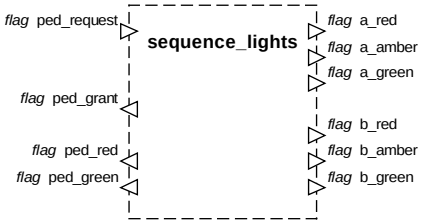
8.2 A Compound System: a Traffic Light Controller



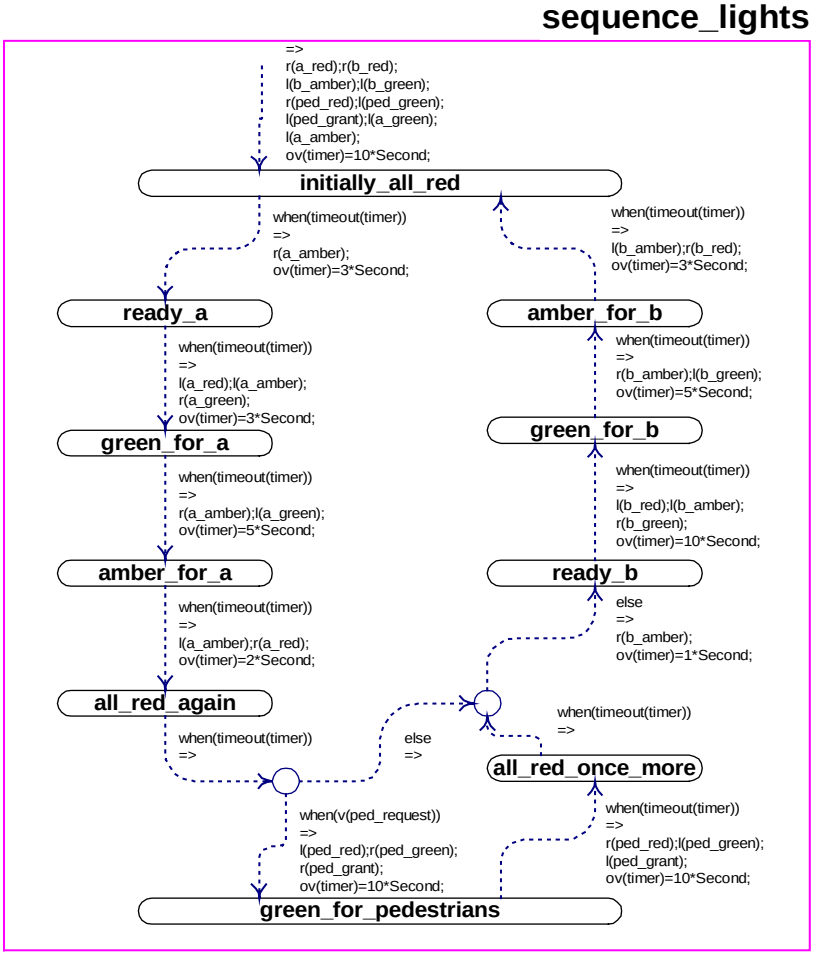


Figure 17: A Traffic Light Sequencer.

The basic sequence of traffic lights is presented in Figure 17. It uses a set of flags to control the lights. The lights are assumed to operate so that the value false means dark and the value true means light. The crossing directions are called a and b. The opposite directions are signalled equally. The pedestrians of both directions are shown green only, when the cars of both directions are shown red.

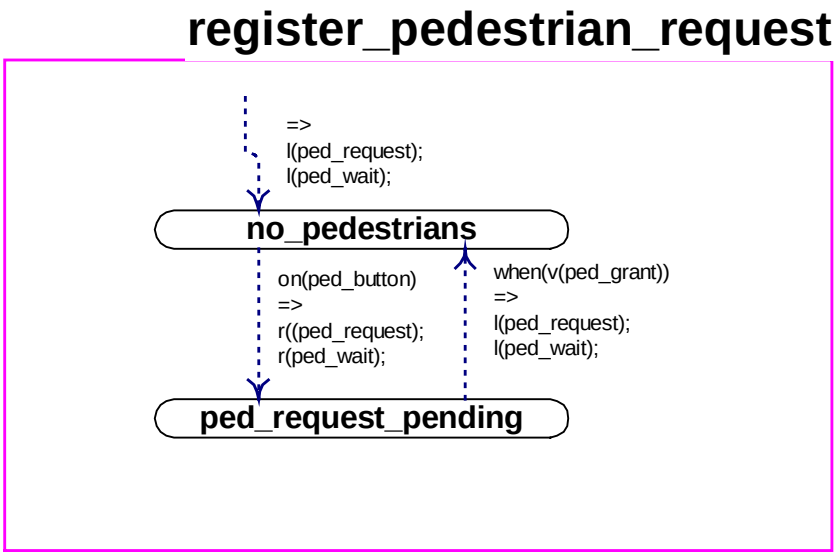
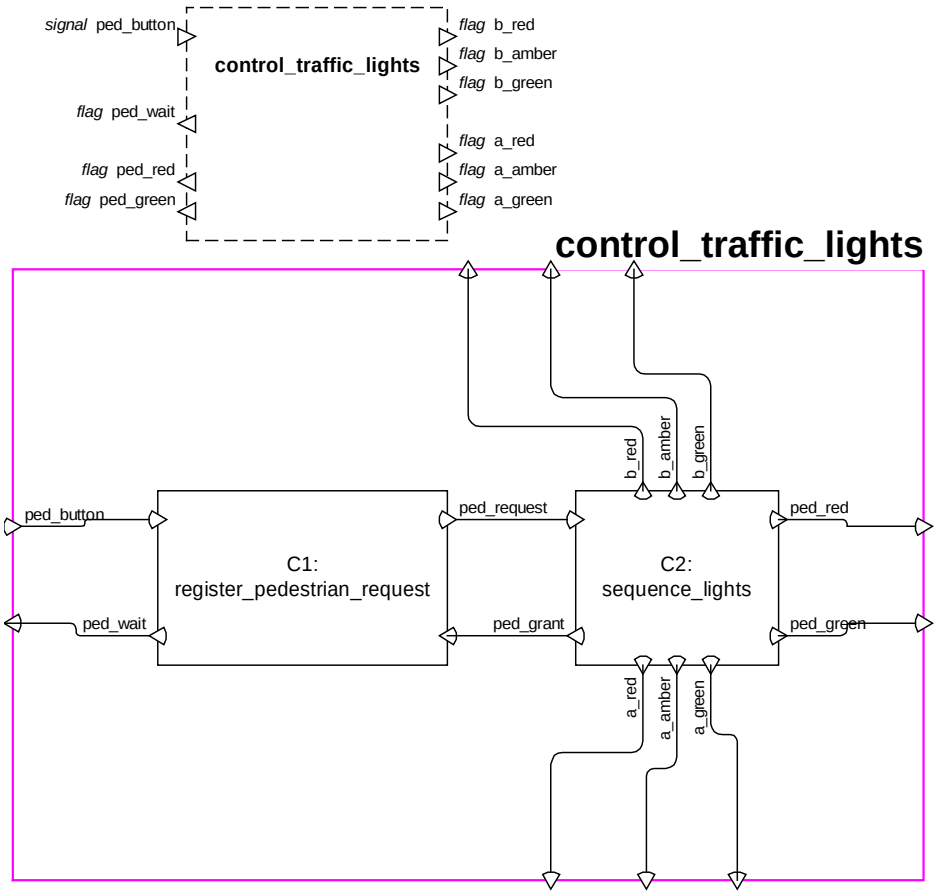


Figure 18: A Pedestrian Button Registration.

Registration of pedestrian passing requests is presented in Figure 18. A signal `ped_button` is received every time a pedestrian presses a request button. The flag `ped_grant` is high, when the pedestrians are allowed to pass. The logic is arranged so, that `ped_button` raises the request flag `ped_request` and lights the wait light by raising `ped_wait` . The wait light is

lit until the pedestrians are let to pass. If the button is pressed while there is a green light for pedestrians, a virtual flash of the wait light takes place. However, the flash does not take *real* time, it cannot be seen, and it does not stress the power electronics or the light bulb.




```
include xtlight2.h
include xtlight3.h
```

Figure 19: A Traffic Light Controller.

Figure 19 shows an architecture diagram. It combines the state machines and makes them a system (a system is just a new component type).

As in state transition diagrams the diagram needs a component body symbol and component specification symbol for the new component type defined by the diagram.

The state machines are represented by component (an actor) instances whose label consists of instance name and component type name.

A connection of ports of the intercommunicating components is drawn using two selectors and a connector. Selectors (sector symbols) which are snapped into component instances select ports to be connected. A connector symbol connects the matching selectors together. The direction of a selector is presented by the direction of sectors sharp corner and it must match the direction of the selected port.

A connection of a port of a component to the port of the external interface is drawn using two selectors and a connector. Selectors (sector symbols) select ports to be connected. One selector is snapped to the component and another is snapped to the component body symbol framing the diagram. A connector symbol connects the matching selectors together. The direction of a selector is presented by the direction of sectors sharp corner and it must match the direction of the selected port.

The actor components in an architecture diagram represent independent, concurrent subprocesses. Those subprocesses communicate by discrete and continuous communication or shared stores. A signal is a special type of discrete communication who has no data value. A flag is a special case of continuous communication whose datatype is R_boolean. In addition, direct control by switching another subprocess on or of is possible. **Other kinds of inter-process communication are not recommended** (global variables, OS calls, etc.)**.**

To follow the subsequent walkthrough example you may take paper copies of the diagrams and use paper clips on the diagrams to show the current states and flags being high.

Lets consider the case when green is shown to the b direction. Moreover, nobody has pressed the button since last green for pedestrians. The flag `b_green` is high (true) and all the others are down (false). Now, a pedestrian presses a button. Let's assume that pressing the button down, rather than releasing it, causes the signal `ped_button` to be injected to the system. From the architecture diagram in the figure 19 the signal goes to the bubble `register_pedestrian_request`.

The state machine `register_pedestrian_request` in the Figure 18 is in the state `no_pedestrians` since button has not yet been pressed. Now, the

signal ped_button triggers the transition to ped_request_pending. The transition lits the wait light and raises the ped_request flag.

The state machine sequence_lights in Figure 17 is showing green light to b direction in the state green_for_b. It does not immediately respond to the new value of the flag.

The machine sequence_lights steps thru the light sequence controlled by the timer until it has shown enough all reds in the state all_red_again. When the time is out and the machine leaves that state, a decision is made depending on the value of the flag ped_request. Because the flag is high, the path to the state green_for_pedestrians is taken, ped_green is lit, and ped_grant is raised.

The state machine register_pedestrian_request is still in the state ped_request_pending. It responds immediately to raising of the ped_grant by dropping the ped_request and turning off the wait light.

New pedestrian requests cannot be registered until red is lit for pedestrians again.

The light sequence continues controlled only by the timer until the sequence_lights is leaving the state all_red_again again.