Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2019-01-10
Sal (3)	U1(35) U3(18) U4(20)
Tid	14-18
Utb. kod	TDDC17
Modul	TEN1
Utb. kodnamn/benämning **Modulnamn/benämning**	Artificiell intelligens En skriftlig tentamen
Institution	IDA
Antal uppgifter som ingår i tentamen	8
Jour/Kursansvarig Ange vem som besöker salen	Mariusz Wzorek
Telefon under skrivtiden	070-3887122
Besöker salen ca klockan	ja
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Anna Grabska Eklund, anna.grabska.eklund@liu.se, ankn. 2362
Tillåtna hjälpmedel	Miniräknare/Hand calculators
Övrigt	
Antal exemplar i påsen	

Linköpings Universitet
Institutionen för Datavetenskap
Patrick Doherty

Tentamen
TDDC17 Artificial Intelligence
10 January 2019 kl. 14-18

Points:

The exam consists of exercises worth 39 points.
To pass the exam you need 20 points.

Auxiliary help items:

Hand calculators.

Directions:

You can answer the questions in English or Swedish.
Use notations and methods that have been discussed in the course.
In particular, use the definitions, notations and methods in appendices 1-3.
Make reasonable assumptions when an exercise has been under-specified.
State these assumptions explicitly in your answer.
Begin each exercise on a new page.
Write only on one side of the paper.
Write clearly and concisely.

Jourhavande: Mariusz Wzorek, 070 388 71 22. Mariusz will arrive for questions around 15.00.

1. The following questions pertain to the course article by Newell and Simon entitled *Computer Science as an Empirical Enquiry: Symbols and Search.*

   (a) What is a *physical symbol system* (PSS) and what are its structural and conceptual components? **[2p]**

   (b) What is the Physical Symbol System Hypothesis? **[1p]**

   (c) Do you think the Physical Symbol System Hypothesis provides an adequate description of the structural and conceptual components required for a system exhibiting intelligence? Provide reasonable justifications for your opinion. **[1p]**

   (d) What is the heuristic search hypothesis? **[1p]**

2. The following questions pertain to machine learning. Give *short and informative* answers.

   (a) Explain and contrast the purpose of γ and α in the Q-learning update formula, **[2p]**

   $$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(R(s_t) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right).$$

   (b) Explain the concept of overfitting in supervised learning. **[1p]**

   (c) What properties of *convolutional* neural networks often make them better suited for image classification than fully-connected (regular) deep neural networks. **[2p]**

3. The following questions pertain to automated planning.

   (a) Recall that a typical partial order planner begins with an *initial plan* having a specific form, and proceeds by resolving so called *flaws* in this plan. For example, there can be *threats* that need to be resolved.

   – What is a *threat*? Remember that there is a specific definition of threats for partial order planning, related to the structure of a plan that is under construction but has not yet been finished.

   – How can we *resolve* a threat? That is, how does a POCL planner modify or extend the plan so that the threat no longer exists? **[2p]** (total for both questions)

   (b) The planning lectures (and lecture notes) gave a formal definition of relaxation which is not based on how you change a planning problem but on what happens with the set of *solutions*. This definition also allowed us to prove more directly that optimal solutions to a relaxed problem can be used to define admissible heuristics.

   – Provide a definition of relaxation by completing the following sentence: Given two classical planning problems P and P', we know that P' is a relaxation of P if and only if ...

   – Also provide at least one general example of how the state space (the state/action graph) of a planning problem instance could be modified by a relaxation, in a way that satisfies the criterion above. **[3p]**(total for both questions)

4. A* search is the most widely-known form of best-first search. The following questions pertain to A* search:

   (a) Explain what an *admissible* heuristic function is using the notation and descriptions in (c). **[1p]**

   (b) What is a *consistent* heuristic function and can such a function be inadmissible? Explain why or why not. **[1p]**

   (c) Let $h(n)$ be the estimated cost of the cheapest path from a node n to the goal. Let $g(n)$ be the path cost from the start node n_0 to n. Let $f(n) = g(n) + h(n)$ be the estimated cost of the cheapest solution through n.

   Provide a sufficiently rigid proof that A* is optimal if $h(n)$ is admissible. You need only provide a proof for either tree-search (seminar slides) or graph-search (in course book). If possible, use a diagram to structure the contents of the proof to make it more readable. **[2p]**

5. The Davis-Putnam Algorithm and its extension DPLL, provide the basis for general propositional model-checking. DPLL offers three improvements over the standard generate-and-test algorithm, TT-Entails, described in the course book. The following questions pertain to DPLL and the three improvements.

   (a) The DPLL algorithm checks whether a formula in propositional logic is satisfiable. Suppose one has a conjunction of propositional formulas Δ and a propositional formula α.

      i. How would DPLL be used to determine whether or not $\Delta \models \alpha$? (Please be precise in the steps used to do this.) **[2p]**

   (b) Name and succinctly describe the three improvements over TT-entails (mentioned in the textbook) that are used in the DPLL algorithm. In doing this, provide a concrete example for each. **[4p]**

6. The following questions pertain to Constraint Satisfaction Problems (CSP's). CSP's consist of a set of variables, a value domain for each variable and a set of constraints. A solution to a CS problem is a consistent set of bindings to the variables that satisfy the constraints.

   Figure 1 shows a constraint graph with eight variables. The value domains for each variable are the numbers 1 to 8. The constraints state that adjacent/connected nodes can not have consecutive numbers and they must be different. For example, if node C is labeled 1, then nodes A,D, and G can not be labeled 2.

   (a) Explain what the *Degree Heuristic* is and why it is used. If the degree heuristic is applied to the constraint graph in figure 1, what are the candidate nodes that could be chosen for labeling? **[1p]**

   (b) Explain what the *Least Constraining Value Heuristic* is and why it is used. If the least constraining value heuristic is applied to one of the candidate nodes chosen in the previous question, what would the potential candidate values be? Explain your choice. **[1p]**

   (c) Suppose node $D = \{8\}$, node $E = \{1\}$ and nodes A, B, C, F, G, H are labeled $\{1, 2, 3, 4, 5, 6, 7, 8\}$. Make the constraint graph arc consistent. In your answer, simply provide labels for nodes A, B, C, F, G, H. You can use the AC-3 algorithm in Appendix 1 to assist you, but showing exhaustive steps is not necessary. **[2p]**

   (d) Provide a solution to the CSP in the example. **[1p]**

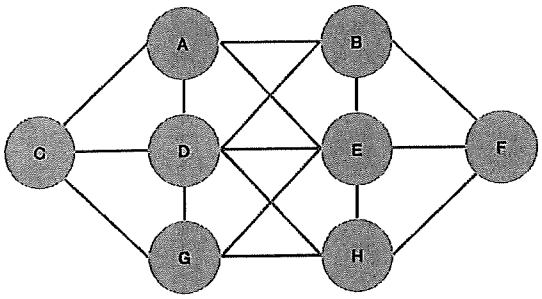

Figure 1: Constraint Graph

7. The following questions pertain to Answer Set Programming. Appendix 3 may be useful to use:

   (a) Given the program Π_1, consisting of the following rules:

   r1: puma $\leftarrow$ *not* cat.

   r2: cat $\leftarrow$ *not* puma.

   r3: tiger $\leftarrow$ *not* cheetah.

   i. What are the *possible* answer sets for Π_1? **[1p]**

   ii. For each possible answer set S, provide the reduct, Π_1^S for Π_1. **[1p]**

   iii. Generate $Cn(\Pi_1^S)$ for each reduct Π_1^S of Π_1. **[1p]**

   iv. What are the *actual* answer sets for Π_1 (Explain why)? **[1p]**

   (b) Why is Answer Set Programming considered to be a nonmonotonic reasoning formalism (Be precise and use an example)? **[1p]**

8. Use the Bayesian network in Figure 2 below together with the associated conditional probability tables to answer the following questions. Appendix 2 may be helpful to use. If you do not have a hand-held calculator with you, make sure you set up the solution to the problems appropriately for partial credit.

   The conditional probability tables (CPT) should be interpreted as follows: For each row in a CPT, the left-hand side enumerates the values of the evidence variables, and each column in the right-hand side provides the probability of the query variable for each possible value of the query variable. For instance, for the CPT associated with the random variable SP, the first row denotes that $P(SP = \text{High} \mid SM = \text{Good}, OI = \text{Good},) = 0.80$ and $P(SP = \text{Low},\mid SM = \text{Good},, OI = \text{Good},) = 0.20$.

   (a) Given the Bayesian network below, write down the full joint distribution it represents. In other words, define $P(SP, SM, OI, IR)$ as a product of the conditional relationships that can be derived from the network below. **[1p]**

   (b) What is $P(SP = \text{High}, SM = \text{Bad}, OI = \text{Bad}, IR = \text{High})$? **[1p]**

   (c) What is $P(SP = \text{Low} \mid SM = \text{Good}, IR = \text{High})$? **[2p]**

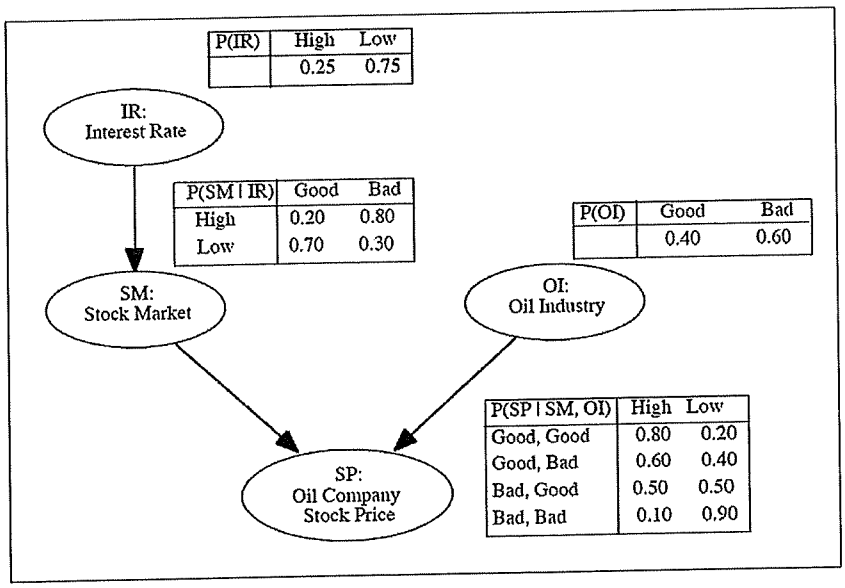

Figure 2: Bayesian Network Example

Appendix 1: Arc-Consistency algorithm

```
function AC-3( csp) returns false if an inconsistency is found and true otherwise
  inputs: csp, a binary CSP with components (X, D, C)
  local variables: queue, a queue of arcs, initially all the arcs in csp

  while queue is not empty do
    (X_i, X_j) ← REMOVE-FIRST(queue)
    if REVISE(csp, X_i, X_j) then
      if size of D_i = 0 then return false
      for each X_k in X_i.NEIGHBORS - {X_j} do
        add (X_k, X_i) to queue
  return true
```

```
function REVISE( csp, X_i, X_j) returns true iff we revise the domain of X_i
  revised ← false
  for each x in D_i do
    if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j then
      delete x from D_i
      revised ← true
  return revised
```

Figure 3: AC3 Arc Consistency Algorithm

Appendix 2

A generic entry in a joint probability distribution is the probability of a conjunction of particular assignments to each variable, such as $P(X_1 = x_1 \wedge \ldots \wedge X_n = x_n)$. The notation $P(x_1, \ldots, x_n)$ can be used as an abbreviation for this.

The chain rule states that any entry in the full joint distribution can be represented as a product of conditional probabilities:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid x_{i-1}, \ldots, x_1) \tag{1}$$

Given the independence assumptions implicit in a Bayesian network a more efficient representation of entries in the full joint distribution may be defined as

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid \mathit{parents}(X_i)), \tag{2}$$

where $\mathit{parents}(X_i)$ denotes the specific values of the variables in $\mathit{Parents}(X_i)$.

Recall the following definition of a conditional probability:

$$\mathbf{P}(X \mid Y) = \frac{\mathbf{P}(X \wedge Y)}{\mathbf{P}(Y)} \tag{3}$$

The following is a useful general inference procedure:

Let X be the query variable, let $\mathbf{E}$ be the set of evidence variables, let $\mathbf{e}$ be the observed values for them, let $\mathbf{Y}$ be the remaining unobserved variables and let α be the normalization constant:

$$\mathbf{P}(X \mid \mathbf{e}) = \alpha \mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y}) \tag{4}$$

where the summation is over all possible $\mathbf{y}$'s (i.e. all possible combinations of values of the unobserved variables $\mathbf{Y}$).

Equivalently, without the normalization constant:

$$\mathbf{P}(X \mid \mathbf{e}) = \frac{\mathbf{P}(X, \mathbf{e})}{\mathbf{P}(\mathbf{e})} = \frac{\sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y})}{\sum_{\mathbf{x}} \sum_{\mathbf{y}} \mathbf{P}(\mathbf{x}, \mathbf{e}, \mathbf{y})} \tag{5}$$

Appendix 3: Answer Set Programming

Computing Answer Sets for a program Π:
Given a program Π:

1. Compute the possible answer sets for Π:
   (a) Powerset 2^{Π} of all atoms in the heads of rules in Π.
2. For each $S \in 2^{\Pi}$:
   (a) Compute the reduct Π^S of Π.
   (b) If $Cn(\Pi^S) = S$ then S is an answer set for Π.
   (c) If $Cn(\Pi^S) \neq S$ then S is not an answer set for Π.

The following definitions may be useful:

Definition 1 A program Π consists of a signature Σ
and a collection of rules of the form:

$$l_0 \vee, \ldots, \vee l_i \leftarrow l_{i+1}, \ldots, l_m, not\ l_{m+1}, \ldots, not\ l_n$$

where the l's are literals in Σ. □

Definition 2 [Satisfiability]
A set of (ground) literals satisfies:

1. l **if** $l \in S$;
2. *not* l **if** $l \notin S$;
3. $l_1 \vee \ldots l_n$ **if** *for some* $1 \leq i \leq n, l_i \in S$;
4. a set of (ground) extended literals **if** S
   satisfies every element of this set;
5. rule r **if** , whenever S satisfies r's body, it satisfies r's head. □

Definition 3 [Answer Sets, Part I]
Let Π be a program not containing default negation
(i.e., consisting of rules of the form):

$$l_0 \vee, \ldots, \vee l_i \leftarrow l_{i+1}, \ldots, l_m.$$

An *answer set* of Π is a consistent set S of (ground) literals
such that

1. S satisfies the rules of Π and
2. S is minimal (i.e., there is no proper subset of S that
   satisfies the rules of Π. □

Appendix 3 is continued on the next page.

Definition 4 [Answer Sets, Part II]
Let Π be an arbitrary program and S be a set of ground literals. By Π^S we denote the program obtained from Π by

1. removing all rules containing *not* l such that $l \in S$;
2. removing all other premises of the remaining rules containing *not*.

S is an answer set of Π if S is an answer set of Π^S.
We refer to Π^S as the *reduct* of Π with respect to S. □

Definition 5 [Consequence operator T_Π]
The smallest model, $Cn(\Pi)$, of a positive program Π can be computed via its associated *consequence operator* T_Π. For a set of atoms X we define,

$$T_\Pi X = \{head(r) \mid r \in \Pi \text{ and } body(r) \subseteq X\}.$$

Iterated applications of T_Π are written as T_Π^j for $j \geq 0$, where

$$T_\Pi^0 X = X$$
$$T_\Pi^i X = T_\Pi T_\Pi^{i-1} X \text{ for } i \geq 1.$$

For any positive program Π, we have $Cn(\Pi) = \bigcup_{i \geq 0} T_\Pi^i \emptyset$.
Since T_Π is monotonic, $Cn(\Pi)$ is the smallest fixpoint of T_Π. □