Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2018-08-27
Sal (2)	**TER2(30)** TERE(1)
Tid	8-12
Kurskod	TDDC17
Provkod	TEN1
Kursnamn/benämning **Provnamn/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	ca kl. 10
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
27 August 2018 kl. 08-12

Points:

The exam consists of exercises worth 41 points.
To pass the exam you need 23 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 1-4.
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, 0703887122. Mariusz will arrive for questions around 10.00.

1. Based on the material in Chapter 2 of the course book, define succinctly in your own words, the following terms:

   (a) Agent, Agent Function, Agent Program [1p]

   (b) Performance Measure, Rationality, Autonomy [1p]

   (c) Reflex Agent. Also provide a schematic diagram of such an agent. [1p]

   (d) Model-based Agent. Also provide a schematic diagram of such an agent. [1p]

   (e) Goal-based Agent. Also provide a schematic diagram of such an agent. [1p]

2. 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]

3. Consider the following logical theory about registered voters (where hilary, bernie and shawn are constants) and we view grounded atomic formulas as propositional atoms. (In this case unification of two grounded atomic formulas is successful when they are identical.):

$$Democrat(\text{hilary}) \tag{1}$$

$$Likes(\text{hilary, bernie}) \tag{2}$$

$$Likes(\text{bernie, shawn}) \tag{3}$$

$$(Likes(\text{hilary, bernie}) \wedge Likes(\text{bernie, shawn})) \rightarrow Likes(\text{hilary, shawn}) \tag{4}$$

$$\neg(\neg Republican(\text{shawn}) \wedge \neg Democrat(\text{shawn})) \tag{5}$$

$$\neg Democrat(\text{shawn}) \tag{6}$$

We would like to show using resolution that a Democrat likes a Republican. To do this, answer the following questions:

   (a) Convert formulas (1) - (6) into conjunctive normal form (CNF) with the help of appendix 1. [1p]

   (b) Prove that $(Democrat(\text{hilary}) \wedge Republican(\text{shawn}) \wedge Likes(\text{hilary, shawn}))$ is a logical consequence of (1) - (6) using the resolution proof procedure. [3p]

      • Your answer should be structured using one or more resolution refutation trees/graphs (as used in the book or course slides).

4. 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 (where a1 is a constant):

      r1: $apple$(a1).
      r2: $tasty$(a1) $\leftarrow apple$(a1), $not\ ab$(a1).
      r3: $rotten$(a1).
      r4: ab(a1) $\leftarrow rotten$(a1).

      1 what is the reduct Π_1^S for Π_1 given that $S = \{apple(a1), ab(a1)\}$? [1p]
      2 what is the reduct Π_1^S for Π_1 given that $S = \{apple(a1), rotten(a1)\}$? [1p]

   (b) Given the program Π_2 consisting of the following two rules (where short, tall are constants):

      r1: $height$(tall) $\leftarrow not\ height$(short)
      r2: $height$(short) $\leftarrow not\ height$(tall).

      What are the *possible* answer sets for Π_2? [1p]

   (c) Given the *possible* answer sets for Π_2 above, which of those possible answer sets are in fact answer sets for program Π_2? [2p]
      When answering this question be sure to show why, by using the reducts, $\Pi_2^{S_i}$, where S_i is instantiated to each possible answer set, and the consequence operator T_Π described in appendix 3.

   (d) Why is Answer Set Programming considered to be a nonmonotonic reasoning formalism? [1p]

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

   (a) Two major classes of learning algorithms are supervised learning and reinforcement learning. Contrast these in terms of what inputs are needed for *training*. [2p]

   (b) The Q-learning algorithm is often paired with an exploration strategy. Explain why. [2p]

   (c) This question pertains to deep learning. Assume you want to train a classifier from image inputs. Explain why a convolutional neural network would, or would not, be suitable for such a task? [2p]

6. Recall that partial-order planning generates plans where actions are not necessarily constrained to be ordered in a sequence but can (at least during search) be placed "in parallel". This requires a completely different search space than the one used for forward-chaining state space search. The standard *partial-order causal-link* search space gives rise to the concept of a *flaw*.

   (a) Name two distinct types of flaw that can arise when a partial-order causal-link planner searches for a plan. [1p]

   (b) Explain each type of flaw clearly. The explanations should be *sufficiently clear* that someone who has not seen the definitions in the book or lecture notes can take a partial plan, analyze it, and identify the flaws it contains (if any). Feel free include examples of partial plans that illustrate each flaw type, but be aware that examples in themselves are not sufficient to define the exact boundaries between flaws and non-flaws. [2p]

   (c) For each type of flaw, describe at least one way that one may be able to resolve it. [2p]

3

7. The following questions pertain to Bayesian Networks. Figure 1 below provides a Bayesian network for a satellite monitoring problem. Conditional tables for the problem are also provided.

(a) Provide an equivalent equation for the joint distribution $P(B, S, E, D, C)$, as a product of conditional distributions based on the independence assumptions associated with the Bayesian network in Figure 1. [2p]

(b) Suppose the solar panel and electric system do not fail, the battery does and there is trajectory deviation, but no communication loss: $P(b, \neg s, \neg e, d, \neg c)$. What is the probability that this can happen? [1p]

(c) Suppose the solar panel fails, but we observe no communication loss. What is the probability of a trajectory deviation: $P(d \mid s, \neg c)$? [2p]
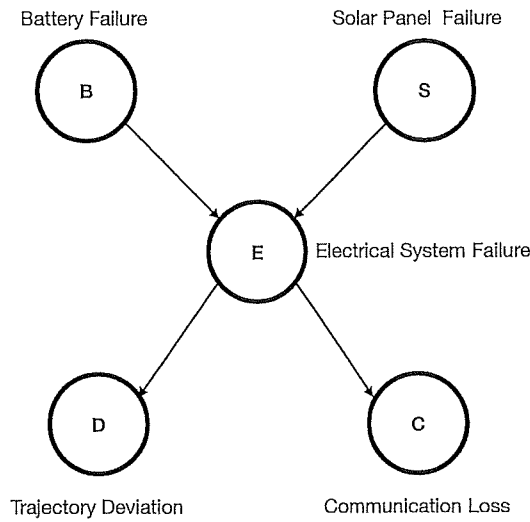
Battery Failure   Solar Panel Failure

Electrical System Failure

Trajectory Deviation   Communication Loss

Figure 1: Bayesian Network Example

B	S	$P(E)$
T	T	0.99
T	F	0.87
F	T	0.85
F	F	0.10

S	P(S)
T	0.02

B	P(B)
T	0.05

E	$P(D)$
T	0.99
F	0.10

E	$P(C)$
T	0.99
F	0.15

8. Constraint satisfaction problems consist of a set of variables, a value domain for each variable and a set of constraints. Figure 2 depicts a constraint graph with variables A, B and C, each with a value domain $\{1, 2, 3, 4\}$. Binary constraints are associated with each arc. The graph it not arc-consistent. Appendix 4 provides the AC3 algorithm which makes a constraint graph arc-consistent.

A solution to a CS problem is a consistent set of bindings to the variables that satisfy the constraints. A standard backtracking search algorithm can be used to find solutions to CS problems. Constraint propagation is the general term for propagating constraints on one variable onto other variables. Constraint propagation may be integrated with a backtracking algorithm or used for preprocessing.

Describe the following:

(a) What is the Forward Checking technique? [1p]

(b) What is arc consistency? Provide a definition. [1p]

(c) Provide a constraint graph that is arc consistent but globally inconsistent. [1p]

(d) Make the constraint graph in figure 2 arc consistent using the AC3 algorithm in Appendix 4. For the answer, only the resulting consistent bindings for each variable in the constraint graph output by the AC3 algorithm are required although you may show more of the process. As stated above, the value domain for each of the variables, A, B and C is $\{1, 2, 3, 4\}$. [2p]
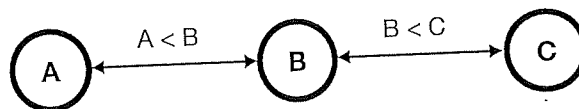

Figure 2: Constraint graph

Appendix 1

Converting arbitrary wffs to CNF form:
(Propositional/grounded 1st-order formula case)

1. Eliminate implication signs using the equivalence: $\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta$.

2. Reduce scopes of negation signs using De Morgan's Laws:

   - $\neg(\omega_1 \vee \omega_2) \equiv \neg\omega_1 \wedge \neg\omega_2$
   - $\neg(\omega_1 \wedge \omega_2) \equiv \neg\omega_1 \vee \neg\omega_2$

3. Remove double negations using the equivalence: $\neg\neg\alpha \equiv \alpha$.

4. Put the remaining formula into conjunctive normal form. Two useful rules are:

   - $\omega_1 \vee (\omega_2 \wedge \omega_3) \equiv (\omega_1 \vee \omega_2) \wedge (\omega_1 \vee \omega_3)$
   - $\omega_1 \wedge (\omega_2 \vee \omega_3) \equiv (\omega_1 \wedge \omega_2) \vee (\omega_1 \wedge \omega_3)$

5. Eliminate $\wedge$ symbols so only clauses remain.

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:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid x_{i-1}, \ldots, x_1) \tag{7}$$

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

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid parents(X_i)), \tag{8}$$

where $parents(X_i)$ denotes the specific values of the variables in $Parents(X_i)$.

Recall the following definition of a conditional probability:

$$\mathrm{P}(X \mid Y) = \frac{\mathrm{P}(X \wedge Y)}{\mathrm{P}(Y)} \tag{9}$$

The following is a useful general inference procedure:

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

$$\mathrm{P}(X \mid \mathrm{e}) = \alpha \mathrm{P}(X, \mathrm{e}) = \alpha \sum_{\mathbf{y}} \mathrm{P}(X, \mathrm{e}, \mathbf{y}) \tag{10}$$

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

Equivalently, without the normalization constant:

$$\mathrm{P}(X \mid \mathrm{e}) = \frac{\mathrm{P}(X, \mathrm{e})}{\mathrm{P}(\mathrm{e})} = \frac{\sum_{\mathbf{y}} \mathrm{P}(X, \mathrm{e}, \mathbf{y})}{\sum_{\mathbf{x}} \sum_{\mathbf{y}} \mathrm{P}(\mathbf{x}, \mathrm{e}, \mathbf{y})} \tag{11}$$

Appendix 3: Answer Set Programming

<u>Computing Answer Sets for a program Π:</u>
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. □

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,
$$T_\Pi X = \{head(r) \mid r \in \Pi \ and \ body(r) \subseteq X\}.$$
Iterated applications of T_Π are written as T_Π^j for $j \geq 0$, where

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

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. □

Appendix 4: 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**
        $(X_i,\ X_j) \leftarrow$ 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 $\leftarrow$ *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 $\leftarrow$ *true*
    return *revised*

Figure 3: AC3 Arc Consistency Algorithm