Channels

Exercise 1 The perfect point-to-point links abstraction allows messages from one sender to arrive at a receiver in a different order than they were sent. Some applications rely on *first-in first-out* (FIFO) order communication, however. Specify a FIFO-order perfect point-to-point links abstraction which ensures, in addition to the guarantees of perfect point-to-point links, that messages are not reordered.

Solution A specification of *FIFO-order perfect point-to-point* links is shown below.

Module:
  Name: FIFOPerfectPointToPointLinks, **instance** *fpl*.
Events:
  Request: *<fpl, Send | q, m>* : Requests to send message *m* to process *q*.
  Indication: *<fpl, Deliver | p, m>* : Delivers message *m* sent by process *p*.
Properties:
  FPL1-FPL3: Same as properties PL1-PL3 of perfect point-to-point links.
  FPL4: *FIFO delivery*: If some process sends message *m1* before it sends message *m2* , then no correct process delivers *m2* unless it has already delivered *m1* .

Exercise 2 Provide an implementation of FIFO-order perfect point-to-point links (*Exercise 1*) on top of perfect point-to-point links using sequence numbers.

Solution The following algorithm, called “Sequence Number”, implements FIFO-order perfect point-to-point links on top of perfect point-to-point links.

```
Implements:
  FIFOPerfectPointToPointLinks, instance fpl.
Uses:
  PerfectPointToPointLinks, instance pl.

upon event <fpl, Init> do
  forall p ∈ Π do
    lsn[p] := 0;
    next[p] := 1;

upon event <fpl, Send | q, m> do
  lsn[q] := lsn[q] + 1;
  trigger <pl, Send | q, (m, lsn[q])>;
```

```
upon event <pl, Deliver | p, (m, sn)> do
  pending := pending ∪ {(p, m, sn)};
  while exists (q, n, sn') ∈ pending such that sn' = next[q] do
    next[q] := next[q] + 1;
    pending := pending \ {(q, n, sn')};
    trigger <fpl, Deliver | q, n>;
```

Exercise 3 Does the following statement satisfy the synchronous-computation assumption? "On my server, no request ever takes more than one week to be processed."

Solution The answer is yes. This is because the time it takes for the server to process a request is bounded and known, one week.

Exercise 4 In a fail-stop model, which of the following properties are safety properties?

1. every process that crashes is eventually detected;
2. no process is detected before it crashes;
3. no two processes decide differently;
4. no two correct processes decide differently;
5. every correct process decides before t time units;
6. if some correct process decides, then every correct process decides.

Solution

1. *Every process that crashes is eventually detected.* This is a **liveness** property; we can never exhibit a time t in some execution and state that the property is violated. There is always the hope that eventually the failure detector detects the crashes.
2. *No process is detected before it crashes.* This is a **safety** property. If a process is detected at time t before it has crashed then the property is violated at time t.
3. *No two processes decide differently.* This is also a **safety** property, because it can be violated at some time t and never be satisfied again.
4. *No two correct processes decide differently.* Since a correct process is a process that never crashes and executes an *infinite* number of steps, the set of correct processes is not known at any given point in time. After any partial execution (i.e. after any *finite* number of steps), we cannot declare a process correct, because it might crash in the future. Therefore, this property is a **liveness** property. Even if two processes have decided differently in some partial execution, the property still might be satisfied in the future by (at least) one of the two processes crashing.

   Note: If we assumed that the set of correct processes was known a priori, this would be a safety property: once two correct processes have decided differently, the property would be violated.
5. *Every correct process decides before t time units.* Applying the same reasoning about the correctness of processes as in 4., this is also a **liveness** property. If every process that has not decided before time t eventually crashes, the property is satisfied. The above note also applies.
6. *If some correct process decides then every correct process decides.* This is a **liveness** property: there is always the hope that the property is satisfied. It is interesting to note that the property can actually be satisfied by having the processes not do anything. Hence, the intuition that a safety property is one that is satisfied by doing nothing may be misleading.