# Homework #1

CS 474: Spring 2023

Due on Friday, Feb 17, 2023 11:59 PM Central

**Homework Policy:** You are allowed to collaborate with your classmates, but report this in your submission. Even if you work on problems together, each student must write up their solution individually in their own words in their submission. The CS 473 course's page on academic integrity is a handy reference: https://courses.engr.illinois.edu/cs473/sp2023/integrity.html. Late submissions are not allowed. If you have personal circumstances that will prevent you from submitting the assignment on time, write to the course staff as soon as possible and we can try to work with you.

**Points: Problem 1: 20 points, Problem 2: 20 points, Problem 3: 20 points; Total: 60 points**

**Problem 1.** Recall the Tseitin transformation that we learned in class. We recursively traverse a propositional formula bottom-up and convert it to CNF, creating an equi-satisfiable formula.

Let us do a similar translation, formally. We assume, without loss of generality, that formulas feature negation only for propositions, i.e., we push negation inward using De Morgan's laws until the only subformula of the form $\neg\psi$ is of the form $\neg p$ for a proposition $p$. This is called Negation Normal Form (NNF).

Given a formula $\alpha$ in NNF, we introduce new propositional variables $q_\psi$ for every subformula $\psi$ of $\alpha$. We then define the following transformation $\mathit{Circuit}(\alpha)$:

$$\begin{aligned}
\mathit{Circuit}(p) &:= q_p \Leftrightarrow p && \text{(for a proposition p)}\\
\mathit{Circuit}(\neg p) &:= q_{\neg p} \Leftrightarrow \neg p && \text{(for a proposition p)}\\
\mathit{Circuit}(\alpha \wedge \beta) &:= (q_{\alpha\wedge\beta} \Leftrightarrow (q_\alpha \wedge q_\beta)) \wedge \mathit{Circuit}(\alpha) \wedge \mathit{Circuit}(\beta)\\
\mathit{Circuit}(\alpha \vee \beta) &:= (q_{\alpha\vee\beta} \Leftrightarrow (q_\alpha \vee q_\beta)) \wedge \mathit{Circuit}(\alpha) \wedge \mathit{Circuit}(\beta)
\end{aligned}$$

**Aside:** To complete the transformation to CNF, we would then rewrite the $\Leftrightarrow$ formulas in the above definition to equivalent CNF formulas. More precisely,

- $p \Leftrightarrow (r \wedge s)$ can be written as $(\neg p \vee r) \wedge (\neg p \vee s) \wedge (\neg r \vee \neg s \vee p)$
- $p \Leftrightarrow (r \vee s)$ can be written as $(\neg p \vee r \vee s) \wedge (\neg r \vee p) \wedge (\neg s \vee p)$

**End Aside.**

Finally, the Tseitin transformation of $\alpha$ is written as $q_\alpha \wedge \mathit{Circuit}(\alpha)$.

In this problem, you will be working solely with the above definition of *Circuit*, rather than the completed CNF transformation. The objective is to prove that the transformation described by *Circuit* yields equi-satisfiable formulas.

More formally, prove that for every propositional formula $\alpha$, the formula $q_\alpha \wedge \mathit{Circuit}(\alpha)$ is satisfiable if and only if $\alpha$ is satisfiable.

*Hint:* Use structural induction in your solution. The induction hypothesis is as follows: For a propositional formula $\alpha$, given a valuation $v$ over the propositions occurring in $\alpha$ such that $v$

satisfies $\alpha$, there exists an extension $v'$ of $v$ such that $v'$ satisfies $q_\alpha$ and $\mathit{Circuit}(\alpha)$. Conversely, if a valuation $v$ satisfies $q_\alpha$ and $\mathit{Circuit}(\alpha)$, then $v$ satisfies $\alpha$.
Note that a valuation is a map from propositional variables to values. A valuation $v'$ is an extension of $v$ if the domain of $v'$ is a superset of $v$, and furthermore, $v$ and $v'$ give the same values to all propositions in the domain of $v$.

**Problem 2.** The compactness theorem for propositional logic is an elegant result that appears in many forms in mathematical and logical literature. The following word problem can be solved using the compactness theorem. The purpose of this problem is to build an intuition about compactness and enable you to identify applications of the theorem in your own research.

Dumbledore, the headmaster of Hogwarts, has a new creative project. At Hogwarts school, students can be sorted into any of four houses: *Gryffindor*, *Hufflepuff*, *Ravenclaw*, and *Slytherin*. He now wants to see if he can sort (countably) infinitely many students into the four houses.

However, he wants to avoid putting any two friends in the same house because when students are chatting with their friends, nobody is laughing at his jokes, all of which start with 'When I was a young boy some 150 years ago...'. He also knows that each student has at most three friends.

Dumbledore is an expert only in magic, not in logic. Therefore, he comes to you for help. Can you tell him whether his project is feasible? Is it possible to sort a countably infinite number of students into four houses with the above constraints? Provide a formal proof to support your answer.

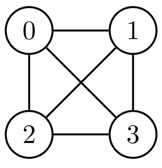
*Hints:* (1) You can assume that finite undirected graphs whose vertices have degree $\leq 3$ are four-colorable. (2) Assume that the set of students is a countable set $S = \{s_1, s_2, \ldots\}$ and you are given a set $F \subseteq S \times S$ that tells you pairs of students who are friends. Friendship is symmetric and nobody is their own friend. (3) To apply compactness, you will have to introduce propositions $p_{i,h}$ that represent whether student $s_i$ is in house $h$ and write the set of formulas whose compactness you want to establish using these propositions.

**Problem 3.** A SAT solver is a program that determines whether a propositional formula is satisfiable. Recall that a propositional formula is satisfiable if there is some assignment of truth values to propositions that makes the formula true. SAT solvers also typically provide concrete assignments that make the formula(s) true, which can tell us a good deal of information. You would have seen this in Problem 2 when you created propositions (whose meaning was in your mind) and used them to represent complex questions as a set of propositional formulas.

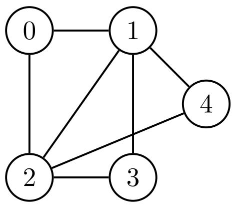
Many problems can be solved in practice by leveraging SAT solvers. For instance, emerging research in AI uses symbolic reasoning based on SAT solving to create systems whose capabilities go beyond the reach of traditional deep learning. Algorithms for effective, practical SAT solving are one of the amazing contributions of modern computer science.

The most common use of a SAT solver is to solve a problem by encoding it as a satisfiability query. More concretely, one solves an instance of a problem by encoding the instance as a query to a SAT solver. The result of the solver (satisfiability check and satisfying values of propositions if applicable) is translated back into the solution for the given problem instance.

In this assignment, you encode the three-coloring problem for two graphs as SAT queries. The query must be such that the formula is satisfiable if and only if the given graph is three-colorable. The following graphs $G$ and $H$ are the graphs you will need to work with:



$G$



$H$

**Questions:**

(a) Model the problem above by writing down a *mathematically* precise formula for a graph that is satisfiable if and only if the graph is three-colorable. Explain your formulation at a high level. This should be in (mathematical) readable form and succinct. Do not paste a Z3 file in your submission!

(b) Model the above formula using a SAT solver, following the instructions (and hints if you wish) below. Upload the files to GitHub publicly and provide a link to the files in your submission. Your files should have the names '<netid>_hw1_g.smt2' and '<netid>_hw1_h.smt2' for the encodings corresponding to $G$ and $H$ respectively. Run a SAT solver on your files and report the solver's solution.

*Bonus:* You can get bonus points if you can extract satisfying valuations and report what it means in terms of the graph-coloring problem. You may also be given bonus points if you write a program that takes an arbitrary graph and determines, using a SAT solver, whether it is three-colorable. If you write a program, submit it, in the same manner, using a publicly accessible link and provide the link here. Provide documentation to run your code. You *must* provide links to SAT files as required by the problem even if you write a program. Submission of a program is purely for bonus points.

*SAT Solving.* We will use Z3 in this course. It is a state-of-the-art SMT (Satisfiability Modulo Theories) solver that can check propositional satisfiability effectively, as well as many other capabilities that we will encounter during this course. You can find a tutorial here: https://microsoft.github.io/z3guide/docs/logic/intro. The section on 'Propositional Logic' in the linked tutorial should be sufficient for this assignment. The solver must output `sat` if the given graph is three-colorable, and `unsat` if it is not.

*Hints:* (1) You will want to define propositions $p_{i,c}$ that represent whether node $i$ has color $c$, and formulate constraints of adjacent nodes not having the same color using these propositions. (2) If you are not used to modeling problems in SAT/SMT, and if you are not used to mathematical encodings in general, we suggest working out the solution using Z3 first so you know that you are right before you write up your high-level formulation. (3) For those not familiar with SAT/SMT solvers, the following template may prove useful:

```
; Lines beginning with semicolons are comments.
; Fix colors 0, 1, 2
; The following line declares a propositional atom
; representing whether node 0 has the color 0
(declare-const p00 Bool)
; Similarly node 1 having color 2
(declare-const p12 Bool)
; More nodes and colors. Copy-paste is your friend.
...

; Write constraints below representing
; what a three-coloring of the given graph is
; in terms of the above propositions
...

; The following line is mandatory. It tells the solver
; to check the satisfiability
; of the above constraints.
(check-sat)
```