

Due May 8, 2:45pm

Instructions: You may work in groups of up to 4 people, but make sure to list your study partners on your homework. As always, you are required to write up your homeworks yourself, and you must never share your write-ups with others.

1. (30 pts.) Beyond suspicion

Most researchers suspect that $\mathbf{P} \neq \mathbf{NP}$. We also know that if $\mathbf{P} \neq \mathbf{NP}$, then $3\text{SAT} \notin \mathbf{P}$. Therefore it seems reasonable to say that $3\text{SAT} \notin \mathbf{P}$ is suspected to be true.

Mark each of the following statements as either True, False, Suspected True, or Suspected False. True means that the statement is provably true. Suspected True means that the statement has not been proven to be true, but it can be proven to follow from a conjecture that is widely suspected to hold (such as that $\mathbf{P} \neq \mathbf{NP}$). Similarly for False and Suspected False. You don't need to justify your answer.

- (a) HAMILTONIAN PATH $\in \mathbf{P}$.
- (b) HAMILTONIAN PATH $\in \mathbf{NP}$.
- (c) HAMILTONIAN PATH is NP-complete.
- (d) There is a polynomial-time algorithm for the 3-COLORING problem.
- (e) There is no polynomial-time algorithm for the TRAVELLING SALESMAN problem.
- (f) HAMILTONIAN PATH $\in \mathbf{P}$ if and only if 3-COLORING $\in \mathbf{P}$.
- (g) 2-COLORING $\in \mathbf{P}$.
- (h) 2-COLORING is NP-complete.

2. (30 pts.) Learn to use a SAT solver

This question will introduce you to a SAT solver and teach you how to use it. Most SAT solvers are not very user-friendly: they expect to be provided a boolean formula in CNF (Conjunctive Normal Form), in a particular input format. Therefore, I'm going to introduce you to STP, a solver that has a more user-friendly interface. STP works by pre-processing your input, to generate a boolean formula in CNF, and then invoking a SAT solver.

To use STP, you declare a number of boolean variables, write down some constraints on the boolean variables that must be true (some boolean formulas that must be true), and then STP tries to see whether it can find any assignment to the boolean variables that makes all of the constraints true. If it can find an assignment, it reports "Satisfiable." and outputs a satisfying assignment; otherwise, it reports "Unsatisfiable." You can declare boolean variables x, y like this:

```
x, y : BOOLEAN;
```

Let's say we want to test whether the formula $(x \vee \neg y) \wedge (\neg x \vee y)$ is satisfiable. We're going to introduce a constraint that encodes this formula.

```
x, y : BOOLEAN;  
ASSERT((x OR NOT(y)) AND (NOT(x) OR y));
```

Log into a Linux machine (`ilinux1.eecs.berkeley.edu`, `ilinux2.eecs`, or `ilinux3.eecs`), store the constraints above into an input file, say `test1.in`, and run the following command:

```
/home/ff/cs170/bin/easystp test1.in
```

You should get the following output:

```
Satisfiable.  
ASSERT( x  = FALSE  );  
ASSERT( y  = FALSE  );
```

The first line tells you that the formula is satisfiable. The second line gives you an example of a satisfying assignment: namely, $x = y = \text{false}$.

Note that STP can only be run from instructional machines that are running Linux. So if you get an error message, double-check that you are logged into `ilinux1.eecs.berkeley.edu`, `ilinux2.eecs`, or `ilinux3.eecs`.

It is OK to introduce multiple `ASSERT` statements. The STP solver will look for a satisfying assignment that makes every `ASSERT` statement true. For instance, an equivalent way to do the above example would have been:

```
x, y : BOOLEAN;  
ASSERT(x OR NOT(y));  
ASSERT(NOT(x) OR y);
```

Try it; you'll see you get the same answer. STP supports other boolean operators as well: \Rightarrow is logical implication ($x \Rightarrow y$ can be encoded in STP as $x \Rightarrow y$); \Leftrightarrow is bidirectional implication, i.e., equality of booleans ($x = y$ could be encoded in STP as $x \Leftrightarrow y$); XOR is exclusive or; and NAND is `nand` ($x \text{ NAND } y$ is equivalent to $\text{NOT}(x \text{ AND } y)$).

Here is another example. Let's say we want to test whether the formula $((x \Rightarrow y) \Rightarrow (y \Rightarrow x)) \wedge (x \vee y) \wedge \neg y$ is satisfiable. One way to do so would be as follows:

```
x, y : BOOLEAN;  
ASSERT(((x => y) => (y => x)) AND (x OR y) AND NOT(y));
```

Another approach would be to introduce temporary variables for some of the subexpressions, and break this down into multiple constraints:

```
x, y : BOOLEAN;
t, u : BOOLEAN;
ASSERT (t <=> (x => y));
ASSERT (u <=> (y => x));
ASSERT (t => u);
ASSERT (x OR y);
ASSERT (NOT (y));
```

Either way, the result is satisfiable, and STP can find a satisfying assignment for you.

SAT solvers make solving many logic puzzles so easy that it takes all the fun out of it. Here's one from Smullyan. You are visiting an obscure island, where every inhabitant is either a knight or a knave. Knights always tell the truth. Knaves always lie (they only tell falsehoods). Some of the inhabitants are werewolves and have the annoying habit of sometimes turning into wolves at night and devouring unlucky tourists. A werewolf can be either a knight or a knave. You run into three inhabitants, Alice, Bob, and Carol. Alice declares, "I am a werewolf." Bob states "I am a werewolf." Carol says "At most one of us is a knight." Is it possible that Alice is a werewolf?

Solve this by expressing it as a satisfiability problem and then using STP to solve the satisfiability problem. You might want to introduce a boolean variable or two for each person indicating that person's status, and then encode their statements as assertions. Your homework solution should include three parts: (1) The final answer (could Alice be a werewolf?); (2) A print-out of the STP input file you use as well as the output from STP; and, (3) An explanation of the boolean variables you used.

Revised May 4 to fix a typo in the boolean formula immediately following "Here is another example."

3. (40 pts.) Sudoku

You may have heard of the game of Sudoku. Now you get a chance to write a computer program to solve it.

Sudoku is a game with a 9×9 grid, broken down into a 3×3 arrangement of blocks, where each block contains 3×3 cells. Each block contains a permutation of the numbers 1,2,...,9 in some order (no number is repeated). Also, each row and each column contains a permutation of those numbers. Some of the cells are filled in for you, and your task is to fill in the rest. You can find more information on Sudoku on the web; e.g., <http://www.sudoku.com/howto play.html>.

Your job is to solve the following Sudoku puzzle, using a SAT solver. (You wouldn't want to solve it by hand.)

					3		8	5
		1		2				
			5		7			
		4				1		
	9							
5							7	3
		2		1				
				4				9

In particular, I want you to generate an input to STP, so that the output of STP tells you how to solve the Sudoku puzzle. I suggest that you think about how to come up with a bunch of boolean variables for each cell, where the values of the boolean variables associated with that cell tells you what number should go in that cell. Then, express the constraints that a valid solution to the Sudoku puzzle must satisfy as boolean formulas in terms of these boolean variables, and feed them into STP. You might want to write a program to generate the input to STP.

Your homework answer should include: (1) the main idea of your approach, (2) a description of the boolean variables you used, (3) a print-out of the program you wrote to generate the input to STP, and (4) the solution to the above Sudoku puzzle (generated using your program).