NTNU

Department of Computer Science

Examination paper for TDT4137 Cognitive Architectures

Academic contact during examination: Asbjørn Thomassen

Phone: 90145710

Examination date: 15 December 2017

Examination time (from-to): 0900-1300

Permitted examination support material: D/ Calculator

Other information:

Language: English

Number of pages (front page excluded): 6

Number of pages enclosed: 0

Informasjon om trykking av eksamensoppgave

Originalen er:

1-sidig ☐ **2-sidig** ☒

sort/hvit ☐ **farger** ☒

skal ha flervalgskjema ☐

Checked by:

5/12-17 [illegible]

Date Signature

Question 1 (20%)

a) Cognitive systems can be divided into subclasses related to the cognition concept. Name three cognitive architectures and place them in their subclass.
Where do systems based on neural networks belong?

Then show which research field each of the architectures is mainly based on.

b) Which subsystems with associated processors do we have in Model Human Processor (MHP)?

Explain briefly three important basic operations in MHP, all performed by the cognitive processor and taking a cycle.

A Roman symbol for 2 (II) comes up on a screen. Soon after, the number 2 also appears on the screen:

The symbol II comes first on screen (Roman numerals for 2)

Then 2 comes up next to the Roman numeral

How long time (in MHP) is it from the second symbol, 2, appears on the screen and one recognizes it as the same number?

c) Hicks law (P7 in MHP) for n objects with equal probability is described by

$$T = I_c H \quad \text{hvor} \quad H = \log_2(n+1).$$

What is T, I_c and H, and what this law express?

d) What is analogy reasoning?
How does Gentner represent knowledge in order to map the target and base in the analogue reasoning?
Explain briefly "The Systematicity Principle".

Question 2 (20%)

a) Explain the behavior of an artificial neuron and the analogy of a biological neuron with dendrites, soma, synapses and axons.

b) Write in pseudo-code the learning algorithm for a perceptron.
The delta rule is:

$$\Delta w_i(p) = \alpha \, x_i(p) \, e_i(p)$$

The function `step(X)` is defined: `0 if X<0 og 1 otherwise.`

What is the role of α ?

Suppose you have a data set with two categories 0 and 1. Each data point is on the form `(x1, x2, yd)` where y_d is desired output (category) for `x1` and `x2`. There are a total of only six points in the set:

```
(0.2, 0.3, 1) (0.4, 0.9, 0) (0.7, 0.9, 1)
(0.8, 0.9, 1) (0.4, 0.3, 0) (0.7, 0.3, 0)
```

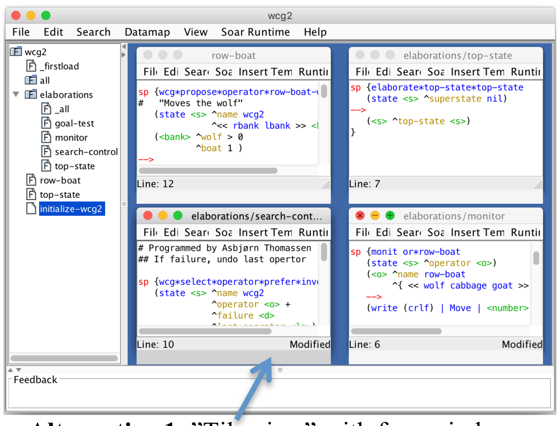
Can you from the given data set see if the perceptron will be able to learn the function (distinguish the two categories `0` and `1`)? Explain the answer.
Tip: Draw the data set in a 2D axis system with `x1` and `x2`

Use the learning algorithm in b) to run an iteration on the first data point in the data set above. Assume we have started the perseptron training with $\theta = 0.2$ and $\alpha = 0.1$ and have the following values on the weights: `w1 = 0.2` and `w2 = 0.1`.

c) What is the idea/reason behind the use of local receptive fields in CNN (Convolutional Neural Networks)? Why do we do sub-sampling (pooling) after a convolutional layer?

Question 3 (20%)

When programming production rules in Soar with VisualSoar, you may choose to view the code-files in tiled or cascaded windows. The figure on the next page shows the two alternatives:



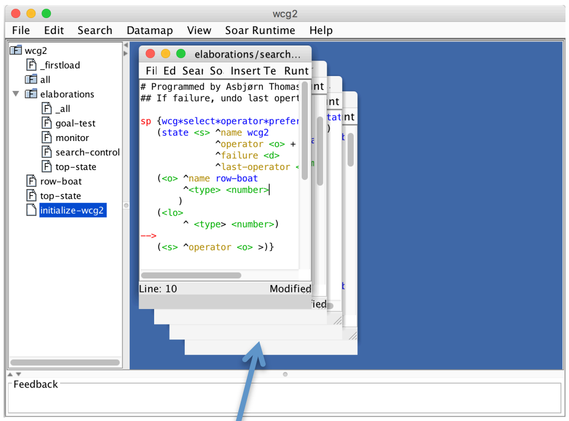
```
sp {wcg*propose*operator*row-boat-
#   "Moves the wolf"
   (state <s> ^name wcg2
              ^<< rbank lbank >> <
   (<bank> ^wolf > 0
           ^boat 1 )
-->
```

```
sp {elaborate*top-state*top-state
   (state <s> ^superstate nil)
-->
   (<s> ^top-state <s>)
}
```

```
# Programmed by Asbjørn Thomassen
## If failure, undo last opertor

sp {wcg*select*operator*prefer*inv
   (state <s> ^name wcg2
              ^operator <o> +
              ^failure <d>
```

```
sp {monit or*row-boat
   (state <s> ^operator <o>)
   (<o> ^name row-boat
        ^{ << wolf cabbage goat >>
   -->
   (write (crlf) | Move | <number>
```



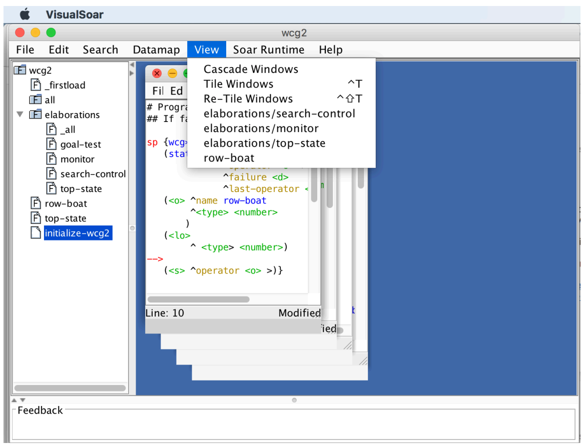
```
# Programmed by Asbjørn Thomas
## If failure, undo last opert

sp {wcg*select*operator*prefer
   (state <s> ^name wcg2
              ^operator <o> +
              ^failure <d>
              ^last-operator <
   (<o> ^name row-boat
        ^<type> <number>
       )
   (<lo>
        ^ <type> <number>)
-->
   (<s> ^operator <o> >)}
```

Alternative 1: ”Tile view” with four windows with Soar-code

Alternative 2: ”Cascade view” of same windows

To view tiled windows as in **Alternative 1**, you may go to the `Tile Windows` command found in the `View`-menu in the menu bar, or just issue a short command `cmd t` by pressing the **cmd**-key followed by the **t**-key.
If you want the cascade view, **Alternative 2**, it is, for an unknown reason, only possible to use menu command `Cascade Windows` in the `View`-menu. The content of the `View`-menu is shown below:



a) Model the goal `selectSoarFileView` in NGOMSL covering the two window alternatives. Generalise the menu method. Draw the goal hierarchy.
What does a deep goal hierarchy indicate?

b) Compute the execution time and show the execution trace with operator times (you may write on the code in a)). What do the different parts of the execution time formula express?
(CP = 1.2, B=0.1, H=0.4, K=0.2, M=1.2, P = 1.1 - all in seconds)

c) Which type of tasks cannot be modelled by GOMS?

Why is it possible to look at some of the operators as parallel operators in your program in a). Give an example.
How is time computed when two or more steps are merged into one.

Question 4 (20%)

EMU is an emotional personal agent that helps you in playing music from your music library. EMU´s happiness is shown through its face and is based on the elapsed time since last play, `sinceTime`, and the trend of how long you play music, `playtime` (average play time). Below you can see two of the many faces of EMU:

sad

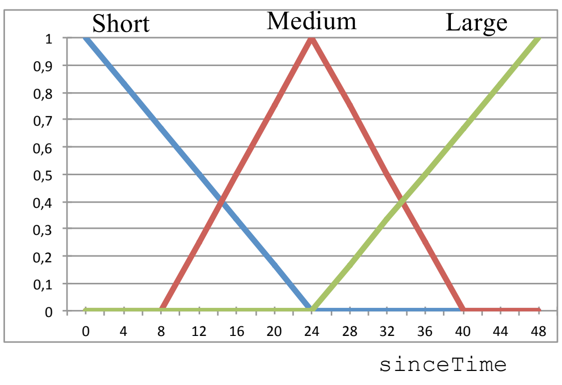
------other faces in-between ----

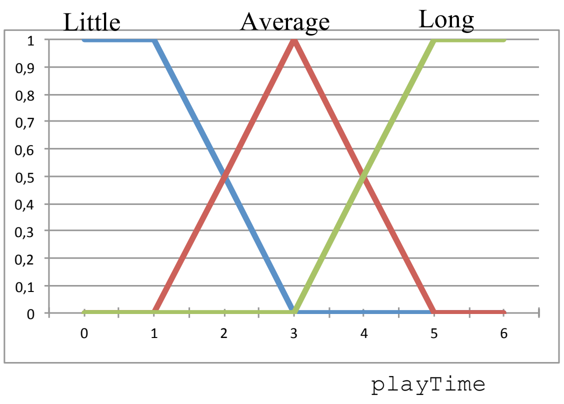
happy

Each time you ask EMU to play, it becomes happier since `sinceTime` (in hours) is reduced and `playTime` (in minutes) is possibly increased. But, EMU becomes unhappy when `sinceTime` and/or `playTime` decrease.

EMU´s face expression is related to *`emotion`* which is a number in the range 0-100. This range represents/covers three moods: Sad (lower part of scale), OK (around middle), and Happy (upper part).

The value of the output-variable *`emotion`* comes from fuzzy-reasoning with two input-variables: `sinceTime` and `playTime`. The variable `sinceTime` has the sets: **Short**, **Medium** and **Large**, and `playTime` has: **Little**, **Average** and **Long**.





a) Given *`sinceTime = 14, playTime = 3.5`* and the following Sugeno-rules with three singeltons:

```
If (playTime is Little) then emotion is 20

If (playTime is Average OR sinceTime is Medium) then emotion is 50

If (playTime is Long AND sinceTime is NOT Short) then emotion is 80
```

How happy is EMU (given by *`emotion`*)? Assume Sugeno-reasoning and show the steps in finding the answer. You may draw and write directly on the figures/rules given above.

What is the difference between Mamadani and Sugeno?

What is the reason for using `hedges` in fuzzy reasoning?
Give an example.

b) Rewrite the rules in a) to Mamdani and draw your choice of output-sets.

What kind of emotion will the EMU-agent show when you are using Mamdani reasoning. You may here use approximate values from your figures/drawings in the computation.

Question 5 (20%)

a) Many AI-systems are making internal models of its environment that can be used in the reasoning. How is that in "Brook´s subsumption architecture?

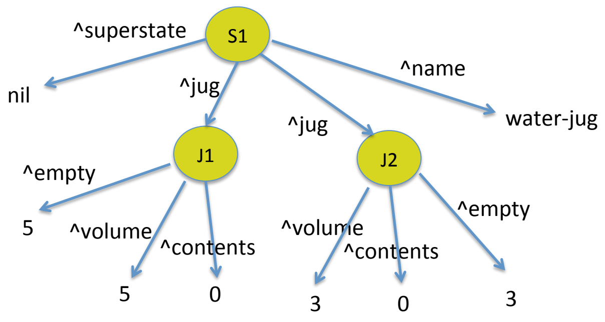
Explain briefly the main idea behind this architecture.

b) Icarus can be looked upon as a layered architecture, with distinct modules that are integrated in a cascade, where low-level modules deliver to high-level modules. Which modules does Icarus have? What is the purpose of the lowest module?

Which situations generate an `impasse` in Icarus and how does the architecture act to try to solve it?

c) How is Soar related to "The Heuristic Search Hypothesis"?

In the well-known "Water-jug"-problem we have two jars with capacity of 3 and 5 liters of water. The problem to be solved is to get 4 liters in one of the jars by doing one or more: fill to top, pour from one to the other, or empty jar. Assume initially that problem has the following representation:



The attribute `^volume` is the jar water capacity in liter, `^contents` is the current number of liters in the jar, and `^empty` is the space left to be full.

Write a Soar rule that checks if the problem is solved, i.e. one of the jar contains 4 litres, and that stops the agent by executing a `(halt)`.
What type of rule is this?

Write code to fill up a jar by first checking if there is space for more water.
How does the Soar-architecture influence your programming (in terms of number of rules and rule types)?

A little help in coding: Below is shown a sketch for how to change value for an attribute `^attr`. First we make a new similar attribute with the new value, here 4, and then we delete the old:

```
sp {exampleRuleName
    (state <s> …..  )
   …..
    (<j> ^attr <attr>)
-->  …….
    (<j> ^attr 4           // make new attribute with value
         ^attr <attr> -)} // delete old attribute with value
```

You can test if an attribute is greater than a value, here 22, with the construction:

```
(<i> ^attr > 22)
```

d) By simple analysis of EEG-data, it is possible to recognize eight types of positive and negative emotions: Joyful, Angry, Protected, Sad, Surprised, Fear, Satisfied and Unconcerned. This can be done by looking at the four EEG-frequency bands: Alpha (α), Beta (β), Theta (θ) and Delta (δ).

How can you extract these bands from the EEG-signal and use them to generate digital codes for classifying emotion ?

The digital codes may generate **same** code for more than one emotion (occur for Surprised and Fear).
What can be done to solve this ambiguity?