COMP 3610 Tutorial 9

12 October, 2023

Note: for the purposes of these exercises, you can assume more expressive boolean and arithmetic expressions with their natural definitions, i.e.

$$\begin{array}{lcl} a & ::= & x \mid !l \mid n \mid -a \mid a + a \mid a - a \mid a * a \mid a/a \mid a \ \textbf{mod}\ a \\ b & ::= & \textbf{true} \mid \textbf{false} \mid \neg b \mid b \wedge b \mid b \vee b \mid a = a \mid a \geq a \mid a \leq a \mid a > a \mid a < a \end{array}$$

Exercise 1

Draw the lattices of possible executions of the following GCL programs:

1. $\langle l_1 := 3 \parallel \textbf{if}\ l_1 = 4\ \textbf{then}\ l_1 := 1\ \textbf{else}\ l_1 := 2, \{l_1 \mapsto 4\}\rangle$
2. $\langle l_1 := !l_1 + !l_1 \parallel l_1 := !l_1 + !l_1, \{l_1 \mapsto 1\}\rangle$

Exercise 2

For the following programs, determine whether they always deadlock, sometimes deadlock, or never deadlock. Show one potential run for all possible scenarios (deadlock, no deadlock). Come up with variants that allow/guarantee the presence/absence of deadlocks.

1. (GCD)

$$\begin{array}{l} \alpha!1 \parallel \beta!1 \\ \parallel \textbf{do}\ \alpha?x \to \beta?y; \alpha!x; \beta!y\ \textbf{od} \\ \parallel \textbf{do}\ \alpha?x \to \beta?y; \alpha!x; \beta!y\ \textbf{od} \end{array}$$

2. (GCD)

$$\begin{array}{l} \alpha!1 \parallel \beta!1 \\ \parallel \textbf{do}\ \alpha?x \to \beta?y; \beta!y; \alpha!x \ [\!]\ \textbf{true} \to \textbf{skip}\ \textbf{od} \\ \parallel \textbf{do}\ \beta?y \to \alpha?y; \beta!y; \alpha!x \ [\!]\ \textbf{true} \to \textbf{skip}\ \textbf{od} \end{array}$$

3. (CCS)

$$\begin{array}{l} P_1() \stackrel{\text{def}}{=} (\alpha?x \to \beta?y \to \alpha!x \to \beta!y \to P_1()) + (\alpha?x \to \beta?y \to \gamma!x \to \textbf{nil}) \\ P_2() \stackrel{\text{def}}{=} (\alpha?x \to \beta?y \to \alpha!x \to \beta!y \to P_2()) + (\gamma?x \to \textbf{nil}) \\ \alpha!1 \to \textbf{nil} \parallel \beta!1 \to \textbf{nil} \parallel P_1() \parallel P_2() \end{array}$$

Exercise 3

For each language, you can pick some reasonable way of getting input and producing output, as appropriate for the given language. Write a program that is given a number $n \geq 1$ and returns the factorial of n,

1. in GCL
2. in CCS

Exercise 4

Consider a number-guessing game: Player A secretly chooses a random number n between 1 and 100. Player B guesses a number m, tells player A, then player A tells player B whether $m = n$, $m < n$, or $m > n$. Player B repeats guessing until $m = n$; their score is how many guesses it took. Model this game using at least two parallel processes communicating via channels, at least one process for player A, and at least one process for player B. Implement an optimal guessing strategy for player B. You can do this in GCL, or CCS, or both. To pick a random number, use nondeterministic choice, but for simulation purposes, you can imagine picking between a smaller number of choices, say 13, 50, 72, and 84, instead of all numbers between 1 and 100.