An Event Graph –based Approach to ARGESIM Comparison "C10 Dining Philosophers II" with Simkit

Arnold Buss, Naval Postgraduate School, Michigan; abuss@nps.navy.mil

Simulator: Simkit, a Java package, provides an API for creating Discrete Event Simulation models based on Event Graph methodology. Simkit's strengths are its modelling flexibility and platform independence. Simkit's Event List algorithm does not throw an error when it becomes empty, but simply terminates the simulation run.

Model: Creating an Event Graph model consists of defining the parameters for the model, defining state variables and their initial values, the state transition functions that form the events, and finally the scheduling relationships between the events. The parameters for the Dining Philosopher's problem consist of *n*, the number of philosophers; {t_M}, the sequence of meditation times; and {t_E}, and the sequence of eating times. For the comparison, *n*=5 and the sequences are specified as iid discrete uniform (1,10) random variates. The state variables are defined as {p[0],… ,p[n-1]} (C/Java numbering) with possible values {M, WL, WR E, } (Meditating, Waiting for Left chopstick, Waiting for a Right chopstick, and Eating, resp.).

The Event Graph is shown in fig. 1, where the initial Run event is omitted for clarity. At the start of the simulation there are n StartMeditating(i) events on the event list with parameters 0…n-1. All the expressions are assumed to be modulo n (e. g., if philosopher 0 is waiting to pick up both chopsticks, EndEating(4) will schedule TakeLeft(0), since 4+1 mod 5 = 0). Note that the StartEating(i) event corresponds to philosopher i picking up the chopstick at the right, which could only occur if he had already picked up the left chopstick.

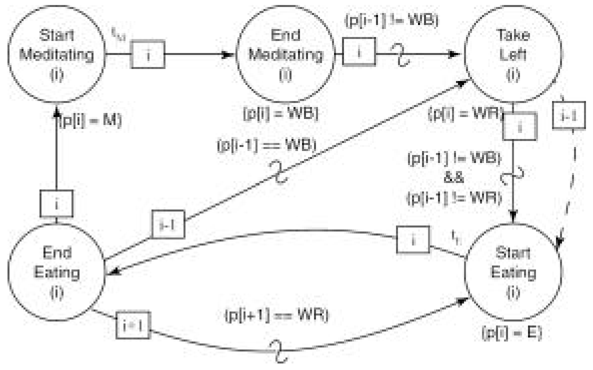

Figure 1: Event Graph for Dining Philosophers

Tiebreaking can be specified by a partial ordering of the Event Graph scheduling edges. For this model, the scheduling edges for the TakeLeft(i) event have higher priority than any other events.

Thus, if StartEating(2) and TakeLeft(1) are scheduled at the same time, TakeLeft(1) will always occur first, and will cancel StartEating(2). Note that the states of the chopsticks can be completely determined by the states of the philosophers.

Task a: Simulation of the System. With a single run, the simulation deadlocked at time 2345079. Statistics on the waiting time and chopstick utilization are shown in Table 1 and Table 2. There is no need to estimate mean the time eating or meditating, since they are completely determined by the probability distributions. The standard deviations for waiting time were gathered, despite the fact that they are meaningless due to correlation of the observations.

Philosopher	0	1	2	3	4	All
Mean	11.43	11.45	11.44	11.42	11.44	11.45
Std	8.07	8.07	8.06	8.09	8.08	8.08

Table 1: Philosopher Waiting Times

Chopstick	0	1	2	3	4	All
Utilization	91.88%	91.97%	91.92%	91.96%	91.93%	91.93%

Table 2: Chopstick Utilization

Task b: Correct Event Management. The model was run in verbose mode, which prints out the state of the event list. The state transitions for the current event were also listed:

```
philosopher[1]:
        Waiting for Left => Waiting for Right
chopstick[1]: 0 => 1
Time: 2345079.000 Current Event: TakeLeft {1}
 ** Event List --  **
2345079.000     TakeLeft        {2}
2345079.000     TakeLeft        {4}
2345079.000     TakeLeft        {0}
2345079.000     StartEating     {3}
2345079.000     StartEating     {1}
 ** End  of Event List --  **
philosopher[2]:
        Waiting for Left => Waiting for Right
chopstick[2]: 0 => 1
Time: 2345079.000 Current Event: TakeLeft {2}
 ** Event List --  **
2345079.000     TakeLeft        {4}
2345079.000     TakeLeft        {0}
2345079.000     StartEating     {1}
 ** End  of Event List --  **
```

Task c: 50 Replications. The model was repeated 50 times and statistics gathered on the deadlock times. The times varied between 9391 and 9172542, with a mean of 3125763.14 and a standard deviation of 2447495.21. Note that in this case the standard deviation *is* meaningful because the runs are independent.

C10 Classification: Object/Event – oriented Approach

Simulator: Simkit Rel. of 2002