Concurrency Exercises 6: General exercises

This sheet consists of questions on a wide variety of topics in concurrency: Modelling, process algebra, and properties of concurrent systems.

1. Simple example:
Write a very simple model of a VCR machine can only do play and stop actions. Model the VCR as a process, VCR, using FSP, and draw the LTS.

2. Choice:
Extend the model of the VCR to include a pause action and draw the LTS. One can only pause after a play action, and it can be followed by either a play action or a stop action. Check that the VCR can perform the actions given by the trace:

play -> stop -> play -> pause -> play -> pause -> stop

3. Choice:
Draw the LTS for the MICROWAVE FSP process below. What is the alphabet of the process, how many states and transitions are there?

```
MICROWAVE = ( put_food_in -> SETTINGS ),
SETTINGS = ( set_heat_level -> set_time -> COOK |
             set_time     -> set_heat_level -> COOK ),
COOK   = ( cook   -> take_food_out -> MICROWAVE ).
```

4. Non-determinism:
Draw the LTS for the LIBRARY FSP process below. What is the alphabet of the process, how many states and transition are there?

```
LIBRARY = ( take_book_to_desk -> give_details -> take_book->OPTIONS ),
OPTIONS = ( renew          -> OPTIONS |
           return_on_time -> LIBRARY |
           return_late   -> FINE )
```
5. **Indexed Processes and Actions:**

A function takes an integer as an input and returns the square of the input. Model the function as a process `SQUARE` in FSP. Use input range `0..3` and draw the LTS. There are several different approaches on how to answer this question:

1) Indexed actions using a range
2) Indexed actions explicitly stated
3) Indexed local process and a declared `range` type.
4) Parameterised process. (Parameter must be upper case)

Answer using any two of the above approaches.

6. **Indexed Processes:**

A Celsius to Fahrenheit converter works by taking a temperature in Celsius as an input and outputs the equivalent temperature in Fahrenheit (i.e. multiply the Celsius by 9, divide by 5 and add 32). Model the converter as a process `C_F_CONVERTER` using FSP. Use an input range of `0..5`.

7. **Guarded Actions**

Draw the LTS for the `SYS` FSP process below. What is the alphabet of the process, how many states and transitions are there, and what is the process doing?

```plaintext
SYS = (in[i:0..5] -> ( when(i*i>10) keep[i] -> SYS 
        | when(i*i<10) getrid [i] -> SYS 
        | when(i*i=10) error -> ERROR )
    ).
```

8. **LTS to FSP**

Model the LTS below in FSP. There are many different ways of doing this, you may want to declare a `const` and a `range` at the beginning.
9. **Parallel Composition**

(a) Draw the composite LTS for the **SMILE** and **LUNCH** FSP processes. What are the 3 possible traces of actions that can happen? How many states and transitions are there in the LTS?

\[
\text{SMILE} = (\text{smile} \rightarrow \text{STOP} ).
\]

\[
\text{LUNCH} = (\text{eat} \rightarrow \text{drink} \rightarrow \text{STOP} ).
\]

\[||\text{LUNCH_SMILE} = (\text{LUNCH} || \text{SMILE}).\]

(b) **Process Labelling**

Using the **LUNCH** process from part (a), model a **JOINT_LUNCH** process using FSP, when two people have lunch together. What is the alphabet of the **JOINT_LUNCH** process, and how many states and transitions are there in the LTS?

10. **Process Labelling using a set of prefix labels:**

Draw the LTS for the **SYS** FSP process below. What is the alphabet of the process, how many states and transition are there?

\[
\text{A} = (a \rightarrow b \rightarrow c \rightarrow A).
\]

\[
\text{Z} = (a \rightarrow c \rightarrow Z).
\]

\[||\text{SYS(N=3)} = (g[i:1..N]:A || \{g[i:1..N]::Z}).\]

11. **Parallel composition, Action Hiding and Minimisation**

\[
\text{P} = (a \rightarrow b \rightarrow P).
\]

\[
\text{Q} = (c \rightarrow b \rightarrow Q).
\]
\[ ||S1 = ( P || Q). \]

\[ S2 = ( a \rightarrow c \rightarrow b \rightarrow S2 \]
  \[ | c \rightarrow a \rightarrow b \rightarrow S2 \]
\].

(a) Show that S1 and S2 describe the same behaviour. Hint: animate the FSP descriptions using LTSA and observe the LTS. The \( || \) operator signifies the parallel composition of the processes P and Q. Type the FSP above into LTSA, check the graphs/LTSs of S1 and S2, are they the same, do S1 and S2 describe equivalent systems, i.e. can they produce equivalent action traces? Parallel composition will create a composite LTS that forces shared actions (actions having the same name in different processes) to execute simultaneously. We say that shared actions are synchronised actions. Note: that non-shared actions are not synchronised.

(b) How many states and transitions will there be if action b is hidden in the P process, which is then composed with Q? Model this in FSP and draw the LTS.

(c) Apply the hiding operator to the above process S1, to remove action b and draw the minimised LTS.

(d) Use transition rules to calculate the computation trees for process S1 performing an a action, followed by a c action and finally a b action. What is the transition sequence for this question (hint: there are 10 rules required for this.)

12. **Relabelling**

Draw the LTS for the \( P \text{ AND} Q \) FSP process below. What is the alphabet of the process, how many states and transitions are there?

\[ P = (x \rightarrow y \rightarrow P). \]
\[ Q = (x \rightarrow y \rightarrow Q). \]
\[ ||PANDQ = (a[i:1..2]:P || Q)/{a[i:1..2].x/x,a[i:1..2].y/y}. \]

13. **Alphabet Extension**

Draw the LTS for the S1 FSP process below. What effect has the alphabet extension on Q had on the LTS of S1?
What is the alphabet of S1? (Hint: First draw the LTS for P and Q ignoring the alphabet extension. Then attempt to draw the LTS for S1)

\[ P = ( b \rightarrow P \]
  \[ | a \rightarrow e \rightarrow P \]
  \[ | c \rightarrow h \rightarrow P). \]
\[ Q = ( c \rightarrow g \rightarrow Q \]
  \[ | d \rightarrow f \rightarrow Q) +\{a,h}\].
14. Model the **MICROWAVE** example from above, this time using parallel composition.
(hint: You will need to use handshaking with shared actions, so that it is not possible to produce silly action traces. eg to `cook` after `take_food_out`. There are many ways of doing this, you may want to use the following three processes: `COOK`, `SET_HEAT` and `SET_TIME`.)

15. **Safety**
Draw the LTS for
property `FRIEND=(come->tea->leave->FRIEND)`.  
(Hint: there are 9 transitions)

16. **Safety**

Draw the LTS for the **SAFETY** FSP process below. What action trace violates the safety property?

property `SAFETY = (a -> (b -> SAFETY|a -> SAFETY)|b -> a -> SAFETY)`.

17. **Liveness**

How many of the following progress properties are violated and why?

```
START = ( select -> BUFF_ONE
         | select -> BUFF_TWO),
BUFF_ONE = (in[i:0..3] -> out[i] -> BUFF_ONE),
```

progress ONE = {out[0], in[1], out[2]}
progress TWO = {in[3]}
progress THREE = {in[1], in[3], out[2], in[5]}
progress FOUR = {in[3], out[4], out[2], in[5]}
progress FIVE = {out[4], in [5]}

18. The following 3 processes describe the actions that students undertake. Use the parallel composition `|` operator to create a process **SYS** modelling the concurrent execution of these different processes. Modify the individual processes (using handshaking with shared actions) so that it is not possible to produce silly action sequences, for example that allow a student to go to lectures undressed, or, that allow a student to perform PLAY actions before WORK actions have completed.

```
PLAY = ( pub       -> PLAY
        | gig       -> PLAY
```

||S1 = ( P | Q).
19. The Rambling Bramble Ornate Garden society require a new electronic system to monitor and control the flow of visitors through its main attraction - Wiggly Park. This garden has a number of controlled areas, including the deer park, the ornamental garden and tea-room, each gated by a one-way turnstile. For simplicity of modelling, we will only consider those three areas for this prototype control system. Visitors may enter Wiggly Park via a West entrance into a deer park, or via an East entrance directly into the Tea-rooms. Visitors may leave via the North exit from the deer park, the South exit from the ornamental garden area, or the East exit from the Tea-rooms. There are also one-way turnstiles to control movement between (i) the deer park and the ornamental garden, and (ii) the ornamental garden and the Tea-rooms.

Arrivals into and departures from the areas are signalled by the turnstiles to a controller, which maintains counts of the numbers of visitors in the three areas. At opening time, the Park Director signals the controller that Wiggly Park is open. The controller must then permit flow through the three Wiggly Park areas subject to specific limits for each of the areas. At closing time, the Park Director signals the controller that Wiggly Park is to close. The controller in turn stops further visitors entering Wiggly Park and in addition closes the turnstile from the ornamental garden to the Tea-rooms.

A generic turnstile can be modelled as follows.

```
|Turnstile| = (start -> Working),
Working| = (pass -> Working | stop -> Turnstile).
```

Instances of these can be created for each specific turnstile. For example, for the west entrance to Wiggly Park (into the deer park) and the south exit from the ornamental garden, one could have:

```
||WestParkIn| = ((deerpark.west:(Turnstile/{arrive/pass}))/
{open/deerpark.west.start,
deerpark.stopEntrance/deerpark.west.stop})).

||SouthGardenOut| = ((garden.south:(Turnstile/{leave/pass}))/
{open/garden.south.start,
garden.stopExit/garden.south.stop})).
```
Note how the WestParkIn process has renamed the pass action of the Turnstile process as arrive and then labels the whole by deerpark.west. Explain the other relabellings.

Create similar instantiated processes for the North exit from the deer park, the East Tearoom entrance and exit, and the four turnstiles controlling flow between deer park and ornamental garden, and the ornamental garden and the tea-rooms.

Then, the instantiated processes can all be composed in parallel to model the set of gates.

||Gates = ( WestParkIn
|| SouthGardenOut
|| EastTearoomIn
|| EastTearoomOut
|| ParkToGarden
|| GardenToPark
|| TearoomToGarden
|| GardenToTearoom ).

Now model a generic area controller that be appropriately instantiated for the deer park, the ornamental garden and the tea-rooms. You might assume that these controllers use the follow signals (actions): arrive, leave, stopEntrance, stopExit. Then, to obtain controllers for the specific areas, label and rename as appropriate.

Finally, create a model for the whole system via a suitable composition of turnstiles and controllers.