

Homework 2

EE 290n - Advanced Topics in Systems Theory

Edward A. Lee

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 $A = \text{seconds, grams, meters, pounds, yards}$ 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 ∞).
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

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

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

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 $m: (A^{**})^2 \rightarrow A^{**}$ be the **fair alternating merge** function, defined as follows. Given two infinite sequences $s_1 = (a_0, a_1, \dots)$ and $s_2 = (b_0, b_1, \dots)$ it outputs the infinite sequence $m(s_1, s_2) = (a_0, b_0, a_1, b_1, \dots)$. 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 $s_1 = (a_0, a_1, \dots, a_n)$ is finite, but s_2 is infinite, then it produces

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

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

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 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.
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.