SNE 16/1 (46), August 2006

A Cellular Automata – Approach to ARGESIM Comparison C 10 'Dining Philosophers II' using MATLAB

Gerhard Höfinger, Vienna University of Technology, Vienna; *Gerhard.Hoefinger@tuwien.ac.at*

Simulator: MATLAB (version 7.01) is (not only) a powerful programming language and was used without any toolboxes to program the simulation model as well as the statistical evaluations. For multiple simulation runs, the code can be translated automatically into C++ and compiled, which makes execution faster - especially code with extensive for – loops gets slow in MATLAB.

Model. The behaviour of the five philosophers can be studied using a one-dimensional cellular automaton (CA). The CA depends on some stochastic effects (time until hunger arises, time it takes to eat) and the behaviour of their neighbours (which causes chopsticks to be in use or not). Time is discrete in this model (steps of one minute), so time can be modelled as steps of the automaton. A trick was used to get the uniform distribution of thinking and eating times into the model: There is not only one cell state for eating (resp. thinking) time, but several states, represented by decreasing numbers, are used as count down timer. The first state in an eating (thinking) period is chosen by a random generator. The matrix (Table 1) in which cell states are stored shows this (1-10: philosopher thinking; 11-20: eating; 21 - waiting for left chopstick; 22 - waiting for right chopstick).

1	7	2	10	2
17	6	1	9	1
16	5	16	8	22
15	4	15	7	22
14	3	14	6	22
13	2	13	5	22
12	1	12	4	22
11	21	11	3	22
4	11	6	2	13
3	10	5	1	12
2	9	4	22	11
1	8	3	12	4
…	…	…	…	…

Table 1: CA states for philosophers.

A - Task: Single Run – Evaluations. Simulation is performed by stepwise update of the CA, with sampling of statistical data. In a fully programmed approach the statistical evaluation takes most of the effort. It is not possible to store all the states of the automaton for all times up to a deadlock (which occurs very rarely). As it takes too much time to evaluate statistically after every simulation step, a compromise was chosen: the states are stored for at most 7000 steps, if memory is full, all necessary statistical computations are performed (Table 2 and Table 3, deadlock after 8.003.569 steps).

Chopstick utilisation	
chopstick 1	0.9196
chopstick 2	0.9194
chopstick 3	0.9195
chopstick 4	0.9194
chopstick 5	0.9197
total	0.9195

Table 2: Chopstick Utilisation, single run.

	Eating		Thinking		Waiting	
	mean	std. Dev.	mean	std. dev.	mean	std. dev.
P 1	5.5007	2.8734	5.4879	2.8699	11.4356	8.0647
P 2	5.4970	2.8696	5.4930	2.8712	11.4412	8.0617
P 3	5.4898	2.8682	5.4902	2.8684	11.4393	8.0727
P 4	5.5020	2.8676	5.4912	2.8702	11.4234	8.0557
P 5	5.4894	2.8694	5.4899	2.8705	11.4329	8.0581
total	5.4958	2.8697	5.4904	2.8700	11.4345	8.0626

Table 3: Statistics for the philosophers' states

B - Task: Correct management of simultaneous access. By construction as matrix operations and CA update, progress in the system is well defined. The Gantt chart (Figure 1) shows that the behaviour of the philosophers was implemented correctly.

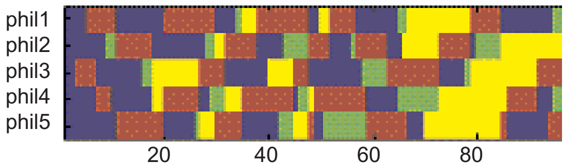

Figure 1: Gantt chart with philosophers actions: red – eating, green and yellow – waiting, blue – thinking

C - Task: Multiple simulation runs. Within 50 simulation runs, the time until a deadlock varied from 16.839 to 23.428.401 time steps (mean time 3.752.924). Figure 2 shows deadlock times versus run number). The CA detects a deadlock by construction.

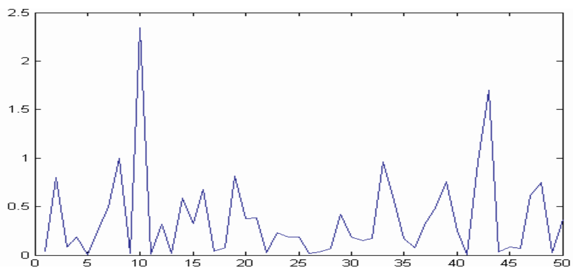

Figure 2: Deadlock times versus run number.

Classification: Cellular automata approach

Corresponding author: Gerhard Höfinger,
Vienna Univ. of Technology, Inst. f. Analysis and Scientific Computing, Wiedner Hauptstr. 8-10, 1040 Vienna, Austria
Gerhard.Hoefinger@tuwien.ac.at

Received: May 20, 2006
Accepted: June 30, 2006