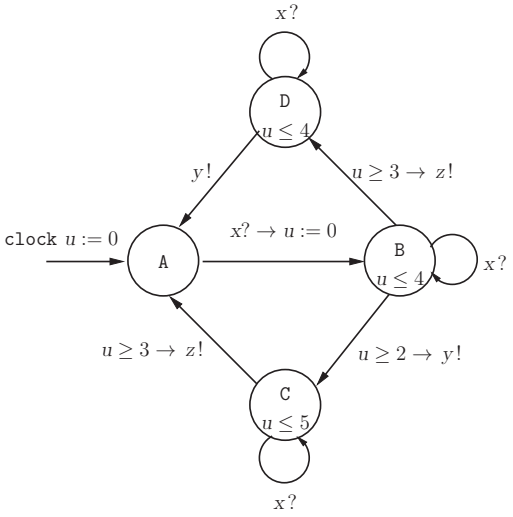
7 Timed Model

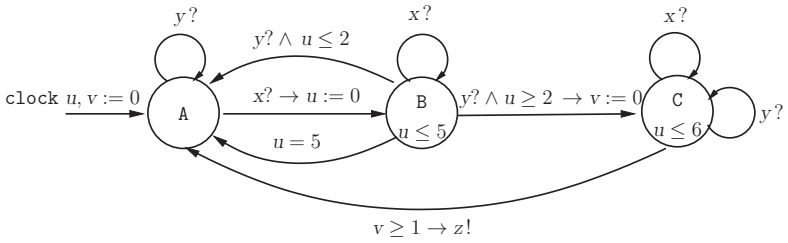
Solution 7.1: The desired behavior is specified by the timed state machine below. In the initial mode A, when the process receives an input event x, it switches to mode B, and the value of the clock variable u indicates the time elapsed since this mode-switch. The process can issue the output event y and switch to mode C when at least 2 time units have elapsed, and can issue the output event z and switch to mode D when at least 3 time units have elapsed. In mode C, the process generates the output event z returning to the initial mode (the clock-invariant $u \leq 5$ associated with mode C along with the guard $u \geq 3$ associated with the mode-switch from C to A enforce the desired bounds on the delay with respect to the relevant input event). Similarly, in mode D, the process generates the output event y returning to the initial mode. Note that when the process enters mode D, at least 3 time units have elapsed since the occurrence of the relevant input event, and thus, the lower bound on the delay for producing the output event y holds automatically.



■

Solution 7.2: The desired behavior is specified by the timed state machine below. In the initial mode, if the process receives an input event y, the state stays unchanged, and if it receives an input event x, it switches to mode B, where the value of the clock variable u indicates the time elapsed since this mode-switch. The clock-invariant of mode B ensures that the process can wait in this mode only for 5 time units, and if no event y is received until the condition $(u = 5)$ holds, the process returns to the initial mode. When the process receives an input event y while in mode B, if the condition $u \leq 2$ holds it returns to the initial mode, and otherwise, it resets the clock v and switches to mode C. The clock-invariant associated with mode C along with the guard associated with

the mode-switch from `C` to `A` enforce the desired bounds on the delay between the output event z and the relevant input events x and y.



■

Solution 7.3: The process maintains three Boolean variables: a keeps track the value of the input channel x, b the value of the input channel y, and c the value of the desired output z. It also has one clock variable u. The Boolean state variables are initialized to 0 at the beginning. The process can be in three modes `Idle`, `Wait1`, and `Wait2`. The initial mode is `Idle`. Whenever an input event x (or y) occurs, it toggles the value of the corresponding state variable a (or b, respectively). Then it checks if this changes the value of $a \vee b$: if it does not, then the process remains in the idle mode, otherwise it switches to mode `Wait1` and resets the clock u to schedule a change in the output variable c. The process can stay in the mode `Wait1` for at most 1 time unit. While in `Wait1`, if another input event is received and this causes a change in the value of the condition $a \vee b$ back to the old value, the process cancels the scheduled output change and goes back to the idle mode. Once the time delay of 1 time unit is up, the process switches to mode `Wait2`, in which it is too late for any change in the condition $a \vee b$ to cancel the scheduled change in output value. While it is waiting in mode `Wait2`, if the process detects any input event, it simply toggles the corresponding state variable. In mode `Wait2`, once the clock exceeds 2, the process can execute an internal transition that toggles the output variable c, and switches back to either `Idle` or `Wait1` depending on whether or not the output variable is consistent with the inputs. The clock-invariant ensures that the process can wait in `Wait2` only as long as the clock does not exceed 4.

The declaration of state variables is given by

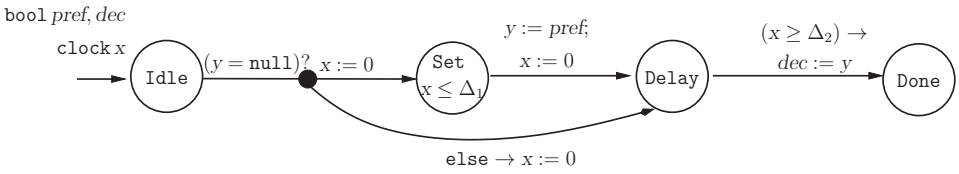
$$\texttt{bool}\ a := 0;\ b := 0;\ c := 0;\ \{\texttt{Idle}, \texttt{Wait1}, \texttt{Wait2}\}\ \mathit{mode} := \texttt{Idle};\ \texttt{clock}\ u := 0.$$

The input task for processing the input channel x is given by

$$\begin{array}{l} a := \neg\, a; \\ \texttt{if}\ (\mathit{mode} = \texttt{Idle}\ \wedge\ c \neq a \vee b)\ \texttt{then}\ \{\, u := 0;\ \mathit{mode} := \texttt{Wait1}\,\} \\ \texttt{else if}\ (\mathit{mode} = \texttt{Wait1}\ \wedge\ c = a \vee b)\ \texttt{then}\ \mathit{mode} := \texttt{Idle}. \end{array}$$

The task for processing the input channel y is analogous but toggles the state variable b first. The output task for the channel z is always enabled and simply

Solution 7.8: The protocol uses a single shared atomic register y ranging over $\{\texttt{null}, 0, 1\}$ initialized to `null`. The protocol executed by each process P is shown below. The variable *pref* contains the preference value of the process. When it starts executing the protocol, it first reads the shared register y. If y still has its initial value `null`, then the process writes its own preference to y. The clock-invariant $x \leq \Delta_1$ ensures that the delay between a process finding y to be `null` and writing its own preference to y is at most Δ_1. If initially the process finds y to be non-null, it skips the writing step and proceeds to `Delay` straight away. In mode `Delay`, the process waits for at least Δ_2 time units, reads the shared register y, chooses it be the decision value, and terminates by switching to the mode `Done`.



A process executes at most two read actions, one write action, and delays itself by at least Δ_2 time units. Thus it is not blocked by other processes and requirement of wait-freedom is satisfied. If a process finds the value of y to be `null`, it writes its own preference to y (which may be overwritten by preferences of other processes). Thus, when a process executes the read operation $dec := y$ before terminating, the value of y is guaranteed to be the preference of one of the processes, and thus, the requirement of validity is also satisfied. Assuming $\Delta_2 > \Delta_1$, the agreement requirement is satisfied. The reasoning is analogous to the one in Fischer's timing-based mutual exclusion protocol. Suppose a process enters mode `Delay` at time t_1 and leaves it at time t_2. Because of the guard-condition of the mode-switch out of the mode `Delay`, we know that $t_2 - t_1 \geq \Delta_2$. The process decides on the value of y it reads at time t_2. We argue that the value of y stays unchanged after time t_2. The value of y at time t_1 must be non-null. Thus a process that reads y for the first time after time t_1 will not write to it, and thus cannot cause the value of y to change. Consider a process that enters the mode `Set` at time $t' \leq t_1$. Because of the upper bound on how long a process can stay in `Set`, it must write to y at time no later than $t' + \Delta_1$, which is no later than $t_1 + \Delta_1$, which must be strictly smaller than t_2 if $\Delta_2 > \Delta_1$. Thus if some process decides on the value of y that it reads at time t_2, then no process writes to y after time t_2. This implies that the decision values are identical for all processes. ■

Solution 7.9: The table below shows the first few steps of a possible execution of the protocol when the message string 100110100 is supplied to the sender at time 0, where the error rate ϵ equals 0.25. It uses the same notational conventions as the ones used in figure 7.15. Note that the delay between the first two *up* events issued by the sender process is only 4.5 time units instead of 6 due to the skew. As a result the receiver can interpret the second bit as 1. At

the number of partitions remains finite, but now even when x exceeds 2 and y exceeds 1, clock-valuations belonging to the same partition agree on whether or not a constraint such as $x - y \leq 1$ holds. Formally, we modify the definition of region equivalence (page 322) so that requirement 1 stays unchanged, but requirement 2 says that: for every pair of clock variables x and y, the fractional part of $\nu(x)$ is less than or equal to the fractional part of $\nu(y)$ if and only if the fractional part of $\nu'(x)$ is less than or equal to the fractional part of $\nu'(y)$. This creates a finer partitioning, but there are still only finitely many regions. The proof of theorem 7.1 now holds even when the enabledness of an action depends on the truth of constraints of the form $x - y \leq k$. ■

Solution 7.15: 1. The DBM corresponding to given constraints is shown below:

$$\begin{bmatrix} 0 & -3 & 0 \\ 4 & 0 & 6 \\ \infty & -1 & 0 \end{bmatrix}$$

2. The DBM is not canonical. In particular, from $x_2 - x_1 \leq -1$ and $x_1 - x_0 \leq 4$, we can conclude that $x_2 - x_0 \leq 3$ and tighten $[2, 0]$ entry to 3. The canonical DBM is given by:

$$\begin{bmatrix} 0 & -3 & 0 \\ 4 & 0 & 4 \\ 3 & -1 & 0 \end{bmatrix}$$

3. To capture the effect of elapse of time, in the DBM above, we set the $[1, 0]$ entry to ∞ and $[2, 0]$ entry to 5, and then canonicalize. The resulting DBM is:

$$\begin{bmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & -1 & 0 \end{bmatrix}$$

4. To capture the guard $x_1 \geq 7$, in the DBM above we set the entry $[0, 1]$ to -7. Canonicalization changes the entry $[0, 2]$ to -3 (this reflects that the guard-condition $x_1 \geq 7$, together with other constraints, implies $x_2 \geq 3$). To set x_1 to 0, we change $[0, 1]$ and $[1, 0]$ entries to 0, $[1, 2]$ and $[2, 1]$ entries to ∞, and canonicalize giving the result:

$$\begin{bmatrix} 0 & 0 & -3 \\ 0 & 0 & -3 \\ 5 & 5 & 0 \end{bmatrix}$$

■

Solution 7.16: Upon entering the mode `A`, all clocks equal 0. Thus all entries in the DBM R_A equal 0. To capture the effect of elapse of time in mode `A`, in the matrix R_A, we set $[1, 0]$ entry to 5 due to the clock-invariant $x_1 \leq 5$, set $[2, 0]$ entry to 3 due to the clock-invariant $x_2 \leq 3$, and set $[3, 0]$ entry to ∞ due

to lack of explicit upper bound on x_3. Canonicalization gives the DBM R'_A:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{bmatrix}$$

To intersect DBM R'_A with the guard-constraint $x_3 \geq 2$, we set the $[0,3]$ entry to -2, and canonicalize. Then to capture the effect of resetting clock x_2 to 0, we update its lower and upper bounds and canonicalize. The resulting DBM R_B is:

$$\begin{bmatrix} 0 & -2 & 0 & -2 \\ 3 & 0 & 3 & 0 \\ 0 & -2 & 0 & -2 \\ 3 & 0 & 3 & 0 \end{bmatrix}$$

To capture the effect of elapse of time in mode `B`, in the matrix R_B, we set $[1,0]$ entry to ∞, $[2,0]$ entry to 2, and $[3,0]$ entry to 6. Canonicalization gives the DBM R'_B:

$$\begin{bmatrix} 0 & -2 & 0 & -2 \\ 5 & 0 & 3 & 0 \\ 2 & -2 & 0 & -2 \\ 5 & 0 & 3 & 0 \end{bmatrix}$$

To intersect DBM R'_B with the guard-constraint $x_1 \geq 3$, we set the $[0,1]$ entry to -3, and canonicalize. Then we update entries $[0,3]$ and $[3,0]$ to 0, $[1,3]$, $[3,1]$, $[2,3]$ and $[3,2]$ entries to ∞, and canonicalize to obtain the DBM R_C:

$$\begin{bmatrix} 0 & -3 & 0 & 0 \\ 5 & 0 & 3 & 5 \\ 2 & -2 & 0 & 2 \\ 0 & -3 & 0 & 0 \end{bmatrix}$$

Finally, to capture the effect of elapse of time in mode `C`, in the matrix R_C, we set $[1,0]$ entry to 8, $[2,0]$ entry to ∞, and $[3,0]$ entry to ∞. Canonicalization gives the DBM R'_C:

$$\begin{bmatrix} 0 & -3 & 0 & 0 \\ 8 & 0 & 3 & 5 \\ 6 & -2 & 0 & 2 \\ 5 & -3 & 0 & 0 \end{bmatrix}$$

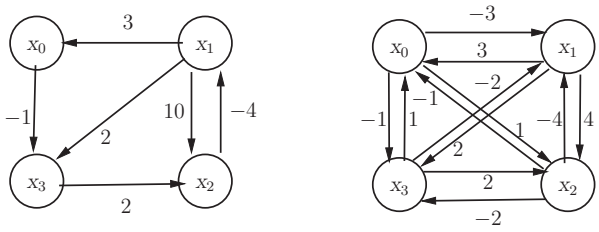
■

Solution 7.17: Note that the clock x_0 is implicitly always 0. Thus the entries in the 0th row and 0th column give bounds on the values of each clock: $x_j \leq R[j,0]$ and $-x_j \leq R[0,j]$. When the clock x_i is reset to 0, its new value coincides with the value of x_0. For each clock x_j, we thus want the entries $R[j,i]$ and $R[i,j]$ to capture bounds on the value of clock x_j. This is achieved by executing the following code which updates the bounds in the ith row and ith column:

For $j = 0$ to m do { $R[j, i] := R[j, 0]$; $R[i, j] := R[0, j]$ }.

■

Solution 7.18: The graph representation of the constraints is shown in the figure below on left. After running the shortest-path algorithm, we get the graph on right that reflects the canonicalized version. Observe that this corresponds to the constraints $x_1 = 3$, $x_2 = -1$, and $x_3 = 1$.



■

Solution 7.19: The set `Bounds` now contains the symbolic constant ∞ and pairs of the form (k, b), where k is an integer and $b \in \{0, 1\}$. The comparison operation over integers is extended to the set `Bounds` in the following manner: for every integer k, $(k, 0) < \infty$ and $(k, 1) < \infty$, and $(k, 0) < (k, 1)$, and for two integers k and k' with $k < k'$, $(k, b) < (k', b')$ for all b and b'. Note that for any two distinct elements a and b in `Bounds`, either $a < b$ or $b < a$, and this defines the minimum: if $a < b$ then $min(a, b) = a$ else $min(a, b) = b$. Addition is defined by the following rule: for every element a in `Bounds`, $a + \infty = \infty + a = \infty$, and $(k, b) + (k', b') = (k + k', b \wedge b')$. That is, to add (k, b) and (k', b'), we add the integer parts k and k' and set the bit to 1 if both the bits equal 1. This means that combining two inequalities, one of which is a strict one, results in a strict inequality.

The given constraints are represented by the following DBM:

$$\begin{bmatrix} (0,1) & (-3,0) & (0,1) \\ (6,1) & (0,1) & (4,0) \\ \infty & (-1,1) & (0,1) \end{bmatrix}$$

This matrix is not canonical. The corresponding canonical DBM is shown below and reflects the implied constraint $x_2 \leq 5$.

$$\begin{bmatrix} (0,1) & (-3,0) & (0,1) \\ (6,1) & (0,1) & (4,0) \\ (5,1) & (-1,1) & (0,1) \end{bmatrix}$$

■