Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2016-10-25
Sal (3)	TER1 TER2 TERE
Tid	14-18
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	Olov Andersson
Telefon under skrivtiden	070 574 33 43
Besöker salen ca klockan	ca kl. 16
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	

Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2016-10-25
Sal (3)	TER1 TER2 TERE
Tid	14-18
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	Olov Andersson
Telefon under skrivtiden	070 574 33 43
Besöker salen ca klockan	ca kl. 16
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	

Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2016-10-25
Sal (3)	TER1 TER2 TERE
Tid	14-18
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	Olov Andersson
Telefon under skrivtiden	070 574 33 43
Besöker salen ca klockan	ca kl. 16
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
25 October 2016 kl. 14-18

Points:

The exam consists of exercises worth 38 points.
To pass the exam you need 19 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 1-5.
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: Olov Andersson, 070 574 3343. Olov will arrive for questions around 16.00.

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. Consider the following logical theory about registered voters (where fred, ted and samantha 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.):

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

$$Likes(\text{fred}, \text{ted}) \tag{2}$$

$$Likes(\text{ted}, \text{samantha}) \tag{3}$$

$$(Likes(\text{fred}, \text{ted}) \wedge Likes(\text{ted}, \text{samantha})) \rightarrow Likes(\text{fred}, \text{samantha}) \tag{4}$$

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

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

   We would like to show using resolution that a registered Democrat likes a registered 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 $(Democrat(\text{fred}) \wedge Republican(\text{samantha}) \wedge Likes(\text{fred}, \text{samantha}))$ 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 (as used in the book or course slides).

3. Constraint satisfaction (CS) problems 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.

   (a) Suppose there are 5 territories T1, T2, T3, T4, and T5, each with a sensor that monitors the area associated with that territory. Each sensor has three possible radio frequencies, F1,F2, F3. Sensors overlap if they are in adjacent areas. The adjacency relation between two territories is symmetric. Let $Adj(x, y)$ represent the adjacency relation where $Adj(T1, T2)$, $Adj(T1, T3)$,$Adj(T2, T3)$, $Adj(T3, T4)$. If two sensors overlap, they can not use the same frequency.

      1. Define a constraint satisfaction problem for this scenario. **[1p]**
      2. Provide a constraint graph for the CS problem. **[1p]**
      3. Provide one solution for the CS problem. **[1p]**

   (b) Provide an example of a constraint problem with a constraint graph that is arc-consistent but has no globally consistent solutions. **[1p]**

   (c) Figure 1 depicts a constraint graph with variables A, B, C, D, E, each with a value domain $\{1, 2, 3, 4\}$. Binary constraints are associated with each arc and domain constraints are associated with each node. The graph is domain consistent but not arc-consistent.

   Make the constraint graph in figure 1 arc consistent. You can use the AC3 algorithm in Appendix 5 to help you. For the answer, only the resulting consistent bindings for each variable in the constraint graph are required. **[2p]**

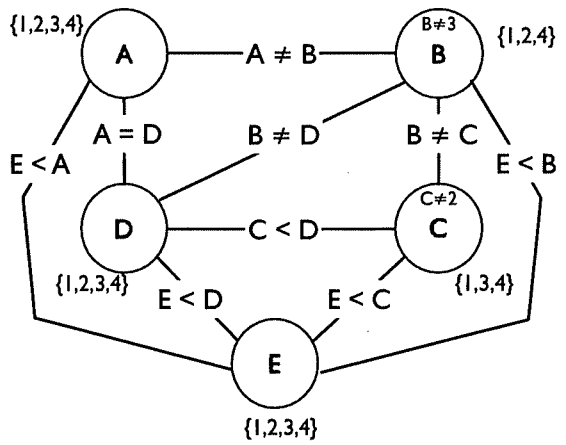


Figure 1: Constraint graph that is domain consistent but not arc consistent

4. The following questions pertain to automated planning.

   (a) What is satisficing planning? Do satisficing planners typically require admissible heuristics? Why or why not? **[2p]**

   (b) Relaxation is an important method for finding admissible heuristic functions. Describe two ways in which relaxation can change the state space of a problem in order to preserve all solutions but also introduce new solutions. **[2p]**

5. 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) Suppose a robot is searching for a path from one location to another in a rectangular grid of locations in which there are arcs between adjacent pairs of locations and the arcs only go in north-south (south-north) and east-west (west-east) directions. Furthermore, assume that the robot can only travel on these arcs and that some of these arcs have obstructions which prevent passage across such arcs.

   Provide an admissible heuristic for this problem. Explain why it is an admissible heuristic and justify your answer explicitly. **[2p]**

   (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]**

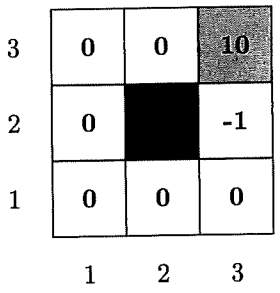
6. The following questions pertain to machine learning. Give detailed answers. Appendix 4 may be helpful to use.

   (a) Assume a supervised learning problem from examples (x, y), where the outputs y belong to the real numbers. Give a suitable loss function for training a simple linear model. **[1p]**

   (b) Assume you are training a classic fully-connected feed-forward neural network with p parameters. Using the *backpropagation algorithm* to compute loss gradients, what is the computational complexity per example? **[1p]**

   (c) Assume a Q-learning agent in the environment below. Actions are {*Up, Down, Left, Right*} and the transition function is *deterministic* (no uncertainty). The numbers in the squares represent the reward the agent is given in each state and the black square is impassable. The gray square (3,3) is a *terminal* state where the episode ends, and the Q-table for its actions can therefore be assumed fixed to its reward.

   The agent always starts each episode in the *lower right* corner (3,1), follows the simple policy of always going up to the terminal state, and then starts the next episode. Using the Q-learning formula (see appendix) with a discount factor of 0.9 and learning rate of 1, compute the utility of going *Up* from the starting state. Show the steps of your calculation.**[2p]**



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 (where ted is a constant):

   r1: *tiger*(ted).

   r2: *killer*(ted) $\leftarrow$ *tiger*(ted), *not ab*(ted).

   r3: *tame*(ted).

   r4: *ab*(ted) $\leftarrow$ *tame*(ted).

   1 what is the reduct Π_1^S for Π_1 given that $S = \{tiger(\text{ted}), ab(\text{ted})\}$? **[1p]**

   2 what is the reduct Π_1^S for Π_1 given that $S = \{tiger(\text{ted}), tame(\text{ted})\}$? **[1p]**

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

   r1: *registered*(republican) $\leftarrow$ *not registered*(democrat)

   r2: *registered*(democrat) $\leftarrow$ *not registered*(republican).

   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]**

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

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

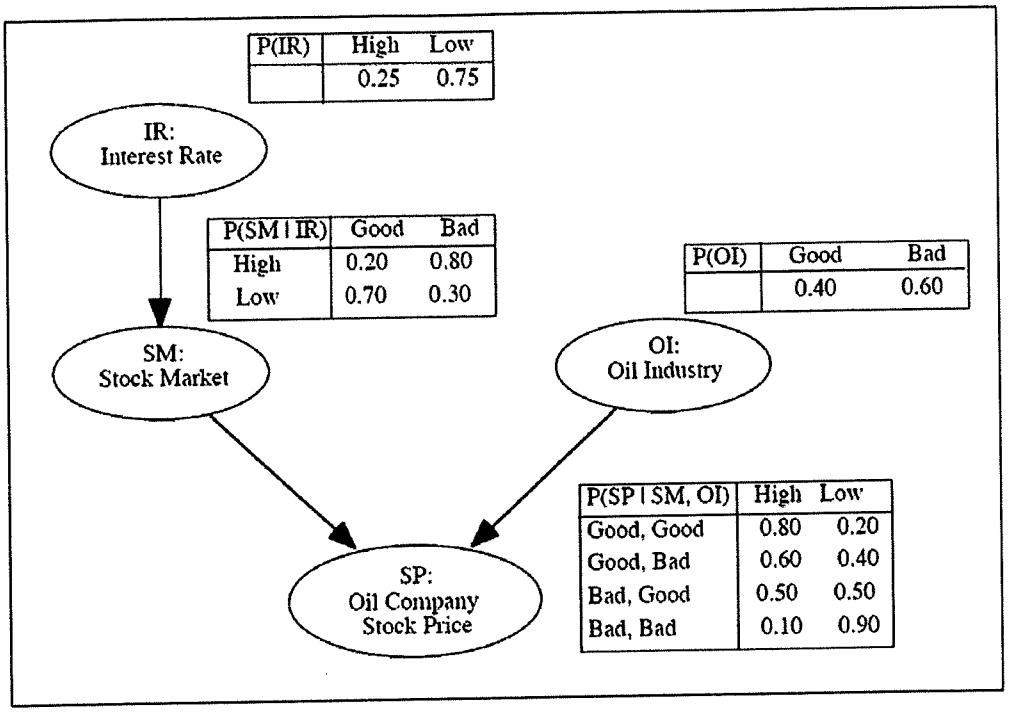


Figure 2: Bayesian Network Example

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

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

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:

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

The following is a useful general inference procedure:

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

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

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:

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

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. □

Appendix 4: Q-learning

Q-learning is a model-free reinforcement learning approach. We use a simplified definition from the course literature:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R(s_t) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{12}$$

where s_t and s_{t+1} are states in a sequence, $R(s_t)$ is the reward for state s_t, $Q(s_t, a_t)$ is the estimated utility of taking action a_t in state s_t, γ is the discount factor and α is a fixed learning rate.

The Q-function is initialized to zero, except for any terminal states where all actions are fixed to the terminal reward. Each time the agent performs an action, it can be updated based on the observed sequence $..., s_t, R(s_t), a_t, s_{t+1}, ...$ The Q-function can then be used to guide agent behavior, for example by extracting a policy.

Appendix 5: 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