Försättsblad till skriftlig tentamen vid Linköpings Universitet

Datum för tentamen	2014-08-25
Sal (1) Om tentan går i flera salar ska du bifoga ett försättsblad till varje sal och ringa in vilken sal som avses	TER3
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	0703887122
Besöker salen ca kl.	ca kl. 10
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Anna Grabska Eklund, ankn. 2362, anna.grabska.eklund@liu.se
Tillåtna hjälpmedel	Hand Calculator/miniräknare
Övrigt	
Vilken typ av papper ska användas, rutigt eller linjerat	Valfritt
Antal exemplar i påsen	

Linköpings Universitet
Institutionen för Datavetenskap
Patrick Doherty

Tentamen
TDDC17 Artificial Intelligence
25 august 2014 kl. 08-12

Points:

The exam consists of exercises worth 37 points.
To pass the exam you need 18 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 does it consist of? **[2p]**

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

   (c) Do you think the Physical Symbol System Hypothesis is true or false or somewhere in between? Provide reasonable justifications for your opinion. **[2p]**

3. Consider the following logical theory (where h and p are variables and john, hus are constants):

$$Lawyer(\mathsf{john}) \tag{1}$$

$$House(\mathsf{hus}, \mathsf{john}) \tag{2}$$

$$\forall p\, [Lawyer(p) \Rightarrow Rich(p)] \tag{3}$$

$$\forall p \exists h\; House(h, p) \tag{4}$$

$$\forall p \forall h\; House(h, p) \wedge Rich(p) \Rightarrow Big(h) \tag{5}$$

$$\forall h\, [(Big(h) \wedge \exists p\; House(h, p)) \Rightarrow Work(h)] \tag{6}$$

   This theory asserts that john is a lawyer; john lives in a house (hus) ; lawyers are rich; any person has a house; rich people have big houses; and big houses are a lot of work to maintain. We would like to show using resolution that John's house is a lot of work to maintain. To do this, answer the following questions:

   (a) Convert formulas (1) - (6) into clausal form with the help of appendix 1. **[2p]**

   (b) Prove that $Work(\mathsf{hus})$ is a logical consequence of (1) - (6) using the resolution proof procedure. **[2p]**

   - Your answer should be structured using a resolution refutation tree (as used in the book).
   - Since the unifications are trivial, it suffices to simply show the binding lists at each resolution step. Don't forget to substitute as you resolve each step.

4. Constraint satisfaction problems consist of a set of variables, a value domain for each variable and a set of constraints. 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. Appendix 3 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 1 arc consistent using the AC3 algorithm in Appendix 3. **[2p]**

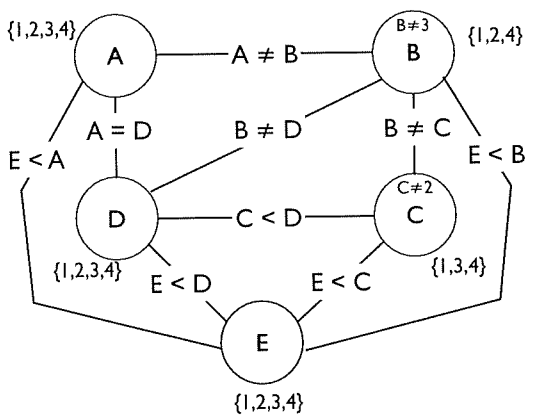


Figure 1: Constraint graph that is domain consistent but not arc consistent

5. The following problem models buggy LISP code where H = *the computer has hardware problems*, B = *there are bugs in the LISP code*, E = *the editor is running*, L = *the LISP interpreter is running*, F = *the cursor is flashing*, and P = *the prompt is displayed.* Use the Bayesian network in Figure 2 together with the conditional probability tables below to answer the following questions. Appendix 2 may be helpful to use.

   (a) Based on the independence assumptions implicit in the Bayesian network in Figure 2, write an equation for the full joint probability distribution $\mathbf{P}(P, F, L, E, B, H)$ as a product of conditional and non-conditional probabilities. **[1p]**

   (b) $\mathbf{P}(p, \neg f, l, \neg e, b, h)$**[1p]**

   (c) $\mathbf{P}(b \mid e, f, \neg p)$**[2p]**

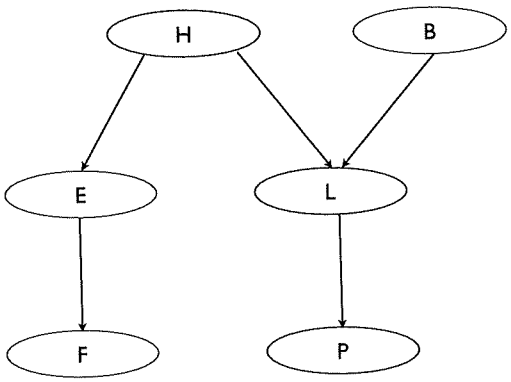


Figure 2: Bayesian Network Example

H	$\mathbf{P}(H)$
T	.1
F	.9

B	$\mathbf{P}(B)$
T	.7
F	.3

H	E	$\mathbf{P}(E \mid H)$
T	T	.2
T	F	.8
F	T	.75
F	F	.25

H	B	L	$\mathbf{P}(L \mid H, B)$
T	T	T	.65
T	T	F	.35
T	F	T	.9
T	F	F	.1
F	T	T	.1
F	T	F	.9
F	F	T	.2
F	F	F	.8

E	F	$\mathbf{P}(F \mid E)$
T	T	.8
T	F	.2
F	T	.1
F	F	.9

L	P	$\mathbf{P}(P \mid L)$
T	T	.7
T	F	.3
F	T	.2
F	F	.8

6. 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.

   The *Manhattan distance* between two locations is the shortest distance between the locations ignoring obstructions. Is the Manhattan distance in the example above an admissible heuristic? 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]**

7. The following questions pertain to automated planning.

   (a) Let $h_{\text{goal}}(s)$ be the number of facts that must be true in the goal but are not yet true in state s. Is this an admissible heuristic? If so, motivate. If not, give a clearly defined counterexample showing a state and a planning problem for which the heuristic is inadmissible. **[1p]**

   (b) Plan-space search begins with an *initial plan* and iteratively modifies this plan in order to repair *flaws*. Name the two distinct *types* of flaw that exist. Explain clearly how these flaw types can be identified in a plan. For the explanation to be sufficiently clear, you may need to provide an example plan drawn in the same way as during the lectures. **[2p]**

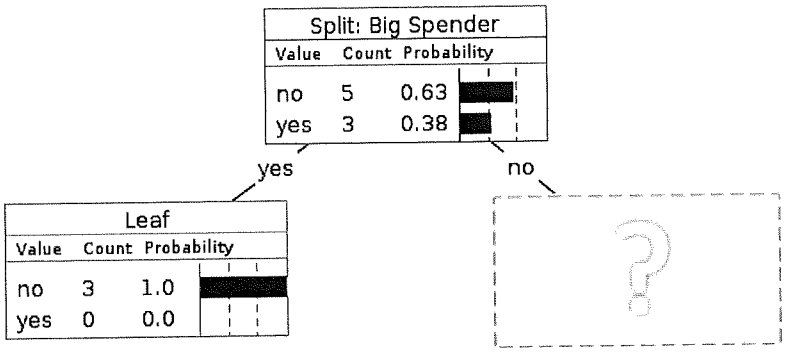
   (c) Explain the fundamental idea underlying *planning graphs*, as discussed in the planning lectures. In particular, what information about *states* does a planning graph contain, and how can this information be used to improve performance when searching for a plan? **[2p]**

No	Big Spender	Membership Card	Payment Option	Insurance?
1	yes	yes	credit	no
2	yes	yes	debit	no
3	yes	no	credit	no
4	no	no	debit	no
5	no	yes	credit	yes
6	no	no	debit	no
7	no	yes	credit	yes
8	no	no	credit	yes

8. You are designing an insurance recommendation agent for a web-shop that should recommend customers who buy a product an extra insurance if they are likely to want one. Some data on customer behavior is collected, of which a small subset is found in the table above. Your objective is to learn agent behavior from this data by training a decision tree classifier.

   (a) Below is an incomplete decision tree which by means of the attributes (features) *Big Spender*, *Membership Card*, and *Payment Option* attempts to correctly classify whether such a customer is likely to also want an extra product insurance. Value, Count and Probability in the boxes correspond to the values of the "Insurance" target concept (Yes/No), the number of examples at that point in the tree for each such value, and their proportions. Complete the missing branch of the decision tree *down to its leaves* using the Information Gain strategy (See Appendix 4) to decide which attribute to split on at each step. Show your calculations! **[2p]**

   (b) List and explain two common reasons for why naively training a supervised learning algorithm with a large hypothesis space, like the decision tree learning above, may not result in the best hypothesis for the problem to be solved? **[2p]**



Appendix 1

Converting arbitrary wffs to CNF form

1. Eliminate implication signs.
2. Reduce scopes of negation signs.
3. Standardize variables within the scopes of quantifiers (Each quantifier should have its own unique variable).
4. Eliminate existential quantifiers. This may involve introduction of Skolem constants or functions.
5. Convert to prenex form by moving all remaining quantifiers to the front of the formula.
6. Put the matrix into conjunctive normal form. Two useful rules are:
   - $\omega_1 \vee (\omega_2 \wedge \omega_3) \equiv (\omega_1 \vee \omega_2) \wedge (\omega_1 \vee \omega_3)$
   - $\omega_1 \wedge (\omega_2 \vee \omega_3) \equiv (\omega_1 \wedge \omega_2) \vee (\omega_1 \wedge \omega_3)$
7. Eliminate universal quantifiers.
8. Eliminate $\wedge$ symbols.
9. Rename variables so that no variable symbol appears in more than one clause.

Skolemization

Two specific examples. One can of course generalize the technique.
$\exists x P(x)$:
Skolemized: $P(\mathsf{c})$ where c is a fresh constant name.
$\forall x_1, \ldots, x_k, \exists y P(y)$:
Skolemized: $P(\mathsf{f}(x_1, \ldots, x_k))$, where f is a fresh function name.

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: 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

Appendix 4

Definition 1
Given a collection S, containing positive and negative examples of some target concept, the entropy of S relative to this boolean classification is

$$Entropy(S) \equiv -p_{\oplus} log_2 p_{\oplus} - p_{\ominus} log_2 p_{\ominus},$$

where $p_{\oplus}$ is the proportion of positive examples in S and $p_{\ominus}$ is the proportion of negative examples in S.

Definition 2
Given a collection S, containing positive and negative examples of some target concept, and an attribute A, the information gain, $Gain(S, A)$, of A relative to S is defined as

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{\mid S_v \mid}{\mid S \mid} Entropy(S_v),$$

where $values(A)$ is the set of all possible values for attribute A and S_v is the subset of S for which the attribute A has value v (i.e., $S_v = \{s \in S \mid A(s) = v\}$).

For help in converting from one logarithm base to another (if needed):

$$log_a x = \frac{log_b x}{log_b a}$$

$$ln\, x = 2.303\, log_{10}\, x$$

Note also that for the example, we define $0log0$ to be 0.