing reduction from ORDERING to NONINTERCHANGEABILITY is constructed. The discrete event simulation model specification associated with NONINTERCHANGEABILITY is the same as that associated with ORDERING. The following two implementations of the discrete event simulation model specification associated with the model implementation for ORDERING are defined. For any valid

event sequence of length m ≤ M_3, these implementations are distinct in that they use different event execution rules.

DES1: For j = 1 **to** m-1

Upon the execution of event E_{j-1} and the establishment of the current state,
determine the resulting state if E_jE_{j+1} were executed.
Store the resulting state, S.
Upon the execution of event E_{j-1} and the establishment of the current state,
determine the resulting state if $E_{j+1}E_j$ were executed.
Store the resulting state, S'. Note that if $E_{j+1}E_j$ results in an invalid event sequence, then set S' = S", an artificial state.
If S ≠ S' and S'≠ S", execute E_jE_{j+1}, output state S, and stop.
If S' = S" and j = m-1, execute E_jE_{j+1} and stop.
If S' = S" and j ≠ m-1, execute E_j and continue.
If S = S' and j = m-1, execute E_jE_{j+1} and stop.
If S = S' and j ≠ m-1, execute E_j and continue.
Next j.

DES2: For j=1 **to** m-1

Upon the execution of event E_{j-1} and the establishment of the current state,
determine the resulting state if E_jE_{j+1} were executed.
Store the resulting state, S.
Upon the execution of event E_{j-1} and the establishment of the current state,
determine the resulting state if $E_{j+1}E_j$ were executed.
Store the resulting state, S'. Note that if $E_{j+1}E_j$ results in an invalid event sequence, then set S' = S", an artificial state.
If S ≠ S' and S'≠ S", execute $E_{j+1}E_j$, output state S', and stop.
If S' = S" and j = m-1, execute E_jE_{j+1} and stop.
If S' = S" and j ≠ m-1, execute E_j and continue.
If S = S' and j = m-1, execute E_jE_{j+1} and stop.
If S = S' and j ≠ m-1, execute E_j and continue.
Next j.

The states S_1 and S_2 given for ORDERING are the same states defined in conjunction with NONINTERCHANGEABILITY. State S" is an artificial state defined only for NONINTERCHANGEABILITY. Finally, let $M_2 = M_3$ and $D_2 = D_3$.

Suppose that NONINTERCHANGEABILITY is solved. That is, there exists a sequence of events $E_1, E_2,..., E_m$, m≤M_3, such that

DES1: $E_0E_1E_2...E_m \Rightarrow S_1$

DES2: $E_0E_1E_2...E_m \Rightarrow S_2$

where $S_1 \neq S_2$. By the definition of DES1 and DES2,

$E_0E_1E_2...E_{m-1}E_m \Rightarrow S_1$

$E_0E_1E_2...E_mE_{m-1} \Rightarrow S_2$

for the model implementation associated with ORDERING. Therefore, the sequence of events $E_0E_1E_2...E_{m-1}E_m$ solves ORDERING.

Conversely, suppose that NONINTERCHANGEABILITY cannot be solved. Then there does not exist a sequence of events of length at most M_3 such that, when executed in the two model implementations, states S_1 and S_2 cannot both be reached. Based on the definitions of these two model implementations for the particular instance of NONINTERCHANGEABILITY, there does not exist a sequence of events of length at most M_2 that solves the arbitrary instance of ORDERING.

The additional run time needed to execute each of the model implementations is polynomial in n per event and, hence, the reduction can be performed in polynomial time in n, where n is the size of the arbitrary instance of ORDERING. □

APPENDIX 4: PROOF OF THEOREM 4

A polynomial Turing reduction from ACCESSIBILITY to STALLING is constructed. The discrete event simulation model specification associated with STALLING is the same as that associated with ACCESSIBILITY. Define stopping condition C as "TRUE = FALSE" (that is, a condition that can never be satisfied) and, for state S, define the following new events:

E: If STATE = S, then cancel all events on L and STATE ← S*.
Else if L = ∅, schedule F, or
if L ≠ ∅, continue execution without altering the states.

F: Cancel all events on the events list.
Schedule event F.

Let S^* be a new state not defined by the discrete event simulation model specification associated with ACCESSIBILITY and only reachable through event E. Define the initial event E_0 to be the same as in ACCESSIBILITY, with the addition of event E being scheduled with non-negative time delay t. This time delay allows for event E to be executed anywhere in an event sequence. Finally, let $M_4=M_1+1$ and $D_4=D_1+2$.

Suppose that the sequence of events $E_1, E_2,..,E_m$ solves STALLING. Therefore, $E_0E_1E_2...E_m \Rightarrow S^*$, where the execution is stalled. This implies that for some j, $j \le m$, $E_j = E$ with $E_0E_1E_2...E_{j-1} \Rightarrow S$. Now suppose that $E_i = E$ for some i, $1 \le i \le m-1$. Suppose also that the current state after the execution of E_{i-1} is S; that is, $E_0E_1E_2...E_{i-1} \Rightarrow S$. The execution of $E_i = E$ will then cancel all of the events on the events list, L. This, however, is impossible since the remainder of the sequence of events --all the way to E_m-- must be executed. Therefore, $E_m = E$, hence $E_0E_1E_2...E_{m-1}$ solves ACCESSIBILITY.

Conversely, suppose that the particular instance of STALLING cannot be solved. Then there does not exist a sequence of events of length at most M_4 such that state S* can be accessed, with condition C not satisfied and the events list empty. Suppose the arbitrary instance of ACCESSIBILITY does have a solution. This means that there exists a sequence of events of length at most M_1 such that state S can be accessed. Since $M_4 = M_1 + 1$, adding event E to this sequence leads into state S^* with condition C not satisfied and the events list empty. However, this implies that the resulting event sequence solves the particular instance of STALLING, which is a contradiction. Therefore, the arbitrary instance of ACCESSIBILITY does not have a solution.

Finally, since E can be executed in polynomial time in n, the reduction can be performed in polynomial time in n, where n is the size of the instance of ACCESSIBILITY. □