An Object-oriented Solution to ARGESIM Benchmark C4 'Dining Philosophers Problem' implemented with AnyLogic

Michael Gyimesi, Andreas Dielacher, Thomas Handl, Christian Widtmann,
Vienna University of Technology, Austria, *mgyimesi@osiris.tuwien.ac.at*

Simulator: AnyLogic is a JAVA based general purpose simulator supporting System Dynamics, agent based and discrete event modelling approaches under the object-oriented model design paradigm. Users may apply the concepts of objects, interfaces, hierarchy, message passing and more to generate their own modelling constructs.

AnyLogic supports a large number of pre-defined probability distributions for stochastic modelling and can be used to design deterministic models as well. Analyses of output data can be performed and visualized directly within the simulator. AnyLogic enables the user to generate interactive 2D and 3D animations of the simulation as well as portable web-enabled models for use as web applets. The model described in this article was created in AnyLogic Version 5.5.

Model: This comparison C4 describes the problem of five philosophers competing for five chopsticks on a shared table. The used model is just as classical, since it features the philosophers, the chopsticks and the table modelled in one object class, respectively.

The table is the top object, instancing the five philosophers and the five chopsticks. In this object oriented approach, each philosopher is connected to each of the two adjacent chopsticks by one message queue. The normal procedure is requesting at first the left and then the right chopstick, which in return are granted to the philosopher by an according message. The philosopher's two requests for the chopsticks are sent with a relative delay (pause). The duration of the activities *thinking*, *pause* and *eating* is distributed uniformly across a certain range of values. The philosophers competing for an insufficient number of chopsticks can be seen as a metaphor on an arbitrary distributed system whose components compete for limited shared resources.

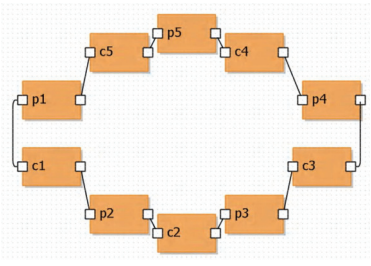

Figure 1: Top-Level Model

So far the model resembles the example included with AnyLogic. An expansion was made, building a hybrid model since the philosophers maintain a certain level of "saturation", which increases or decreases according to a simple differential equation while the philosophers are eating or not eating. We discarded this hybrid approach in favor of a purely discrete model by replacing the continuous "saturation" function of the example model by a discrete "calories" function, described in the context of task b.

In the original model, philosophers can reach a deadlock situation when for example all philosophers simultaneously request their respective left chopstick, causing them to wait forever for their respective right chopstick.

A-Task: A deadlock occurs when all five philosophers, having requested and received the left chopstick, simultaneously enter the *pause* state with no right chopstick available. Thus a timeout has been introduced, indicated by the edge leading from *waitingRight* back to *thinking*, after which a philosopher returns the left chopstick and thereby avoids a deadlock for the whole system. The parameter *maxPause* which influences the time between the left and the right request has a significant impact on the likelihood of a deadlock because it controls the amount of time available for the adjacent philosopher to seize the right chopstick.

Various simulation runs under variation of *maxPause*, left and right timeout illustrated that a high maximum pause has a crucial impact on the blocking of one chopstick and that the consecutive high waiting times result in easy starvation. This can be alleviated by using high timeout values particularly while in *waitingRight*, since this value has a direct impact on the time that the left chopstick is seized without actually being used. The timeout value for waiting for the left chopstick directly affects the probability that an available chopstick is going to be seized. The sum of

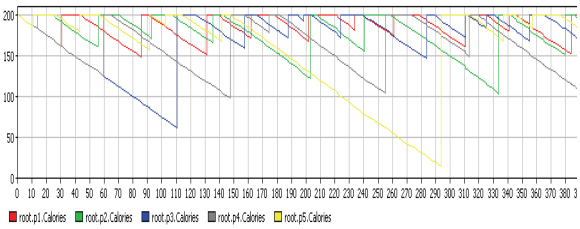

Figure 2: Results for symmetric timeouts for waitingLeft and waitingRight

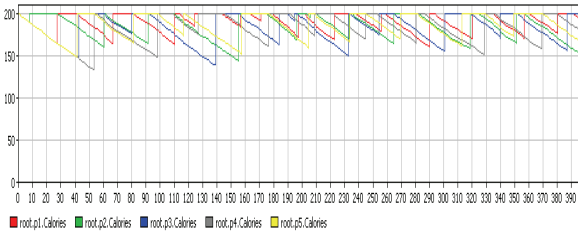

Figure 3: Results for asymmetric timeouts for waitingLeft and waitingRight

both timeouts and *maxPause* affects the fraction of time that is available for actually doing work. Generally speaking, high timeout values favor quick and efficient use of chopsticks, keeping in mind that this is lost for doing actual work. After adding timeouts, *maxPause* is only relevant for the number of calories consumed while inactively waiting to seize the right chopstick.

The results in figure 2 and 3 reflect the caloric distribution for symmetric timeouts of 10 for waiting for both chopsticks as well as for asymmetric timeouts of 2 for the left and 50 for the right chopstick, with 200 calories and the value of 20 for all max values. It can be clearly seen that the asymmetric variant shows better caloric characteristics.

B-Task: Figure 4 depicts a philosopher's state machine incorporating all elements for modelling task a and b.

A philosopher always leaves the *eating* state with the full number of calories while in the original approach, he didn't necessarily have to eat until full saturation. This adaptation resembles that once the critical section is reached, a component will only leave the section again when it has fulfilled its task whose deadline will then be fully restarted. The duration of remaining in the *eating* state is still randomly distributed, since the duration of the critical section may not be constant but e.g. data dependent. The notion of calories thus resembles the timeout that each component (or philosopher) is able to wait until he is required to enter the critical section. When calories reaches zero (i.e. the component misses its deadline for the critical section), the philosopher enters the state *dead* and is no longer participating in the dinner i.e. the system, indicating a potential system failure.

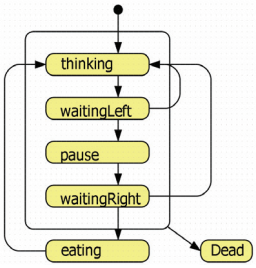

Figure 4: Philosophical behaviour

Thus care has to be taken to initialize a philosopher with a reasonable number of calories with regard to the maximum thinking, pause and eating periods. In our experiments, initial 200 calories turned out to be adequate for maximum thinking and eating periods of 20 time units each, with varying maximum pause and timeout values. Despite being a discrete variable, the used version of AnyLogic did only accept equations being attributed to a state rather than program statements. Thus the discrete decrease of calories had to be modelled as a first order derivate with a constant value of -1.

C-Task: An important aspect of a distributed system is the amount of time that it can spend on its actual task. Thus the timeout value used for the edge from *waitingLeft* to *thinking* is significant for this ratio. When the timeout is too long, much time is wasted waiting for a chopstick. If it is unreasonably short, the philosopher will quickly enter the state *thinking* again, spend the next period working and thus risk running out of calories.

Resumé: The modeling approach used in this solution is an object-oriented process approach. AnyLogic provides a powerful environment with different well suited possibilities for building models of distributed parallel systems and performing experiments with different settings.

Corresponding author: Michael Gyimesi,
Department of Analysis and Scientific Computing
Vienna University of Technology
Wiedner Hauptstraße 8-10, 1040 Vienna, Austria
mgyimesi@osiris.tuwien.ac.at

Received: June 28, 2007
Revised: February 10, 2008
Accepted: February 20, 2008