Intelligent Systems Programming (ISP)
BSc and MSc Exam

IT University of Copenhagen

29 May 2018

This exam consists of 8 numbered pages with 3 problems. Each problem is marked with the weight in percent it is given in the evaluation. You have 4 hours to complete the exam. **Notice that the first version of problem 3 only is to be solved by Bachelor (BSc) students, while the second version only is to be solved by Master (MSc) students**.

You can answer the questions in English or Danish.

Use notations and methods that have been discussed in the course.

Make appropriate assumptions when a problem has been incompletely specified.

Good luck!

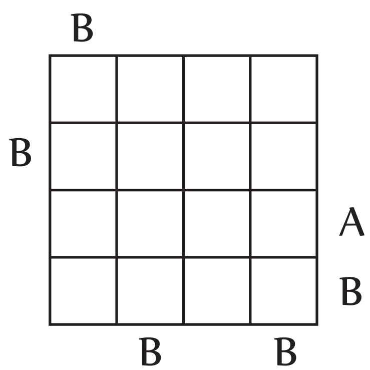
Problem 1: Constraint Programming (35%)

Consider the following letter puzzle. The puzzle consists of a board with $n \times n$ squares plus some clues appearing immediately next to the board at the beginning or end of a row or column (see figure (a) below). The clues are letters, and to solve the puzzle, each square must either be left empty or filled with a letter, such that:

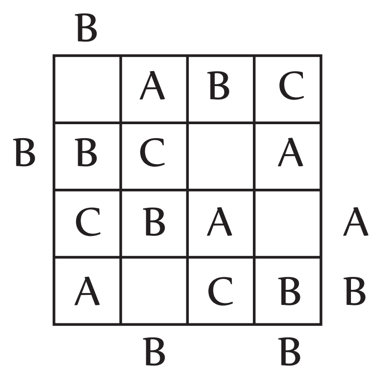
(i) In each row and each column, the $n-1$ first letters in the alphabet appear exactly once in a square, while one square is left blank.

(ii) A clue next to a column/row specifies the first letter to appear in that column/row when the column/row is read in the direction of the clue. That is, if e.g. an A appears to the right of the third row, then the first letter in the third row when read from right to left, has to be an A.

An example of a letter puzzle (a) and a possible solution (b) is shown below.



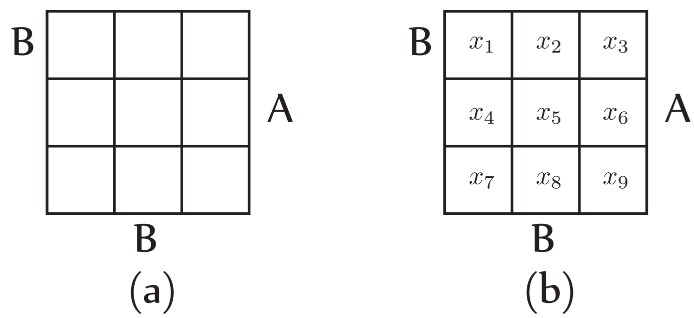
(a)



(b)

In the following we will look at the 3×3 letter puzzle shown in figure (a) below and formulate it as a CSP with 9 variables, $x_1, \ldots, x_9$, where x_i denotes the letter or blank space that should be put in the corresponding square shown in figure (b) below. Each variable x_i for $i \in \{1, 2, \ldots, 9\}$ has the domain $D_i = \{\circ, A, B\}$, where $\circ$ denotes an empty space.

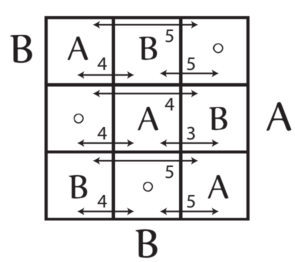
a) Write binary constraints corresponding to the rules in (i) and (ii) above applied to the puzzle shown in figure (a) below. Note that AllDiff constraints should be rephrased as a number of binary constraints.
Draw the constraint graph. In the graph, draw the nodes in a 3×3 grid as they appear in figure (b) below.



b) Make the CSP node consistent and write the domains for the variables, whose domains changed (if any). Make the CSP arc-consistent and write the domains for the variables, whose domains changed (if any).

c) Use the MAC algorithm to solve the puzzle. Draw the search tree with a node for each variable branched on. In the nodes, write which variable is branched on, and write the value assignment of each arc. In each node, write the remaining domains of each unassigned variable after arc-consistency is applied. Indicate failures and which domains are empty. You should use the MRV (minimum-remaining value) heuristic to choose the variable to branch on. If there is a tie, branch on the variables in numerical order of their subscripts. Assign the domain values in the order $\circ \prec A \prec B$.

d) Instead of MAC, use hill climbing to solve the problem. The initial state is shown below. The neighborhood of a state appears by swapping the content of two squares in the same row (thus, each state has a neighborhood of 9 other states). The cost of a state is the number of conflicts, where a conflict is a clue that is not satisfied or a column where A and B does not appear exactly once. You are allowed to take side-steps, and if so, you should not return immediately to the previous state. Indicate possible swaps for a state by arrows and write the cost of the resulting state above the arrow as shown for the initial state below.



Problem 2: Heuristic Search and Logic (35%)

Prof. Smart is working on using search to prove logical equivalences in propositional logic. The problem has five rules. Each rule is defined by a precondition and an effect as shown in the table below:

Rule	Precondition	Effect
R1	$x \Rightarrow \text{False}$	$\neg x \vee \text{False}$
R2	$\neg x \vee \text{False}$	$\neg x$
R3	$\neg x \wedge y$	$y \wedge \neg x$
R4	$x \wedge \neg y$	$\neg y \wedge x$
R5	$\neg x \wedge x$	False

False represents the propositional constant *False*, while x and y represent propositional symbols. A rule is applicable if its precondition matches a sub-sentence of a sentence in propositional logic. The result of applying a rule is that the matched sub-sentence is substituted with the effect of the rule. Notice that x and y only can match a single propositional symbol. They cannot match general sentences. Four examples of rule application are shown below. Notice in the last example that a parenthesis around a sub-sentence disappear if it no longer is needed.

$$
\begin{array}{ccc}
(A \Rightarrow \text{False}) \wedge \neg B & \xrightarrow{\text{Rule R1}} & (\neg A \vee \text{False}) \wedge \neg B \\
\neg B \wedge A \wedge \neg A & \xrightarrow{\text{Rule R3}} & A \wedge \neg B \wedge \neg A \\
\neg B \wedge A \wedge \neg A & \xrightarrow{\text{Rule R4}} & \neg B \wedge \neg A \wedge A \\
(\neg A \vee \text{False}) \wedge B & \xrightarrow{\text{Rule R2}} & \neg A \wedge B
\end{array}
$$

Assume that you are given a sentence S_0 in propositional logic and that you search for new sentences by using actions that correspond to applying these five rules.

a) Would this approach be sound in the sense that all new sentences are logically equivalent to S_0 (why/why not)?

b) Would this approach be complete in the sense that any sentence that is logically equivalent to S_0 can be found in this way (why/why not)?

c) Draw the complete state space that can be reached by applying the rules from the initial sentence $A \wedge (A \Rightarrow (\neg B \wedge B))$. Annotate each edge with the rule it corresponds to. A part of this state space is shown below.

$$
\boxed{A \wedge (A \Rightarrow (\neg B \wedge B))} \quad \underset{\text{R4}}{\overset{\text{R3}}{\rightleftarrows}} \quad \boxed{A \wedge (A \Rightarrow (B \wedge \neg B))}
$$

Prof. Smart wants to use his search approach to determine if a sentence is logically equivalent to *False*. He claims that if the cost of all rule applications is one, a valid heuristic for this search problem is to count the number of propositional symbols and constants and subtract one. As an example, the heuristic value of $A \wedge (A \Rightarrow \text{False})$ is two since there are two propositional symbols in the sentence (A and A) and one constant (False).

d) Is the heuristic valid (why/why not)?

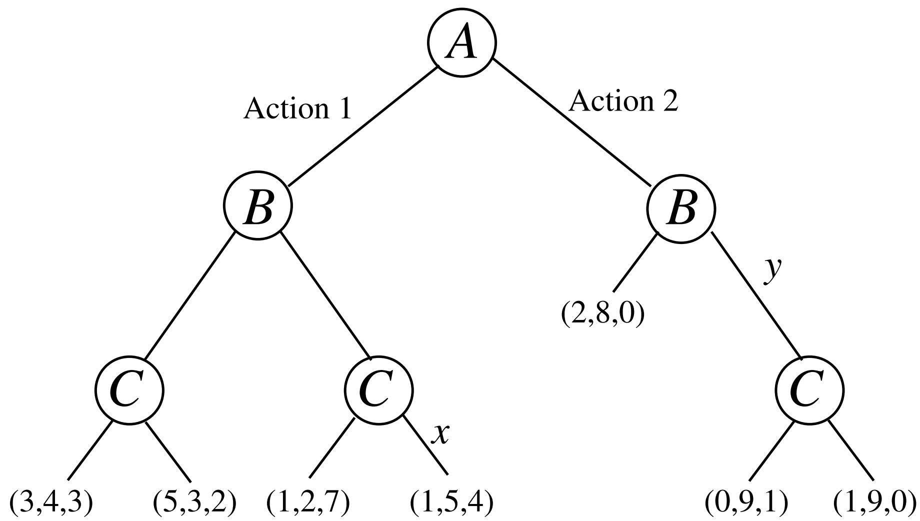
e) Is the heuristic admissible (why/why not)?

f) Is the heuristic consistent (why/why not)?

g) Which sequence of rule applications corresponds to the solution returned by A* Tree Search given an initial state of the sentence $A \wedge (A \Rightarrow (\neg B \wedge B))$ and using Prof. Smarts heuristic?

Problem 3: FOR BSC STUDENTS ONLY (30%)

The figure below shows a game tree of a turn-taking game with three players A, B, and C.



Each leaf-node is a terminal state. Its utility is given as a triple (a, b, c), where a, b, and c are the utility of player A, B, and C, respectively.

a) The game is zero-sum. How do the utilities in the game tree show this?

b) Write the multi-player minimax value next to each internal node of the tree.

c) What is the minimax decision for player A in the root node and why?

The MINIMAX function below can be used to calculate the minimax value of a state of this 3-player game.

function MINIMAX(*state*) **returns** the Minimax value of *state*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow (-\infty, -\infty, -\infty)$
  for each a **in** ACTIONS(*state*) **do**
    $w \leftarrow$ MINIMAX(RESULT(*state*,a))
    if VALUE(w,PLAYER(*state*)) $>$ VALUE(v,PLAYER(*state*)) **then**
      $v \leftarrow w$
  return v

TERMINAL-TEST(s) returns true if s is a terminal state of the game. UTILITY(s) returns the utility triple of a terminal state s. ACTIONS(s) returns the applicable actions of the player that is to take a turn in state s. RESULT(s,a) returns the state resulting from executing action a in state s. VALUE(u,p) returns the utility of player p in utility triple u. Finally, PLAYER(s) returns the player that is to take a turn in state s.

d) Write pseudo-code of a function MINIMAX-DECISION(s) that returns the minimax decision of the player that is to take a turn in state s.

e) Prof. Smart claims that if the MINIMAX function explores actions from left to right in the tree, it can make cuts similar to α and β cuts at edge x and y. Is this correct (why/why not)?

Problem 3: FOR MSC STUDENTS ONLY (30%)

Linear Programming

Consider the dual LP problem shown below.

$$
\begin{array}{lrcrcr}
\text{Minimize} & -y_1 & & & & \\
\text{Subject to} & y_1 & - & y_2 & \geq & 0 \\
 & & & -y_2 & \geq & -4 \\
 & & & y_1, y_2 & \geq & 0
\end{array}
$$

a) Write the dual LP problem on standard form.

b) Draw the geometric interpretation of the dual LP problem.

c) Determine based on your answer to b) alone, whether the primal LP problem is feasible, infeasible, or unbounded. Explain your answer.

d) Write the primal LP problem.

Binary Decision Diagrams

Consider the following Boolean expression

$$(x \Rightarrow y) \wedge (\neg x \wedge \neg y)$$

e) Draw the BDD for the expression using variable order $x \prec y$.

f) Is the expression valid, unsatisfiable or neither? Give a reason for your answer.