Homework No. 2 - Solution

EECS 290N

Edward A. Lee
EECS Department
University of California at Berkeley
Berkeley, CA 94720, U.S.A.

1. Classify each of the following as a partial order relation, a total order relation, or neither. State whether it is a CPO.

   (a) The set A = seconds, grams, meters, pounds, yards of units and a relation R defined by $(a, a') \in A$ if a can be converted to a' by scaling by a unitless constant.

   (b) The set of all countable (including finite) subsets of $\mathbb{R}$, ordered by set inclusion.

   (c) The set $\{1/n \mid n \in \mathbb{N}\} \cup \{0\}$, ordered by the natural numerical order (where $1/0$ is interpreted as ∞).

   (d) The set $\{1/n \mid n \in \mathbb{N}\}$, ordered by the reverse numerical order (where again $1/0$ is interpreted as ∞).

   Solution.

   (a) This is neither because it does not satisfy the reflexive property. It is not a CPO because it is not an order relation.

   (b) This is a partial order. It is not a CPO. To show this, consider the chain consisting of the sets $\{0, 1\}$, $\{0, 0.5, 1\}$, $\{0, 0.25, 0.5, 0.75, 1\}$, etc., where in each new set, a new element is added halfway between each pair of elements. The least upper bound of this set is the range $[0, 1]$ of real numbers, which is not countable.

   (c) This is a total order and a CPO. It clearly has a bottom element, and every chain has a LUB in the set.

   (d) This is a total order, but not a CPO because the set itself has no least upper bound in the set.

   □

2. Let $(A, \leq)$ and $(B, \leq)$ each be a CPO. We can form a poset $(A \times B, \leq)$ where the order is a **lexicographic order**, sometimes called the **dictionary order**, where for all $(a_1, b_1), (a_2, b_2) \in A \times B$,

$$(a_1, b_1) \leq (a_2, b_2) \iff (a_1 = a_2 \text{ and } b_1 \leq b_2) \text{ or } (a_1 \neq a_2 \text{ and } a_1 \leq a_2).$$

   (a) With this order, show that $(A \times B, \leq)$ is a CPO.

(b) Suppose that $A = T_A^{**}$, $B = T_B^{**}$, and both CPOs use the prefix order, for arbitrary sets T_A and T_B. Suppose that $a \in T_A$ and $b_1, b_2 \in T_B$. Consider the set of sequences

$$C = \{((a), (b_1)), ((a, a), (b_2)), ((a, a, a), (b_1)), ...\}$$

Under the lexicographic order, this is a chain. Find its LUB.

Solution.

(a) It is straightforward to show that it is a poset, and that it has a least element, $(\perp_A, \perp_B)$, assuming A and B are posets with least elements $\perp_A$ and $\perp_B$, respectively. It is sufficient to show that every chain C has a LUB. Consider a chain $C = \{(a_n, b_n) \mid n \in \mathbb{N}\}$. Note that $\hat{\pi}_1(C) = \{a_n \mid n \in \mathbb{N}\}$, is a chain in A, by the definition of the lexicographic order. Hence it has a LUB, which we denote $a = \bigvee \hat{\pi}_1(C)$. If $a \notin \hat{\pi}_1(C)$, then

$$\bigvee C = (a, \perp_B).$$

If $a \in \hat{\pi}_1(C)$, then define the set

$$Q = \{b \in \hat{\pi}_2(C) \mid (a, b) \in C\}.$$

By the definition of the lexicographic order, this is a chain in B, and hence has a LUB. In this case,

$$\bigvee C = (a, \bigvee Q).$$

(b) The LUB is

$$\bigvee C = ((a, a, a, ...), \perp_B).$$

□

3. Given a set A and the CPO $(A^{**}, \sqsubseteq)$, for each of the following functions, state whether it is monotonic, continuous, both, or neither. Assume the domain and codomain of every function is A^{**}. Assume the period operator represents concatenation of sequences.

(a) The **unit delay** function d given by $\forall\ s \in A^{**}$, $d(s) = (a).s$ where $a \in A$.

(b) The **trailer function** t given by, $\forall\ s \in A^{**}$,

$$t(s) = \begin{cases} s.a & \text{if } s \text{ is finite} \\ s & \text{otherwise} \end{cases}$$

where $a \in A$.

(c) The **is finite function** f given by, $\forall\ s \in A^{**}$,

$$f(s) = \begin{cases} (a) & \text{if } s \text{ is finite} \\ (a') & \text{otherwise} \end{cases}$$

where $a, a' \in A$, $a \neq a'$.

(d) An alternative **is finite function** f' given by, $\forall\ s \in A^{**}$,

$$f(s) = \begin{cases} (a) & \text{if } s \text{ is finite} \\ (a, a) & \text{otherwise} \end{cases}$$

where $a \in A$.

(e) Let $m\colon (A^{**})^2 \to A^{**}$ be the **fair alternating merge** function, defined as follows. Given two infinite sequences $s_1 = (a_0, a_1, \cdots)$ and $s_2 = (b_0, b_1, \cdots)$ it outputs the infinite sequence $m(s_1, s_2) = (a_0, b_0, a_1, b_1, \cdots)$. That is, it alternates the elements of the sequences. If either or both of the inputs is finite, then it alternates their elements until the shorter of the two runs out of elements, and then it outputs the remaining values from the longer of the two. For example, if $s_1 = (a_0, a_1, \cdots, a_n)$ is finite, but s_2 is infinite, then it produces

$$m(s_1, s_2) = (a_0, b_0, a_1, b_1, \cdots, a_n, b_n, b_{n+1}, b_{n+2}, \cdots) .$$

If s_2 is also finite, but longer than s_1, then the result will be similar to the above, but finite, ending with the last value of s_2

Solution.

(a) Monotonic and continuous.

(b) Not monotonic or continuous.

(c) Not monotonic or continuous.

(d) Monotonic but not continuous.

(e) Neither monotonic nor continuous.

□

4. A PN model in Ptolemy II executes until one of the following occurs:

- All processes are blocked on reads of input ports. This is called a **true deadlock**.
- The Stop actor reads a true-valued input. This actor can be found in the ExecutionControl sublibrary of of the FlowControl library.
- A buffer overflows. This occurs when the number of unconsumed tokens on a communication channel exceeds the value of the *maximumQueueCapacity* parameter of the director. Note that if you set *maximumQueueCapacity* to 0 (zero), then this will not occur until the operating system denies the Ptolemy system additional memory, which typically occurs when you have run out system memory.
- An exception occurs in some actor process.

These are the only mechanisms for stopping an execution. In this problem, we explore how to use these mechanisms to deterministically halt the execution of a PN model. Specifically, in each case, we consider a Source actor feeding a potentially infinite sequence of data tokens to a Display actor. We wish to make this sequence finite with a specific length, and we wish to ensure that the Display actor displays every element of the sequence.

(a) Suppose that you have a Source actor with one output port and no parameters whose process iterates forever producing outputs. Suppose that its outputs are read by a Display actor. Find a way to use the Stop actor to deterministically stop the execution, or argue that there is no way to do so. Specifically, the Source actor should produce a fixed number of outputs, and every one of these outputs should be consumed and displayed by the Display actor before execution halts.

(b) Most Source actors in Ptolemy II have a *firingCountLimit* parameter that limits the number of outputs they produce. Show that this can be used to deterministically halt the execution without the help of a Stop actor.

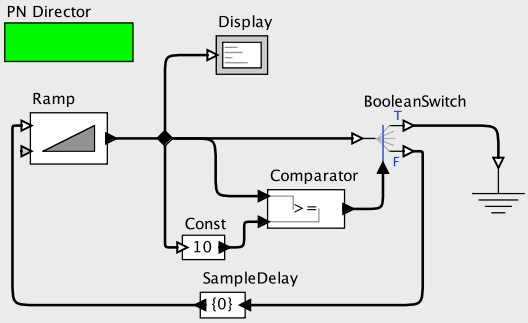
(c) Many Source actors in Ptolemy II have *trigger* input ports. If these inputs are connected, then the actor process will read a value from that input port before producing each output. Show how to use this mechanism, with or without the Stop actor, to achieve our goal of deterministically halting execution, or argue that it is not possible to do so.

Solution.

(a) With some care, the Stop actor can be used to deterministically stop execution of model after a fixed amount of data has been *produced*, but it cannot ensure that the data is *consumed*. In particular, there is no way to ensure that the Display actor will consume (and hence display) all its inputs before the model halts.

(b) This is straightforward. Just set the *firingCountLimit* parameter to some finite value and the model stops executing after the specified amount of data has been produced *and consumed*. The reason that the data is also consumed is that the model will halt only when all processes have either exited or have become blocked on a read. The Display actor does not become blocked on a read until it has displayed all previously available data

(c) A trigger input can be used to produce a finite input sequence, thus starving the model, and halting it after deterministically displaying a finite number of results. For example, the following model will deterministically display the integers 0 to 10, then halt:



This model deterministically halts after displaying 11 numbers from the Ramp.

□

5. This exercise explores the implementation of an all-to-all scatter/gather in Ptolemy II. Specifically, construct a model that generates four arrays with values:

{”a1”, ”a2”, ”a3”, ”a4”}
{”b1”, ”b2”, ”b3”, ”b4”}
{”c1”, ”c2”, ”c3”, ”c4”}
{”d1”, ”d2”, ”d3”, ”d4”}

and converts them into arrays with values

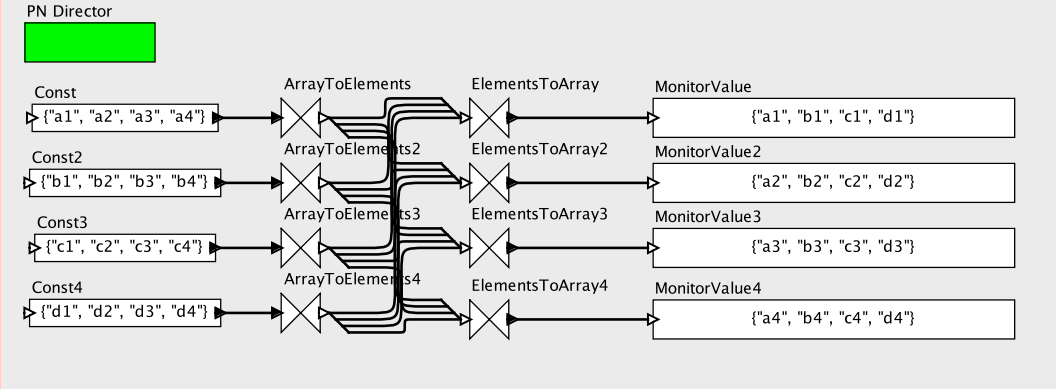
{”a1”, ”b1”, ”c1”, ”d1”}

{”a2”, ”b2”, ”c2”, ”d2”}
{”a3”, ”b3”, ”c3”, ”d3”}
{”a4”, ”b4”, ”c4”, ”d4”}

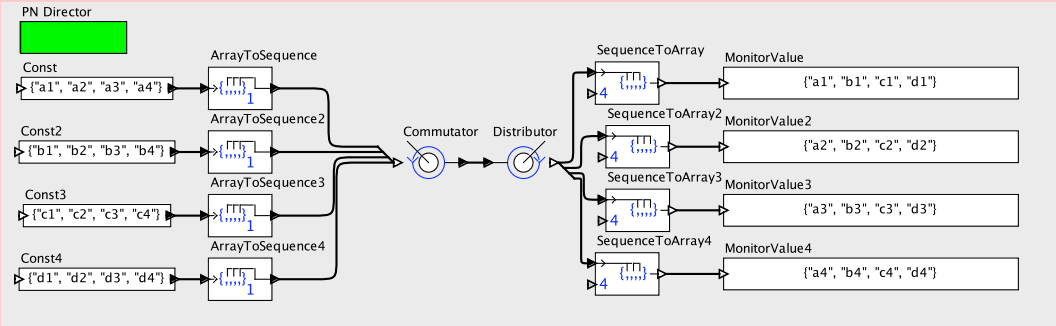
Experiment with the use of ArrayToElements and ElementsToArray, as well as ArrayToSequence and SequenceToArray (for the latter, you will also likely need Commutator and Distributor). Comment about the relative merits of your approaches. **Hint:** You will likely have to explicitly set the widths of the connections to 1. Double click on the wires and set the value.

You may also experiment with MultiInstanceComposite, but be advised that there appear to be some concurrency bugs at this time that keep this higher-order actor from working as expected.

Solution. Below is an implementation using ArrayToElements and ElementsToArray:



Below is an implementation using ArrayToSequence and SequenceToArray:



In the above, the Distributor is set to use a *blockSize* of 4.
The latter solution has the advantage of more readable connections, at least in this graphical syntax. □