CS 498mp3: Logic in Computer Science Spring 2017: Homework 1
Thursday 9:30am, February 16th
Hand over in class, before lecture begins.
*** Solutions ***

---

1. **Proof by structural induction** [20 points]

   Consider a formal syntax for well-formed propositional logic give as:

   $$\textit{Formulas}\ \alpha, \beta ::- \ \ p_i \mid (\neg\alpha) \mid (\alpha \vee \beta) \mid (\alpha \wedge \beta)$$

   where $P = \{p_1, p_2 \ldots, \}$ is an infinite set of propositions, and where $i \in \mathbb{N}$ above.

   Prove formally that in any formula α, if c is the number of binary propositional connectives ($\vee$ and $\wedge$) and n is the number of occurrences of propositions in α, then $n = c+1$. Prove this by structural induction on formulas.

   Solution: For any formula α, let $c(\alpha)$ denote the set of binary propositional connectives in α and let $n(\alpha)$ denote the number of occurrences of propositions in α.

   We need to prove that for every formula α, $n(\alpha) = c(\alpha) + 1$.

   We will prove this by structural induction.

   Base case: Let $\alpha = p_i$, where $p_i \in P$.
   Then clearly $n(\alpha) = 0$ and $c(\alpha) = 1$, and hence $n(\alpha) = c(\alpha) + 1$.

   Induction step:
   Case 1: Let $\alpha = (\neg\beta)$.
   Then by the induction hypothesis, we know that $n(\beta) = c(\beta) + 1$.
   Since $n(\alpha) = n(\beta)$ and $c(\alpha) = c(\beta)$, it follows that $n(\alpha) = c(\alpha) + 1$.

   Case 2: Let $\alpha = (\beta_1\ op\ \beta_2)$, where $op \in \{\wedge, \vee\}$.
   Then by the induction hypothesis, we know that
   $n(\beta_1) = c(\beta_1) + 1$ and $n(\beta_2) = c(\beta_2) + 1$.
   Now, $n(\alpha) = n(\beta_1) + n(\beta_2)$ and $c(\alpha) = c(\beta_1) + c(\beta_2) + 1$.
   So it follows that $n(\alpha) = n(\beta_1) + n(\beta_2) = c(\beta_1) + c(\beta_2) + 2 = c(\alpha) + 1$. □

2. **Resolution** [20 points]

   Prove the following is valid using resolution:

   $$((p \Rightarrow q) \wedge (r \Rightarrow s)) \Rightarrow ((p \vee r) \Rightarrow (q \vee s))$$

   To prove validity using resolution, we first negate the formula to get:

   $$\neg(((p \Rightarrow q) \wedge (r \Rightarrow s)) \Rightarrow ((p \vee r) \Rightarrow (q \vee s)))$$

   $$\equiv ((p \Rightarrow q) \wedge (r \Rightarrow s)) \wedge \neg((p \vee r) \Rightarrow (q \vee s))$$

$$\equiv ((p \Rightarrow q) \wedge (r \Rightarrow s)) \wedge ((p \vee r) \wedge \neg q \wedge \neg s))$$

This can be written now in CNF as:

$$(\neg p \vee q) \wedge (\neg r \vee s) \wedge (p \vee r) \wedge \neg q \wedge \neg s$$

Hence we can write this as a set of clauses: $\{\neg p, q\}, \{\neg r, s\}, \{p, r\}, \{\neg q\}, \{\neg s\}$

Resolving $\{\neg p, q\}$ and $\{\neg q\}$ wrt q gives: $\{\neg p\}$.

Resolving $\{\neg p\}$ and $\{p, r\}$ wrt p gives: $\{r\}$.

Resolving $\{r\}$ and $\{\neg r, s\}$ wrt r gives: $\{s\}$.

Resolving $\{s\}$ and $\{\neg s\}$ wrt s gives: $\{\}$.

Hence we have proved validity of the original formula. □

3. **Understanding König's Lemma** [30 points]

There is a gigantic box with balls, each marked with a natural number. There are infinitely many balls in this box, and in fact infinitely many balls marked with the same number.

A robot (called Golem, if you must know, and yes, he is made of clay) sets out by pulling out a single ball with some number n, and places it in his basket. Now, in every round, he takes one ball from his basket, say with a number i on it, and puts it back in the box, and takes out *any* number of balls from the box with the condition that all of them have the same number on them, but this number is less than i.

For example, in a round, he may transfer a ball numbered 5 from the basket to the box, and take back 30 balls labeled 4. Or put one ball numbered 5 into the box and take back 3413343 balls labeled 3. His basket has unbounded capacity.

Note that if Golem transfers a ball numbered 0 from the basket to the box, he cannot take back any ball.

Prove formally that no matter how Golem goes about picking the balls, he will eventually empty his basket.

Hint: Use König's Lemma.

Let us represents Golum's moves using a tree, where the nodes are labeled with numbers. The root of the tree is labeled by the number n on the single ball that Golum picks in the beginning. Whenever Golum puts a ball numbered i in the box and takes out some number of balls, we represent this by adding children to the node representing the ball numbered i, one child for each ball Golum takes out and label it using the number that is on it.

Clearly the tree constructed is such that every node has a finite number of children, since Golum draws only a finite number of balls each time he puts in a ball. Also, clearly, in any path from the root in this tree, the labels on the numbers *strictly decrease* since Golum always replaces a ball with balls with strictly smaller numbers. Consequently, the paths in the tree are finite (in fact, are at most of length n).

Now, if Golum does not eventually empty his basket, then he goes on forever, and the tree is infinite. Since the tree is finitely branching and has no infinite path, the tree is finite. This contradiction proves that Golum will eventually empty his basket. □

4. **Modeling using propositional logic** [30 points]

(a) Three boxes are in a room; let's call them Ouro, Zoloto, and Thangam. One contains gold, the other two are empty. Each box has imprinted on it a clue as to its contents; the clues in each box are:

(Ouro) "The gold is not in this box",

(Zoloto) "The gold is not in this box",

(Thangam) "The gold is in Box Zoloto".

Only one message is true; the other two are false.

You want to find out which box contains the gold.

Model the problem as a satisfiability problem in propositional logic. Use three variables O, Z, T, to denote whether the gold is in the Ouro, Zoloto, or Thangam box, respectively. Use three other variables to denote the truthhood of each of the labels on the boxes. Model all constraints using formulas, and feed them to a SAT solver to find what's true.

Present both your hand-written constraint and the SAT solver's answer.

Note: You can use Z3 for SAT solving online at `http://www.rise4fun.com/z3`. Here's a sample syntax for checking whether $(p \vee q) \wedge (p \Rightarrow (\neg q)) \wedge (q \Rightarrow (\neg p))$ is satisfiable, and to ask Z3 to give you a model.

```
(declare-const p Bool)
(declare-const q Bool)
(assert
    (and
       (or p q)
       (=> p (not q))
       (=> q (not p))
    )
)
(check-sat)
(get-model)
```

Let Ot, Zt and Tt be propositional variables that are true iff the statement written on the boxes Ouro, Zoloto, and Thangam are true, respectively.

Then we express the following constraints:

- One of the three boxes contain gold while the other two do not:

$$(O \wedge \neg Z \wedge \neg T) \vee (Z \wedge \neg O \wedge \neg T) \vee (T \wedge \neg O \wedge \neg Z)$$

- One message is true, while the other two are false:

$$(Ot \wedge \neg Zt \wedge \neg Tt) \vee (Zt \wedge \neg Ot \wedge \neg Tt) \vee (Tt \wedge \neg Ot \wedge \neg Zt)$$

- Ot is true iff the message on the box Ouro is true:

$$Ot \Leftrightarrow \neg O$$

- Zt is true iff the message on the box Zoloto is true:

$$Zt \Leftrightarrow \neg Z$$

- Tt is true iff the message on the box Zoloto is true:

$$Tt \Leftrightarrow Zt$$

Modeling the above in Z3 using the following encoding of the SAT constraint:

```
(declare-const O Bool)
(declare-const Z Bool)
(declare-const T Bool)
(declare-const Ot Bool)
(declare-const Zt Bool)
(declare-const Tt Bool)

(assert
  (and
        (or (and O (not Z) (not T))
            (and Z (not O) (not T))
            (and T (not O) (not Z)))

        (or (and Ot (not Zt) (not Tt))
            (and Zt (not Ot) (not Tt))
            (and Tt (not Ot) (not Zt)))

        (iff Ot (not O))
        (iff Zt (not Z))
        (iff Tt Z)

  ))
(check-sat)
(get-model)
```

gives the model where O and Zt are true and the rest of the variables are false, showing that the gold being in Ouro is consistent with all constraints (with the statement on the Zoloto box being the only one that's true).

Adding the following conjunct to the above formula to ask for another solution:

```
(not (and O (not Z) (not T) (not Ot) Zt (not Tt)))
```

gives an unsatisfiable formula, showing that setting O and Zt to true and the rest to false is the only solution. Hence the gold must be in the Ouro box. □

(b) Using a similar technique as above, model the following in propositional logic, using an appropriate set of propositions, and solve it using Z3.

There are three suspects for a murder: Adams, Brown, and Clark. Adams says "I didnt do it. The victim was an old acquaintance of Brown's. But Clark hated him." Brown states "I didn't do it. I didn't even know the guy. Besides I was out of town all that week." Clark says "I didn't do it. I saw both Adams and Brown downtown with the victim that day; one of them must have done it." Assume that the two innocent men are telling the truth, but that the guilty man might not be. Who did it?

Let us introduce the following propositions:

- A: Adams is the murderer
- B: Brown is the murderer
- C: Clark is the murderer
- vBf: the victim and Brown are friends
- Chv: Clark hates the victim
- Ao: Adams was out of town
- Bo: Brown was out of town

We can now capture that if Adams is not the murderer, then what he says is true:

$$\neg A \Rightarrow (\neg A \wedge vBf \wedge Chv)$$

We capture that if Brown is not the murderer, then what he says is true:

$$\neg B \Rightarrow (\neg B \wedge \neg vfB \wedge Bo)$$

And we capture that if Clark is not the murderer, then what he says is true:

$$\neg C \Rightarrow (\neg C \wedge \neg Bo \wedge \neg Ao \wedge (A \vee B))$$

We now express that one of these people is the murderer:

$$A \vee B \vee C$$

We can also express that the other two are not:

$$(A \Rightarrow (\neg B \wedge \neg C)) \;\wedge\; (B \Rightarrow (\neg A \wedge \neg C)) \;\wedge (C \Rightarrow (\neg A \wedge \neg A))$$

We can now ask whether the conjunction of the above formulas is satisfiable, in Z3:

```
(declare-const A Bool)
(declare-const B Bool)
(declare-const C Bool)
(declare-const vBf Bool)
(declare-const Chv Bool)
(declare-const Ao Bool)
(declare-const Bo Bool)

(assert
  (and
  (=> (not A) (and (not A) vBf Chv))
  (=> (not B) (and (not B) (not vBf) Bo))
  (=> (not C) (and (not C) (not Bo) (not Ao) (or A B)))
  (or A B C)
  (=> A (and (not B) (not C)))
  (=> B (and (not A) (not C)))
  (=> C (and (not A) (not B)))
  )
)

(check-sat)
(get-model)
```

This gives a model where B is true (and A and C are false). Adding the constraint $\neg B$ to the above formula gives unsat in Z3. Consequently, the only way to satisfy the constraints is that B is true. Hence Bob is the murderer. $\square$