Solution for Assignment 1

COMP3211, 2015 Spring

Problem 1 (30%)

(a)

Assume the same specification as that of the example in the lecture. The production system could be:
Method 1

$$\bar{s_2} \rightarrow North$$
$$\bar{s_8} \rightarrow West$$
$$1 \rightarrow Nil$$

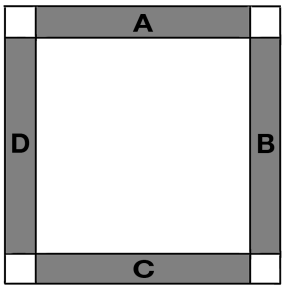
Method 2

$$inCorner \rightarrow Nil$$
$$1 \rightarrow Boundary\ Following$$

(b)

We divide the grid into two regions: the inner 8×8 grid, and the others (the outer region).

- If the initial position is in the inner 8×8 grid, then only one action is to be taken until the robot hits the boundary as inputs for the eight sensors of each cell in the inner 8×8 grid are all the same (all zero). w.l.o.g. we assume that single action is moving North. As a result, the robot will end up in region A, as illustrated below.



  Since the perceptions for cells in region A,B,C and D, respectively, are all the same, only one action is allowed for each region. If the four actions for A, B, C and D are all not going back to the region, then the robot will stay in the outer region forever, hence cannot visit every cell. If any of these four actions is stepping into the inner grid, the robot will end up in **a loop**:
  {goes into the inner region → go North till region A → goes into the inner region from exactly the same position as last time}
  Therefore it cannot visit every cell neither.

- If the initial position is in the outer region, the proof is the same as the latter part above. The robot will either stay in the outer region forever or go in and out in a loop.

(c)

Using method 1 in part (a), we can first let the robot go to the left upper corner of the grid. Once the robot is in the left upper corner, we then let it traverse the cells one by one.
Two states are needed:
PA denotes the previous action, and **N** for North, **S** for South, **W** for West and **E** for East.
Corner indicates whether the robot has been to the left upper corner. **1** for yes, **0** for no.
Initially, **PA** = Null and **Corner** = 0.
The production system is as follows:

$$\overline{\textbf{Corner}}\bar{s_2} \rightarrow North$$
$$\overline{\textbf{Corner}}\bar{s_8} \rightarrow West$$
$$\overline{\textbf{Corner}} \rightarrow Set\ \textbf{Corner} = 1, and\ \textbf{PA} = E$$
$$(\textbf{PA} = E)s_2 \rightarrow South$$
$$(\textbf{PA} = S)\bar{s_6} \rightarrow South$$
$$(\textbf{PA} = S) \rightarrow East$$
$$(\textbf{PA} = E) \rightarrow North$$
$$(\textbf{PA} = N)\bar{s_2} \rightarrow North$$
$$(\textbf{PA} = N) \rightarrow East$$
$$s_2 s_4 \rightarrow Nil$$

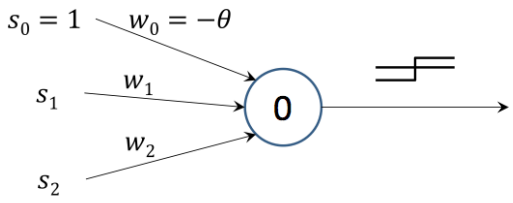
Problem 2 (10%)

$$x_1x_2 + x_1\bar{x_3}\bar{x_4} + x_2(\bar{x_3} + \bar{x_4})$$

Problem 3 (20%)

The maximal set of training instances for a logical disjunction with 2 inputs is shown below:

s_1	s_2	d
1	1	1
1	0	1
0	1	1
0	0	0



A mininal training set could be:
$\{(s_0 = 1, s_1 = 1, s_2 = 1, d = 1), (s_0 = 1, s_1 = 0, s_2 = 0, d = 0)\}$

	$W_{old} = (w_0, w_1, w_2)$	$X = (x_0, x_1, x_2)$	d	Σ	f	$d = f?$	$W_{new} = (w_0, w_1, w_2)$
1	$(0, 0, 0)$	$(1, 1, 1)$	1	0	1	*yes*	$(0, 0, 0)$
2	$(0, 0, 0)$	$(1, 0, 0)$	0	0	1	*no*	$(-1, 0, 0)$
3	$(-1, 0, 0)$	$(1, 1, 1)$	1	-1	0	*no*	$(0, 1, 1)$
4	$(0, 1, 1)$	$(1, 0, 0)$	0	0	1	*no*	$(-1, 1, 1)$
5	$(-1, 1, 1)$	$(1, 1, 1)$	1	1	1	*yes*	$(-1, 1, 1)$
6	$(-1, 1, 1)$	$(1, 0, 0)$	-1	0	0	*yes*	$(-1, 1, 1)$

After interation 5 & 6, we got the converging sequence of weights (-1,1,1), and this weight vector works correctly with all the training instances in the maximal set. Therefore, (1,1) and (0,0) are the minimal set.

Problem 4 (20%)

There is no unique solution for this question. You may get the points as long as your answer makes sense and shows that you have understood what fitness function does.

(a)

There are several factors that might be considered in control of an elevator. These are ***average waiting time*** for a passenger outside the elevator before the elevator arrives, ***average waiting time*** for a passenger inside the elevator before it arrives at the passengers desired floor, and ***maximal waiting times***. During the early stages of evolving an elevator controller, it might be that these times are infinite because some passengers may never be serviced. Therefore, we might also include factors such as ***number of passengers that are never serviced***. The fitness function itself can then be composed from these factors.
Relative merits: the number of passengers that are never serviced can be regarded as the most fundamental performance indicator of an elevator, and it also indicates the tolerance level. However, it does not indicate the efficiency of an elevator. The various waiting times have more indication about the efficiency as well as the user experience.

(b)

Here the factors might be ***average and maximal waiting times*** in the main street and the side street, respectively. Main street and side street traffics are exclusive at any point of time, so the fitness function will have to take into account the desired tradeoff. Other factor can be ***the volume of traffic*** along the main street during different periods of time(peak hours, off-peak hours).
Relative merits: The waiting times put more emphasis on the driver's experience while the volume of traffic measures the traffic condition on the whole. It makes a more comprehensive fitness function if the peak and off-peak hours are treated differently.

Problem 5 (20%)

Here is a set of features and their intended meanings:
have – wasp senses that it has the cricket
at_threshold – wasp senses that it is at the threshold of its burrow
sok – wasp senses that it is inside the burrow and everything is ok
cat – wasp senses that the cricket is at the threshold

Here is a set of actions and their effects:
go_threshold – wasp goes from wherever it is to the threshold of its burrow
go_inside – wasp goes from the threshold of its burrow to the inside of its burrow
go_outside – wasp goes from the inside of its burrow back to the threshold
drag – wasp drags the cricket inside of its burrow
find – wasp goes from wherever it is to find and pick up a cricket
drop – wasp drops the cricket at the threshold
set_ok – wasp sets condition of burrow (ok) to 1
set_nok – wasp sets condition of burrow (ok) to 0

Here is a production system (initially, **ok** = 0):

$$\begin{gathered} at\_threshold \cdot cat \cdot ok \rightarrow drag \\ sok \rightarrow set\_ok, go\_outside \\ at\_threshold \cdot cat \rightarrow go\_inside \\ at\_threshold \cdot have \rightarrow drop \\ have \rightarrow go\_threshold \\ 1 \rightarrow find, set\_nok \end{gathered}$$