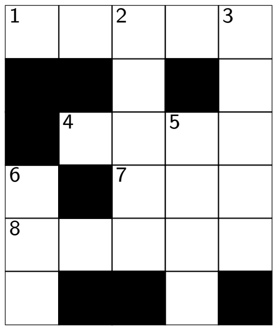
COMP3702/7702 Artificial Intelligence
Semester 2, 2020
Tutorial 6 - Sample Solutions

Exercise 6.1

The crossword puzzle from last tutorial:



Words: AFT, ALE, EEL, HEEL, HIKE, HOSES, KEEL, KNOT, LASER, LEE, LINE, SAILS, SHEET, STEER, TIE.

This can be modeled as a CSP, with:

Variables: 1-across, 2-down, 3-down, 4-across, 5-down, 6-down, 7-across and 8-across

Domains: After applying node consistency

1-across: {HOSES, LASER, SAILS, SHEET, STEER}

2-down: {HOSES, LASER, SAILS, SHEET, STEER}

3-down: {HOSES, LASER, SAILS, SHEET, STEER}

4-across: {HEEL, HIKE, KEEL, KNOT, LINE}

5-down: {HEEL, HIKE, KEEL, KNOT, LINE}

6-down: {AFT, ALE, EEL, LEE, TIE}

7-across: {AFT, ALE, EEL, LEE, TIE}

8-across: {HOSES, LASER, SAILS, SHEET, STEER}

Constraints: Letters used in two between words (nb. all constraints are binary).

a) Backtracking search can be manually traced or implemented by following the pseudocode below, in which backtracking search is described recursively:

```
function BACKTRACKING-SEARCH(csp) returns solution/failure
    return RECURSIVE-BACKTRACKING({ }, csp)
function RECURSIVE-BACKTRACKING(assignment, csp) returns soln/failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment given CONSTRAINTS[csp] then
            add {var = value} to assignment
            result ← RECURSIVE-BACKTRACKING(assignment, csp)
            if result ≠ failure then return result
            remove {var = value} from assignment
    return failure
```

Note that this pseudocode includes containers or objects for `Variables` and `Constraints`.

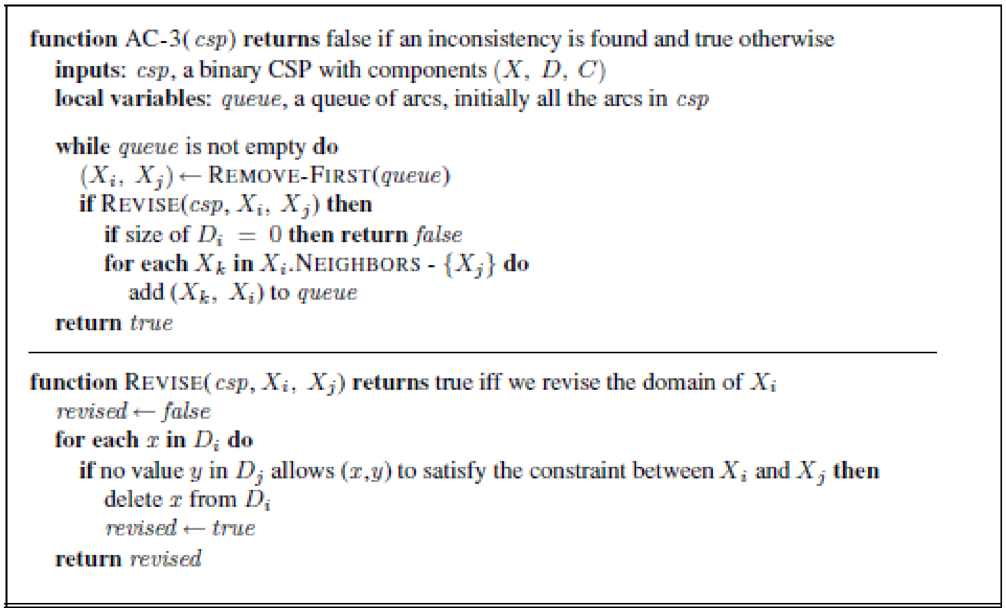
As a demonstration, we can manually trace the `Backtracking-Search` algorithm:

1. Select 1-across, as the first unassigned variable.
2. Check if the first value in the domain of 1-across conflicts with the existing (empty) assignment; it doesn't, so expand a node for 1-across=HOSES.
3. Now select 2-down, as the next unassigned variable; this is using the recursive function call.
4. Check if the first value in the domain of 2-down, HOSES, conflicts with the existing assignment — it does! Throw it away.
5. The second value, LASER also conflicts with the existing assignment, so throw it away. At this point we have iterated over the inner **if** statement twice.
6. The third value in the domain, 2-down=SAILS is consistent with 1-across=HOSES, so expand a child node of 1-across=HOSES given by 2-down=SAILS
7. Select 3-down as the next unassigned variable.
8. ...

Eventually, you will find the solution as:

H	O	S	E	S
■	■	A	■	T
■	H	I	K	E
A	■	L	E	E
L	A	S	E	R
E	■	■	L	■

b) There are a few ways of implementing arc-consistency. One *efficient* algorithm, commonly called AC-3, is described in the pseudocode below (note that the method `Revise` is stated separately for easy comprehension):



```
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 6.3 The arc-consistency algorithm AC-3. After applying AC-3, either every arc is arc-consistent, or some variable has an empty domain, indicating that the CSP cannot be solved. The name "AC-3" was used by the algorithm's inventor (Mackworth, 1977) because it's the third version developed in the paper.

If you apply the arc-consistency algorithm correctly, you will find that it solves the crossword puzzle, and no further search is required.

c) AC-3 solves the problem in many fewer operations that backtracking search.

Exercise 6.2

We can encode Mr Jones' knowledge of the situation into a set of logical sentences, and use these sentences to determine if there is a situation where one of the chests is guaranteed to contain gold. Let the variable A represent the situation where trunk A contains gold, so that ¬A means trunk A is empty, and similarly for B and C.

Mr Jones knows one trunk contains gold, while the other two are empty. This doesn't give us any information about *which* exactly has gold, so we need to consider each possibility. Expressed in a logic formula, the sentence is:

$$S_1 = (A \land \neg B \land \neg C) \lor (\neg A \land B \land \neg C) \lor (\neg A \land \neg B \land C)$$

The three clues are:

1. $\neg A$
2. $\neg B$
3. B

Mr Jones knows that only one of these is true, and the other two are false. Again, we need to consider each possibility to form our second logical sentence:

$$S_2 = (\neg A \wedge \neg(\neg B) \wedge \neg B) \vee (\neg(\neg A) \wedge \neg B \wedge \neg B) \vee (\neg(\neg A) \wedge \neg(\neg B) \wedge B)$$

Double negation elimination gives

$$(\neg A \wedge B \wedge \neg B) \vee (A \wedge \neg B \wedge \neg B) \vee (A \wedge B \wedge B)$$

Using $X \wedge \neg X = F$ (because we can't have something true and false at the same time) and $X \wedge X = X$ (because AND-ing the same thing is redundant) we get:

$$(\neg A \wedge F) \vee (A \wedge \neg B) \vee (A \wedge B)$$

AND-ing with false is always false:

$$F \vee (A \wedge \neg B) \vee (A \wedge B)$$

OR-ing with false is just the statement being OR-ed with:

$$(A \wedge \neg B) \vee (A \wedge B)$$

Using distributivity:

$$(A \wedge (\neg B \vee B)$$

Saying something can be false or true is not giving us any information – it is really just saying "true", and AND-ing something with true is itself. So we are left with

$$S_2 : A$$

What this means is that to satisfy S_2, A must be true, i.e. the gold is in trunk A. Are we done? Well no, we still need to check that there is a combination of truth values for A, B, C that makes S_1 true. If we didn't do that, there might be a possibility that the gold is in trunk A but also in one of the other trunks, which violates the requirement that it be in exactly one trunk. So let's do a truth table to figure this out (See Table 1).

S_1 looks like a complicated sentence, but it is actually easy to compute the truth values intuitively. Remember what it says in words: one trunk contains gold, the other two are empty. So S_1 is true whenever exactly one of A, B, C on the left of the table is true, and false otherwise.

Since there is only one case where both sentences are true, we can indeed guarantee that the gold is in trunk A only.

Question 6.3

When we are asked to determine if an entailment is correct (or holds, or is true) we can convert the entailment into an implication, and check if the implication is valid. Checking whether or not the implication is valid means solving a validity problem. Remember that a sentence is valid when every combination of variable assignments in the sentence causes it to be true.

A	B	C	$S_1 = (A \land \neg B \land \neg C)$ $\lor(\neg A \land B \land \neg C)$ $\lor(\neg A \land \neg B \land C)$	$S_2 = A$
F	F	F	F	F
F	F	T	T	F
F	T	F	T	F
F	T	T	F	F
T	F	F	T	T
T	F	T	F	T
T	T	F	F	T
T	T	T	F	T

Table 1: Truth table

a)

$$(A \land B) \models (A \Leftrightarrow B)$$

Convert to an implication:

$$(A \land B) \to (A \Leftrightarrow B)$$

Construct a truth table and check if the implication is valid:

A	B	$A \land B$	$A \Leftrightarrow B$	$(A \land B) \to (A \Leftrightarrow B)$
F	F	F	T	T
F	T	F	F	T
T	F	F	F	T
T	T	T	T	T

Table 2: Truth table

Every row in the implication column of Table 2 is true, therefore the implication is valid, and the original entailment is correct.

b)

$$(A \Leftrightarrow B) \models (A \land B)$$

Convert to an implication:

$$(A \Leftrightarrow B) \to (A \land B)$$

Construct a truth table and check if the implication is valid:

Not every row in the implication column is true (the first row is false), so therefore the implication is not valid, and the original entailment is not correct.

A	B	$A \Leftrightarrow B$	$A \wedge B$	$(A \Leftrightarrow B) \rightarrow (A \wedge B)$
F	F	T	F	F
F	T	F	F	T
T	F	F	F	T
T	T	T	T	T

Table 3: Truth table

Question 6.4

a) Let B represent bumper cars is open, C represent carousel, H represent haunted class, R represent roller coaster, and F represent ferris wheel is open.

Write the constraints in clauses:

(a) $B \vee C$

(b) $\neg B \rightarrow R$ which is $B \vee R$

(c) $C \rightarrow (B \vee H)$ which is $\neg C \vee B \vee H$

(d) $H \rightarrow F$ which is $\neg H \vee F$

(e) $\neg(B \wedge F)$ which is $\neg B \vee \neg F$

(f) $R \rightarrow F$ which is $\neg R \vee F$

(g) $\neg R \rightarrow (H \vee F)$ which is $R \vee H \vee F$

Final Conjunctive Normal Form (CNF):

$$(B \vee C) \wedge (B \vee R) \wedge (\neg C \vee B \vee H) \wedge (\neg H \vee F) \wedge (\neg B \vee \neg F) \wedge (\neg R \vee F) \wedge (R \vee H \vee F)$$