Homework Assignment 1

Bailey Wickham & Alex MacLean
CSC530

May 23, 2021

1 Propositional Logic and Normal Forms

1. (5 points) Use the solution skeleton in `classify.rkt`, write a Rosette procedure that takes as input a formula F in propositional logic and outputs

   - `'TAUTOLOGY` if $I \models F$ for every interpretation I;
   - `'CONTRADICTION` if $I \not\models F$ for every interpretation I; and,
   - `'CONTINGENCY` if there are two interpretations I and I' such that $I \models F$ and $I' \not\models F$.

   Your procedure may contain at most two solver-aided queries (such as `solve`), and if it contains more than one query, then the two queries must be different (i.e., you cannot use `solve` twice).

   See `classify.rkt`

2. (5 points) Convert the following formula to an equisatisfiable one in CNF using Tseitin's encoding:

$$\neg(\neg r \to \neg(p \wedge q))$$

   Write the final CNF as the answer. Use a_ϕ to denote the auxiliary variable for the formula ϕ; for example, $a_{p\wedge q}$ should be used to denote the auxiliary variable for $p \wedge q$. Your conversion should not introduce auxiliary variables for negations.

$$\begin{array}{ll} a_\phi & \wedge \\ (a_\phi \leftrightarrow \neg(\neg r \to \neg a_{p\wedge q})) & \wedge \\ (a_{p\wedge q} \leftrightarrow (p \wedge q)) & \end{array}$$

$$(a_\phi \leftrightarrow \neg(\neg r \to \neg a_{p\wedge q})) \equiv (\neg a_\phi \vee \neg r) \wedge (\neg a_\phi \vee a_{p\wedge q}) \wedge (r \vee \neg a_{p\wedge q} \vee a_\phi)$$
$$(a_{p\wedge q} \leftrightarrow (p \wedge q)) \equiv (\neg a_{p\wedge q} \vee p) \wedge (\neg a_{p\wedge q} \vee q) \wedge (\neg p \vee \neg q \vee a_{p\wedge q})$$

$$a_\phi \wedge (\neg a_\phi \vee \neg r) \wedge (\neg a_\phi \vee a_{p\wedge q}) \wedge (r \vee \neg a_{p\wedge q} \vee a_\phi) \wedge (\neg a_{p\wedge q} \vee p) \wedge (\neg a_{p\wedge q} \vee q) \wedge (\neg p \vee \neg q \vee a_{p\wedge q})$$

3. (10 points) Let ϕ be a propositional formula in NNF, and let I be an interpretation of ϕ. Let the *positive set* of I with respect to ϕ, denoted $pos(I, \phi)$, be the literals of ϕ that are satisfied by I. As an example, for the NNF formula $\phi = (\neg r \wedge p) \vee q$ and the interpretation $I = [r \mapsto \bot, p \mapsto \top, q \mapsto \bot]$, we have $pos(I, \phi) = \{\neg r, p\}$. Prove the following theorem about the monotonicity of NNF:

   Monotonicity of NNF: For every interpretation I and I' such that $pos(I, \phi) \subseteq pos(I', \phi)$, if $I \models \phi$, then $I' \models \phi$.

   (**Hint:** Use structural induction.)

Proof. Proceed by structural induction.

Base case: Suppose that ϕ is of the form: p or $\neg p$ and $I \models \phi$. Then $pos(I, \phi)$ must contain p or $\neg p$ respectively. Since $pos(I, \phi) \subseteq pos(I', \phi)$ then that element must also be in $pos(I', \phi)$. Therefore $I' \models \phi$.

Inductive hypothesis: Suppose there exists a ϕ_1 and ϕ_2 such that for every interpretation I and I' such that $pos(I, \phi_i) \subseteq pos(I', \phi_i)$, if $I \models \phi_i$, then $I' \models \phi_i$.

Inductive step: Let $\phi = \phi_1 \wedge \phi_2$ and I satisfy ϕ. Let $pos(I, \phi) \subseteq pos(I', \phi)$ for some interpretation I'. Then $I \models \phi_1$ and $pos(I, \phi_1) \subseteq pos(I, \phi) \subseteq pos(I', \phi)$ so by the inductive hypothesis $I' \models \phi_1$. A similar argument can be applied to ϕ_2. Since $I' \models \phi_1$ and $I' \models \phi_2$, $I' \models \phi$. The case for $\phi = \phi_1 \vee \phi_2$ follows similarly.

Conclusion: By induction, every interpretation I and I' such that $pos(I, \phi) \subseteq pos(I', \phi)$, if $I \models \phi$, then $I' \models \phi$.

$\square$

4. (10 points) Let ϕ be an NNF formula. Let $\hat{\phi}$ be a formula derived from ϕ using a modified version of Tseitin's encoding in which the CNF constraints are derived from implications rather than bi-implications. For example, given the formula

$$a_1 \wedge (a_2 \vee \neg a_3),$$

the new encoding is the CNF equivalent of the following, where x_0, x_1, x_2 are fresh auxiliary variables:

$$\begin{array}{ll} x_0 & \wedge \\ (x_0 \rightarrow a_1 \wedge x_1) & \wedge \\ (x_1 \rightarrow a_2 \vee x_2) & \wedge \\ (x_2 \rightarrow \neg a_3) & \end{array}$$

Note that Tseitin's encoding to CNF starts with the same formula, except that $\rightarrow$ is replaced with $\leftrightarrow$. As a result, the new encoding has roughly half as many clauses as the Tseitin's encoding.

Prove that $\hat{\phi}$ is satisfiable if and only if ϕ is satisfiable.

(**Hint**: Use the theorem from Problem 3.)

2 Graph Coloring with SAT (40 points)

A graph is *k-colorable* if there is an assignment of k colors to its vertices such that no two adjacent vertices have the same color. Deciding if such a coloring exists is a classic NP-complete problem with many practical applications, such as register allocation in compilers. In this problem, you will develop a CNF encoding for graph coloring and apply them to graphs from various application domains, including course scheduling, N-queens puzzles, and register allocation for real code.

A finite graph $G = \langle V, E \rangle$ consists of vertices $V = \{v_1, \ldots, v_n\}$ and edges $E = \{\langle v_{i_1}, w_{i_1} \rangle, \ldots, \langle v_{i_m}, w_{i_m} \rangle\}$. Given a set of k colors $C = \{c_1, \ldots, c_k\}$, the *k-coloring* problem for G is to assign a color $c \in C$ to each vertex $v \in V$ such that for every edge $\langle v, w \rangle \in E$, $\mathsf{color}(v) \neq \mathsf{color}(w)$.

6. (10 points) Show how to encode an instance of a k-coloring problem into a propositional formula F that is satisfiable iff a k-coloring exists.

(a) Describe a set of propositional constraints asserting that every vertex is colored. Use the notation $\mathsf{color}(v) = c$ to indicate that a vertex v has the color c. Such an assertion is encodable as a single propositional variable p_v^c (since the set of vertices and colors are both finite).

$$\bigwedge_{v \in V} (\bigvee_{c \in C} p_v^c)$$

(b) Describe a set of propositional constraints asserting that every vertex has at most one color.

$$\bigwedge_{v \in V} \bigwedge_{c \in C} (p_v^c \rightarrow \neg(\bigvee_{c' \in C - \{c\}} p_v^{c'}))$$

(c) Describe a set of propositional constraints asserting that no two adjacent vertices have the same color.

$$\bigwedge_{\langle v,w \rangle \in E} \bigwedge_{c \in C} (\neg p_v^c \vee \neg p_w^c)$$

(d) Identify a significant optimization in this encoding that reduces its size asymptotically. (**Hint:** Can any constraints be dropped? Why?)

The constraint ensuring that each vertex has only 1 color (b) can be dropped. If the SAT solver assigns a vertex multiple colors this just means that any of them can be picked for a valid coloring.

(e) Specify your constraints in CNF. For $|V|$ vertices, $|E|$ edges, and k colors, how many variables and clauses does your encoding require?

$$\bigwedge_{\langle v,w \rangle \in E} \bigwedge_{c \in C} (\neg p_v^c \vee \neg p_w^c) \wedge \bigwedge_{v \in V} (\bigvee_{c \in C} p_v^c)$$

Clauses: $k(|V| + |E|)$; *Variables:* $k|V|$

7. (20 points) Implement the above encoding in Racket, using the provided solution skeleton. See the `README` file for instructions on obtaining solvers and the database of graph coloring problems. Your program should generate the encoding for a given graph (see `graph.rkt`), call a SAT solver on it (`solver.rkt`), and then decode the result into an assignment of colors to vertices (see `examples.rkt` and `k-coloring.rkt`).

   Your implementation should be able to solve all of the easy and medium instances in under 15 minutes on an ordinary laptop. (The reference implementation does so in about 7 minutes.)

   See `k-coloring.rkt`

8. (5 points) Describe a CNF encoding for k-coloring that uses $O(|V| \log k + |E| \log k)$ variables and clauses.

9. (5 points) Most modern SAT solvers support *incremental solving*—that is, obtaining a solution to a CNF, adding more constraints, obtaining another solution, and so on. Because the solver keeps (some) learned clauses between invocations, incremental solving is generally the fastest way to solve a series of related CNFs. How would you apply incremental solving to your encoding from Problem 7 to find the smallest number of colors needed to color a graph (i.e., its chromatic number)?

3 Optimal Graph Coloring with Variations on SAT (10 points)

Consider the following variations on the propositional satisfiability (SAT) problem discussed in Lecture 5:

Partial Weighted MaxSAT Given a CNF formula $\phi_H = \bigwedge_{c \in H} c$ corresponding to a set of *hard* clauses H, and a CNF formula $\phi_S = \bigwedge_{c \in S} c$ corresponding to a set of *soft* CNF clauses S with weights $w : S \to \mathbb{Z}^+$, the Partial Weighted MaxSAT problem is to find an assignment A to the problem variables that satisfies all the hard clauses and that maximizes the weight of the satisfied soft clauses. That is, $A \models \bigwedge_{c \in H} c$, and if we let $C = \{c \in S | A \models c\}$, then there is no $C' \subseteq S$ such that $H \cup C'$ is satisfiable and $\sum_{c' \in C'} w(c') > \sum_{c \in C} w(c)$.

Pseudo-Boolean Optimization Let B be a set of *pseudo-boolean constraints* of the form $\sum a_{ij} x_j \geq b_i$, where x_j is a variable over $\{0, 1\}$ and a_{ij}, b_i, c_j are integer constants. The Pseudo-Boolean Optimization problem is to satisfy all constraints in B while minimizing a linear function $\sum c_j \cdot x_j$.

Let $G = \langle V, E \rangle$ be a finite graph and $C_k = \{c_1, \ldots, c_k\}$ a set of k colors. Let $P(G, C_k)$ be the CNF formula produced by applying your encoding from Problems 6-7 to the graph G and the coloring C_k. As before, we use p_v^c to denote the propositional variable indicating that the vertex $v \in V$ has the color $c \in C_k$.

10. (5 points) Explain how to create a Partial Weighted MaxSAT instance $P_{\text{opt}}(G)$ such that every solution to $P_{\text{opt}}(G)$ represents a valid χ-coloring of G where χ is the chromatic number of G (i.e., the smallest possible number of colors needed to color G).

    Your encoding of $P_{\text{opt}}(G)$ may use $P(G, C_k)$ for at most one k of your choosing. So, $P_{\text{opt}}(G)$ cannot use, for example, both $P(G, C_1)$ and $P(G, C_2)$.

    Write down $P_{\text{opt}}(G)$ by specifying the set H of hard clauses, the set S of soft clauses, and the function $w : S \to \mathbb{Z}^+$ that assigns a positive weight to each soft clause in S.

$$H = \bigwedge \ldots$$
$$S = \bigwedge \ldots$$
$$w(s) = \ldots \text{ for each clause } s \in S$$

11. (5 points) Explain how to create a Pseudo-Boolean Optimization instance $P_{\text{opt}}(G)$ such that every solution to $P_{\text{opt}}(G)$ represents a valid χ-coloring of G where χ is the chromatic number of G (i.e., the smallest possible number of colors needed to color G).

    To create $P_{\text{opt}}(G)$, observe that every CNF instance can be transformed into a set of equivalent pseudo-boolean constraints. To apply this observation, explain how to do the transformation.

    As before, your encoding of $P_{\text{opt}}(G)$ may use the pseudo-boolean equivalent of $P(G, C_k)$ for at most one k of your choosing.

    Write down $P_{\text{opt}}(G)$ by specifying the pseudo-boolean constraints to solve and the linear function to minimize:

$$\begin{aligned} \text{minimize} \quad & \sum \ldots \\ \text{subject to} \quad & \bigwedge \ldots \end{aligned}$$