Assignment 2

Due: Oct 31, 2024 at 23:59

Total points:	20
Deliverables:	Your implementation of a Sudoku solver (see the README file inside sudoku/) `hw2.pdf` - your solutions to the following problems.
Instructions:	*Your solutions for this file can be either typeset or handwritten.* Please clearly separate your solutions to each of the questions in your document.

1 Propositional Logic and Normal Forms (3 points)

1. (3 points) Convert the following formula to an equisatisfiable one in CNF using Tseytin transformation:

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

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.

2 SAT solving (5 points)

2. (5 points) In this problem, you will manually craft the execution of a SAT solver on a sample CNF, and use this trace to reconstruct the implication graphs of the underlying CDCL algorithm.

   Consider the following CNF:

   $$(\neg x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee x_3) \wedge (\neg x_4 \vee \neg x_2) \wedge (\neg x_4 \vee \neg x_1 \vee x_2) \wedge (x_3 \vee \neg x_1) \wedge (\neg x_3 \vee \neg x_2 \vee x_4) \wedge (x_1 \vee x_4) \wedge (\neg x_2 \vee x_1)$$

   Reconstruct the behavior of the CDCL algorithm by filling out the following abstract trace template, given as a list of abstract trace entries:

   - **Level** i *; decision level i*
     - **Decision:** d_i *; decision literal at level i or NA if level due to backtracking*
     - **BCP:** $p_{i_0}, \ldots, p_{i_n}$ *; literals inferred by BCP at level i, in the detailed trace order*
     - **Conflict Clause:** $l_{i_0} \ldots l_{i_k}$ *; conflict clause or NA if no conflict at level i*
     - **Implication Graph:** graph image *; implication graph at level i*
   - ...

   To produce the abstract trace, create an abstract trace entry ("Level") whenever the decision level changes in the detailed trace due to a new decision or backtracking.

   Your abstract trace must make decisions based on the order x_1, x_2, x_3, x_4. When a decision is about to make on a variable x_i, count the number of times p_{x_i} that x_i appears positively in the formula and the number of times n_{x_i} that x_i appears negatively in the formula. If $p_{x_i} > n_{x_i}$, decide x_i False or 0. If $p_{x_i} \leq n_{x_i}$, decide x_i True or 1.

3 Graph Coloring with SAT (5 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 $\langle v, w \rangle \in E$, $\mathsf{color}(v) \neq \mathsf{color}(w)$.

3. (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 p_v^c to indicate that a vertex v has the color c. Such an assertion is encodable as a single propositional variable (since the set of vertices and colors are both finite).

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

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

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

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

4 SMT Solving (7 points)

4. (2 points) Apply the congruence closure algorithm to decide the satisfiability of the following $T_=$ formula:

$$f(g(x)) = g(f(x)) \wedge f(g(f(y))) = x \wedge f(y) = x \wedge g(f(x)) \neq x$$

Provide the level of detail as in the lecture. In particular, show the intermediate partitions (sets of congruence classes) after each merge or propagation step, together with a brief explanation of how the algorithm arrived at that partition (e.g., "by literal $f(x) = y$, merge $f(x)$ with y").

5. (5 points) Consider the following formula in $T_= \cup T_R$:

$$g(x + y, z) = f(g(x, y)) \wedge x + z = y \wedge z \geq 0 \wedge x \geq y \wedge g(x, x) = z \wedge f(z) \neq g(2x, 0)$$

   (a) (2 points) Purify the formula and show the resulting $T_=$ and T_R formulas. Show the purification results using the table below. Apply purification to the (current) innermost term first. If there are several innermost terms, prefer the leftmost one. Use a_i to refer to the i^{th} auxiliary literal, starting with a_1. All occurrences of the same term should be mapped to the same auxiliary literal. You do not need to show the individual steps of the purification process, just the final result.

$T_=$	T_R
...	...

   (b) (3 points) Use the Nelson-Oppen procedure to decide the satisfiability of the purified formula. In one sentence, state which version of the procedure you are using and why. Show the equality propagation by filling out the table below. If T_i infers the j^{th} equality (or disjunction of equalities), enter it into the j^{th} row and i^{th} column only—leave the remaining column in that row empty.

$T_=$	T_R
...	...