Propositional Logic: Basic Logic

Due Nov 13, 2020 by 11:59pm **Points** 12 **Submitting** a file upload **File Types** py

This is a short assignment that has the goal of getting you familiar with using a propositional logic as a knowledge representation language in order to solve some problems.

You will need to download and unzip the code in **simple_sat.zip (https://canvas.umn.edu/courses/193654/files/16556699/download?download_frd=1)** . This zip file contains a few python modules that you can use to help determine satisfiability for propositional logic knowledge bases. The module that you will use is sat_interface, and an example of using it is in the writeup below.

The code I included there is adapted from **https://github.com/sahands/simple-sat (https://github.com/sahands/simple-sat)**

Note: If you really don't want to use python, you are welcome to find a sat solver for your language of choice and use that instead, but make sure that you include all the necessary code to be able to run your program when you submit.

An Example Problem

Suppose that liars always speak what is false, and truth-tellers always speak what is true. Further suppose that Amy, Bob, and Cal are each either a liar or truth-teller. Amy says, "Bob is a liar." Bob says, "Cal is a liar." Cal says, "Amy and Bob are liars." Which, if any, of these people are truth-tellers?

To solve this using propositional logic, we need to define this problem in those terms. Let us consider the following three boolean variables to use in our knowledge base:

- **A** Amy is a truth-teller
- **B** Bob is a truth-teller
- **C** Cal is a truth-teller

With these three variables, we will build sentences from logical connectives that fully encompass all the relationships described in the problem.

Let us consider the first piece of information:

Amy says, "Bob is a liar."

Amy is making the claim that Bob is not a truth-teller. If Amy is a truth-teller, then Bob must be a liar. But conversely, if Bob is a truth-teller, then Amy must be a liar. Therefore, we can encode this piece of information as:

$A \Leftrightarrow \neg B$

Similarly, the second and third sequences can be encoded as

$B \Leftrightarrow \neg C$

and

$C \Leftrightarrow (\neg A \wedge \neg B)$

Next, we'll need to convert this to conjunctive normal form. After some work, you would get the following clauses:

$\neg A \vee \neg B$
$B \vee A$
$\neg B \vee \neg C$
$C \vee B$
$\neg C \vee \neg A$
$\neg C \vee \neg B$
$A \vee B \vee C$

To represent this information in our python sat interface, you would create a list of strings, with each string representing a separate clause, "or" symbols left as spaces, and "not" symbols replaced with "~", as follows:

(Note: this example uses the python interactive shell, you'll probably want to write your code inside an actual module.)

```
>>> import sat_interface
>>> example_prob = sat_interface.KB(["~A ~B", "B A", "~B ~C", "C B", "~C ~A", "~C ~B", "A B C"])
>>> example_prob.is_satisfiable()
True
```

So this would create the knowledge base, and then test its satisfiability. Great! Except I am interested in knowing specifics! So, to check a specific piece of knowledge, I can try using the test_literal() method:

```
>>> example_prob.test_literal("A")
False
>>> example_prob.test_literal("~A")
True
```

This pair of tests tells me that the knowledge base is unsatisfiable when I add the clause **A**, and remains satisfiable when I add the clause **~A**, so I can conclude that Amy is, in fact, a liar.

Note: I may need to test both A and ~A to draw conclusions! (Why?)

Your task

Write a python module named proplogic.py that uses the sat_interface module to create and solve each of the following logic problems. Your "solution" should be contained in a function that, when called, creates the appropriate clauses, then tests each of the literals in the problem, then prints out its results.

Liars and Truth-tellers II

Three people, Amy, Bob, and Cal, are each either a liar or a truth-teller. Assume that liars always lie, and truth-tellers always tell the truth.

- Amy says, "Cal and I are truthful."
- Bob says, "Cal is a liar."
- Cal says, "Bob speaks the truth or Amy lies."

What can you conclude about the truthfulness of each?

Write your answer in a function named *tt2*

Liars and Truth-tellers III

Three people, Amy, Bob, and Cal, are each either a liar or a truth-teller. Assume that liars always lie, and truth-tellers always tell the truth.

- Amy says, "Cal is not honest."
- Bob says, "Amy and Cal never lie."
- Cal says, "Bob is correct."

What can you conclude about the truthfulness of each?

Write your answer in a function named *tt3*

Robbery and a Salt

The salt has been stolen! Well, it was found that the culprit was either the Caterpillar, Bill the Lizard or the Cheshire Cat. The three were tried and made the following statements in court:

CATERPILLAR: Bill the Lizard ate the salt.

BILL THE LIZARD: That is true!

CHESHIRE CAT: I never ate the salt.

As it happened, at least one of them lied and at least one told the truth. Who ate the salt?

Write your answer in a function named *salt*

An honest name

Three golfers named Tom, Dick, and Harry are walking to the clubhouse.

The first man in line says, "The guy in the middle is Harry."

The man in the middle says, "I’m Dick."

The last man says, "The guy in the middle is Tom."

Tom, the best golfer of the three, always tells the truth.

Dick sometimes tells the truth, while Harry, the worst golfer, never does.

Figure out who is who.

Write your answer in a function named *golf*

Submission

Submit your answers in a module named *proplogic.py*

Do not submit other python code, you may assume that we will have access to the sat_interface code.

Propositional Logic: Basic Logic Rubric

Criteria	Ratings		Pts
problem tt2 KB: 1 point code: 1 point result: 1 point	**3 to >0.0 pts** **Full Marks**	**0 pts** **No Marks**	3 pts
problem tt3 KB: 1 point code: 1 point result: 1 point	**3 to >0.0 pts** **Full Marks**	**0 pts** **No Marks**	3 pts
problem salt KB: 1 point code: 1 point result: 1 point	**3 to >0.0 pts** **Full Marks**	**0 pts** **No Marks**	3 pts
problem golf KB: 1 point code: 1 point result: 1 point	**3 to >0.0 pts** **Full Marks**	**0 pts** **No Marks**	3 pts
			Total Points: 12