Tutorials COMP3411/9814 20TO
Open learning with additional questions
Solutions

Week 1- Tutorial 2: Solutions

2.1: a) PEAS Descriptions

Present to your tutorial group the PEAS model you developed for the activity from "The PEAS Model of an Agent".

2.1: b) PAGE Descriptions

Key point: The architecture depends on the specification of the desired behaviour, e.g. a reflexive/ reactive approach might be sufficient for Robocup.

2.2: Classifying Environments & Choice of Agent Type

Use the PEAS model to classify an environment from the tutorial in Week 1.

1. Classify each task according to the properties given in lectures in the Classifying Environments page of Week 2 module. Give a one-line justification for each choice.
2. Select a suitable agent type (or, discuss the relative merits of the different agent types presented).

2.3 Sorting in Prolog

1. Write a prolog predicate `insert(Num, List, NewList)` that takes a number Num along with a list of numbers List which is already sorted in increasing order, and binds NewList to the list obtained by inserting Num into List so that the resulting list is still sorted in increasing order.

```
% Base Case
insert(New,[],[New]).

% New node goes at front of list
insert(New,[Head|Tail],[New,Head|Tail]) :-
   New =< Head.

% New node is inserted into existing list
insert(New,[Head|Tail],[Head|Tail1]) :-
   New > Head,
   insert(New,Tail,Tail1).
```

2. Write a predicate `isort(List,NewList)` that takes a List of numbers in any order, and binds NewList to the list containing the same numbers sorted in increasing order. Hint: your predicate should call the insert() predicate from part (1).

```
% Base Case
isort([],[]).

% Sort the Tail and then insert the Head
isort([Head|Tail],List) :-
   isort(Tail,Tail_Sorted),
   insert(Head,Tail_Sorted,List).
```

3. Write a `predicate split(BigList,List1,List2)` which takes a list BigList and divides the items into two smaller lists List1 and List2, as evenly as possible (i.e. so that the number of items in List1 and List2 differs by no more that 1). Can it be done without measuring the length of the

```
% Base Case 0: empty list
split([],[],[]).

% Base Case 1: list with one item
split([A],[A],[]).

% Assign first item to first list, second item to second list
% and recursively split the rest of the list.
split([A,B|T],[A|R],[B|S]) :-
   split(T,R,S).
```

4. Write a predicate `merge(Sort1,Sort2,Sort)` which takes two lists Sort1 and Sort2 that are already sorted in increasing order, and binds Sort to a new list which combines the elements from Sort1 and Sort2, and is sorted in increasing order.

```
% If one list is empty, return the other list
merge(A,[],A).
merge([],B,B).

% If first item of first list is smaller,
% it becomes first item of the merged list
merge([A|R],[B|S],[A|T]) :-
   A =< B,
   merge(R,[B|S],T).

% If first item of second list is smaller,
% it becomes first item of the merged list
merge([A|R],[B|S],[B|T]) :-
   A > B,
   merge([A|R],S,T).
```

Bonus Challenge (for students who have already studied sorting algorithms in another programming language):

Write a predicate `mergesort(List,NewList)` which has the same functionality as the isort() predicate from part (2) above, but uses the MergeSort algorithm. Hint: you will need to use the split() and merge() predicates from parts (3) and (4) above.

```
% Base Cases: empty list or list with one item
mergesort([],[]).
mergesort([A],[A]).

% If the list has more than one item,
% split it into two lists of (nearly) equal size,
% sort the two smaller lists, and merge them.
mergesort([A,B|T],S) :-
   split([A,B|T],L1,L2),
   mergesort(L1,S1),
   mergesort(L2,S2),
   merge(S1,S2,S).
```

2.4 Compare the following two definitions of ‘autonomy’ for agents

The definition in lectures allows the knowledge of the designer to determine how the agent acts (though each action is based on percepts), even though the agent controls the execution of those actions, for example the delivery robot. Russell and Norvig’s definition covers “agents” whose percepts directly cause the agent to “act”, e.g. some plankton move around in water but cannot “control” this movement in that they cannot do otherwise (for example, they can only drift with the

current but not swim against the current). Whether these creatures have “sensors” is another question. The definition in lectures covers such creatures that have self-control (can propel themselves through water) but no perception.

Summary: Autonomy is very hard to define, and is closely connected to both choice and control (and free will in humans). Any other suggestions for a definition? It is not easy to define what “perception” is either.

2.5 Consider the Delivery Robot

The robot can become stuck in a cul-de-sac. Now what strategy can the robot use to become “unstuck”?