C.3.3 Independent section

1. The alphabets are

$$Inputs = \{feed, pet, time\ passes, absent\}$$

$$Outputs = \{purrs, bites, throws\ up, rubs, dies, absent\}$$

The state machine is shown in figure C.7. The update of the state machine is given by the following function

```
% PET - A function representing the state update of a virtual pet.
% The first argument must be in {'happy', 'hungry', 'dies'}
% The second argument must be in {'absent', 'pet', 'feed', 'time passes'}
% The two returned values are the next state of the
% pet, and the output of the state machine.

function [newstate, out] = pet(state, in)

% The default behavior is to stutter.
newstate = state;
out = 'absent';

switch(state)
case 'happy'
    switch(in)
    case 'pet'
        out = 'purrs';
    case 'feed'
        out = 'throws up';
    case 'time passes'
        newstate = 'hungry';
        out = 'rubs';
    end
case 'hungry'
    switch(in)
    case 'feed'
        out = 'purrs';
        newstate = 'happy';
    case 'pet'
        out = 'bites';
    case 'time passes'
        out = 'dies';
        newstate = 'dies';
    end
case 'dies'
    out='dies';
end
```

A program to execute the state machine is:

```
% RUNPET - Execute the virtual pet state machine

% initial state.
petstate='happy';

% loop forever.
while 1
    % Get the user input as a string.
    str=input('enter one of absent, pet, feed, time passes: ','s');

    % If the user entered quit or exit, then break the loop.
    if strcmp(str,'quit') break; end
    if strcmp(str,'exit') break; end

    % Update the pet.
    [petstate, output] = pet(petstate, str);

    % print what the pet does.
    disp(output)
    if strcmp(petstate, 'dies') break; end
end
```

Here is a sample run:

```
>> runpet
enter one of absent, pet, feed, time passes: pet
purrs
enter one of absent, pet, feed, time passes: feed
throws up
enter one of absent, pet, feed, time passes: time passes
rubs
enter one of absent, pet, feed, time passes: feed
purrs
enter one of absent, pet, feed, time passes: time passes
rubs
enter one of absent, pet, feed, time passes: pet
bites
enter one of absent, pet, feed, time passes: time passes
dies
>>
```

2. The output alphabet for the driver machine is

$$Outputs' = \{feed, time\ passes\}$$

which is a subset of *Inputs*, the input alphabet for the pet. Thus, this machine can be composed in cascade with the pet. The state machine is shown in figure C.8. The state mirror those of the pet with the same name.

The update of the state machine is given by the following function

```
% DRIVER - A function representing the state update of
% a state machine providing inputs to keep a virtual pet alive.
% The first argument must be in {'happy', 'hungry'}
% The second argument must be in {'absent', '1'}
% The two returned values are the next state of the
% driver, and the output of the state machine.

function [newstate, out] = driver(state, in)

% Alternate producing 'time passes' and 'feed'.
if ~strcmp(in, 'absent')
    switch(state)
    case 'happy'
        out = 'time passes';
        newstate = 'hungry';
    otherwise
        out = 'feed';
        newstate = 'happy';
    end
else
    % The default behavior is to stutter.
    newstate = state;
    out = 'absent';
end
```

A program to run the driver and pet for 100 iterations is

```
% DRIVEPET - Execute the virtual pet state machine composed
% in cascade with the driver state machine.

% initial state.
driverstate='happy';
petstate='happy';

% loop 100 times, since this is automatically driven.
for i=1:100,
    % Update the state of the driver and get its output.
    % The input to the driver is always '1'.
    [driverstate, petinput] = driver(driverstate, '1');
```

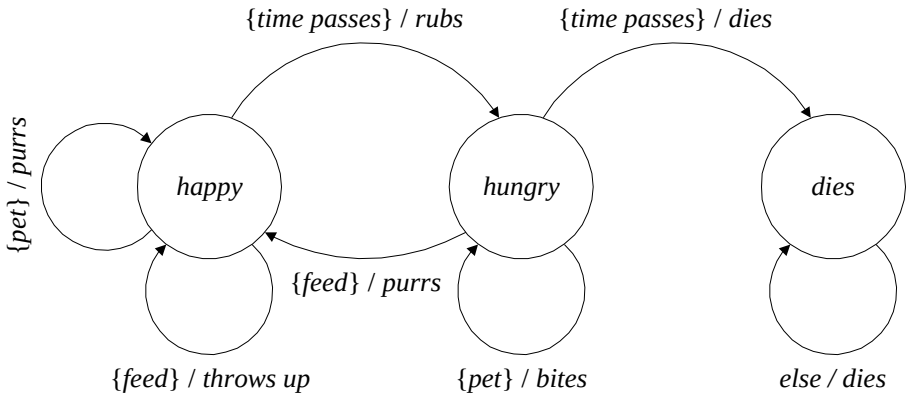

Figure C.7: A state machine for a virtual pet.

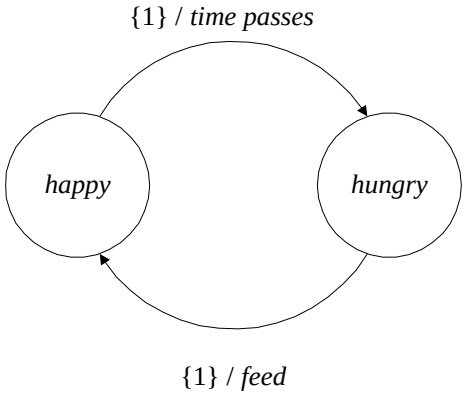

Figure C.8: A state machine that provides inputs to a virtual pet.

```
    % update the state of the pet and get its output.
    [petstate, output] = pet(petstate, petinput);

    % Display the output of the pet.
    disp(output)
end
```

A sample run is

```
>> drivepet
rubs
purrs
rubs
purrs
rubs
purrs
...
```