CSC410

Exam 2

Date: October 23, 2026
weight: 17% of course mark, length: 60 minutes

First Name: ———————————————

Last Name: ———————————————

Student Number: ———————————————

Read Before You Start

- Write your name and student number, and please do it so that it matches your record on Quercus.
- This exam contains **3 problems** (some with multiple subproblems).
- Problem 3 has an additional part (d) for graduate students only. Undergraduate students need not answer this part.
- The exam totals 35 points for undergraduate students and 40 points for graduate students.
- The exam is designed so that the answers require **no English** words. You are welcome to write English sentences, but be warned that they will not be graded. Only the formal components required for each answer are graded.
- The exam may look long, but most questions have very short answers. The exception is Problem 3(c), where you need to write a few short steps of an algorithm. The generous 60-minute time is there to remove any stress from this process. It is not an indication that this exam requires 1 hour of work!

Problem 1: Binary Resolution (10 points)

Consider the propositional formula F given below (in CNF). Prove F is unsatisfiable using *binary resolution*.

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

Answer. (a) Here is a 4-step resolution proof:

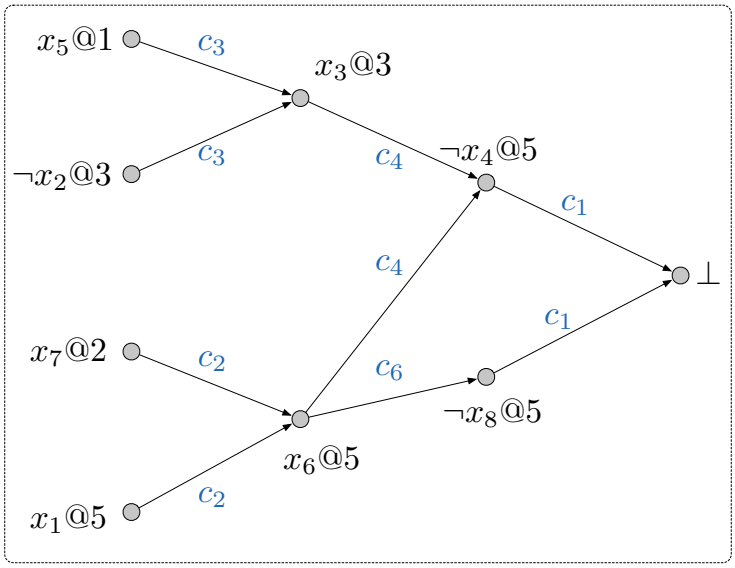
$$\dfrac{\dfrac{\dfrac{x_3 \vee x_4 \qquad \dfrac{x_1 \vee x_4 \qquad \neg x_1 \vee \neg x_3 \vee x_4}{\neg x_3 \vee x_4}}{x_4} \qquad \dfrac{\neg x_2 \vee \neg x_4 \qquad x_2 \vee \neg x_4}{\neg x_4}}{\bot}$$

Problem 2: DPLL Conflict Clause Learning (10 points)

Consider a formula F that is being checked for satisfiability in CNF using the following set of clauses:

$$c_1 = x_4 \vee x_8 \qquad c_2 = \neg x_1 \vee x_6 \vee \neg x_7 \qquad c_3 = x_2 \vee x_3 \vee \neg x_5$$
$$c_4 = \neg x_3 \vee \neg x_4 \vee \neg x_6 \qquad c_5 = \neg x_2 \vee \neg x_3 \vee \neg x_5 \qquad c_6 = \neg x_6 \vee \neg x_8$$

Consider a hypothetical snapshot of the DPLL algorithm with the following implication graph from decisions previously made at levels 1,2, and 3 of the DPLL algorithm, and the current decision node at level 5. As you can see the implication graph is complete, and a conflict has already been found at level 5.



Produce **all** *asserting* clauses that could be learned by the DPLL algorithm before it backtracks. In the case of each clause, indicate the level to which the algorithm would backtrack after learning the clause. To justify your answer, write down the sequence of binary resolution steps (and all the intermediate clauses) that gets you to your asserting clause. For clarity, label each clause that you use with its identifier from the list, e.g. $c_5 : \neg x_2 \vee \neg x_3 \vee \neg x_5$ instead of just $\neg x_2 \vee \neg x_3 \vee \neg x_5$.

Note: A part of the graph would have existed at an earlier level. That is alright. It is not a mistake or an oversight. You just answer the questions about how the algorithm behaves at level 5.

Answer.

$$\frac{c_6 : \neg x_6 \vee \underline{\neg x_8} \qquad c_1 : x_4 \vee \underline{x_8}}{x_4 \vee \neg x_6}$$

$$\frac{\underline{x_4} \vee \neg x_6 \qquad c_4 : \neg x_3 \vee \underline{\neg x_4} \vee \neg x_6}{\neg x_3 \vee \neg x_6}$$

$\neg x_3 \vee \neg x_6$ is the first asserting clause. It corresponds to the UIP $x_6@5$, which is on every path from $x_1@5$ to the conflict node. If we backtrack to level 3 with this clause, it will become a unit clause resulting to a decision $\neg x_6@3$ at that level.

If we continue:

$$\frac{\neg x_3 \vee \underline{\neg x_6} \qquad c_2 : \neg x_1 \vee \underline{x_6} \vee \neg x_7}{\neg x_1 \vee \neg x_3 \vee \neg x_7}$$

$\neg x_1 \vee \neg x_3 \vee \neg x_7$ is another asserting clause that the algorithm can learn, and it will backtrack to level 3 after learning it. This one corresponds to the trivial UIP $x_1@5$. This clause will become unit at level 3, and result in a decision $\neg x_1@3$.

For the full mark, it suffices to include the 3 resolution steps, correctly identify the two asserting clauses and the corresponding backtracking level. The English explanation in this answer is for your benefit.

Problem 3: Symbolic Reachability (15 points)

There are $n \geq 4$ Boolean switches $x_1, \ldots, x_n$, which initially look like this:

$$\underbrace{00100\cdots0}_{n}$$

with precisely one "1" at position three. Each move in the system chooses an adjacent pair of bits that is 10 and flips both bits (i.e. $10 \longrightarrow 01$), leaving all other bits unchanged. So, for instance the first move from the initial state would be:

$$\underbrace{00100\cdots0}_{n} \longrightarrow \underbrace{00010\cdots0}_{n}$$

The error state is

$$E = \underbrace{0100\cdots0}_{n}$$

Answer each part below independently.

(a) (5 points) Give a (propositional logic) formula describing precisely the set of **all reachable states**.

Answer.

$$R(x) = \neg x_1 \wedge \neg x_2 \wedge \left(\bigvee_{i=3}^{n} x_i\right) \wedge \bigwedge_{3 \leq i < j \leq n} (\neg x_i \vee \neg x_j).$$

(b) (4 points) Give the **weakest inductive invariant** that contains the initial state and excludes E; i.e. the inductive invariant admitting the largest possible set of states.

Answer.

$$J(x) = \left(\neg x_1 \vee x_2 \vee \bigvee_{i=3}^{n} x_i\right) \wedge \left(x_1 \vee \neg x_2 \vee \bigvee_{i=3}^{n} x_i\right).$$

Justification (not necessary for the grade, but added for your benefit):

We need to exclude (at a minimum) the following two states:

$$A = \underbrace{100\cdots0}_{n} \text{ and } B = \underbrace{010\cdots0}_{n}$$

One is the error state, and the other one can get to the error state with one transition. It turns out that just excluding the two does give us an inductive invariant, and no further effort is necessary. It also includes the initial state. So, the answer is:

$$\neg(x_1 \wedge \bigwedge_{i=2}^{n} \neg x_i) \wedge \neg(\neg x_1 \wedge x_2 \bigwedge_{i=3}^{n} \neg x_i)$$

which, by pushing the negation in and applying the Demorgan law becomes equivalent to the answer given above.

A tempting (almost correct) answer would have been:

$$\neg x_1 \wedge \neg x_2$$

It is inductive, and it rules out the error state. But, the one given above is strictly weaker than this. So, the correct answer to this problem comes from the methodical reasoning that follows the spirit of PDR rather than generic clever thinking that would produce the above, but is not completely right.

(c) (6 points) We want to do a partial PDR run for this item. We assume $n = 4$, and we are at a snapshot of the algorithm where we have 3 frames (including the initial frame) as listed below:

$$R_0 = \neg x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4, \qquad R_1 = x_1 \vee \neg x_2 \vee x_3 \vee x_4, \qquad R_2 = \top.$$

Recall that the error state is 0100 and as such the check $R_2 \wedge E$ is SAT, with the satisfying assignment $s = 0100$. Explain what PDR does step by step, until it is done with this round, and creates the next (empty) frame R_3. Do not forget about any clause pushing that may happen when the empty frame R_3 is created. You may use T as a shorthand for the transition relation formula without defining it.

You do not need to justify the SAT/UNSAT answers. Simply declare them. Hint: if you do this right, you should end up with the same exact answer as your part (b) specialized to $n = 4$.

Note: Below, you see an empty table that you can fill for your answer. It is given to you for pure convenience. If you do not like the table format, skip it, and do the answer in any format you like. You may use English text (and/or diagrams) if it helps you organize your answer. We will grade the essential checks and conclusions in each step, and the correct progression of the algorithm, not the explanatory text. There is an extra empty page at the end of the exam that you may use for your solution.

Answer.

Condensed answer is below.

#	query	result	Algorithm's Action	R_1	R_2
1	$R_1 \wedge T \wedge s'$	**SAT**	satisfying assignment: $t = 1000$	$x_1 \vee \neg x_2 \vee x_3 \vee x_4$	$\top$
2	$R_0 \wedge T \wedge t'$	**UNSAT**	add clause excluding 1000 to R_1	$x_1 \vee \neg x_2 \vee x_3 \vee x_4$, $\neg x_1 \vee x_2 \vee x_3 \vee x_4$	$\top$
3	$R_2 \wedge E$	**SAT**	start over: $s = 0100$	$x_1 \vee \neg x_2 \vee x_3 \vee x_4$, $\neg x_1 \vee x_2 \vee x_3 \vee x_4$	$\top$
4	$R_1 \wedge T \wedge s'$	**UNSAT**	add clause excluding 0100 to R_2	$x_1 \vee \neg x_2 \vee x_3 \vee x_4$, $\neg x_1 \vee x_2 \vee x_3 \vee x_4$	$x_1 \vee \neg x_2 \vee x_3 \vee x_4$
5	$R_2 \wedge E$	**UNSAT**	New empty frame R_3	$x_1 \vee \neg x_2 \vee x_3 \vee x_4$, $\neg x_1 \vee x_2 \vee x_3 \vee x_4$	$x_1 \vee \neg x_2 \vee x_3 \vee x_4$
6	push $\neg x_1 \vee x_2 \vee x_3 \vee x_4$ $R_1 \rightarrow R_2$? $R_1 \wedge T \wedge (1000)'$	**UNSAT**	push the clause	$x_1 \vee \neg x_2 \vee x_3 \vee x_4$, $\neg x_1 \vee x_2 \vee x_3 \vee x_4$	$x_1 \vee \neg x_2 \vee x_3 \vee x_4$, $\neg x_1 \vee x_2 \vee x_3 \vee x_4$
			$R_1 = R_2$: **terminate**		

For grading, we only care about what check is performed, and what step the algorithm follows as a result of the check. The entire algorithm is explained fully with below for your benefit.

(i) Looking for a predecessor of $s = 0100$. Check $R_1 \wedge T \wedge s$ is SAT?

$$(x_1 \vee \neg x_2 \vee x_3 \vee x_4) \wedge T \wedge \neg x_1' \wedge x_2' \wedge \neg x_3' \wedge \neg x_4'.$$

is **SAT**, with $t = \mathbf{1000}$.

(ii) Looking for a predecessor of t. *Step 2.* Check Check $R_0 \wedge T \wedge t$ is SAT?

$$\neg x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4 \wedge T \wedge x_1' \wedge \neg x_2' \wedge \neg x_3' \wedge \neg x_4'.$$

is **UNSAT**.

(iii) Since t is not reachable from R_0, add the clause excluding $t = 1000$ to R_1:

$$\neg x_1 \vee x_2 \vee x_3 \vee x_4 \quad \text{added to } \mathbf{R_1}.$$

(iv) Start over. $R_2 \wedge E$ is still SAT with $s = 0100$. Check $R_1 \wedge T \wedge s$ is SAT?

$$(x_1 \vee \neg x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee x_4) \wedge T \wedge \neg x_1' \wedge x_2' \wedge \neg x_3' \wedge \neg x_4'.$$

is **UNSAT**.

Since s is not reachable from R_1 in one step, add the clause excluding $s = 0100$ to R_2:

$$x_1 \vee \neg x_2 \vee x_3 \vee x_4 \quad \text{added to } \mathbf{R_2}.$$

Now $R_2 \wedge E$ is **UNSAT**, since E is exactly 0100. The frames are

$$R_1 = (x_1 \vee \neg x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee x_4),$$
$$R_2 = x_1 \vee \neg x_2 \vee x_3 \vee x_4.$$

(v) New emtpy frame R_3 The clause $x_1 \vee \neg x_2 \vee x_3 \vee x_4$ is already in R_2, so only the new clause needs a pushing query.

$$(x_1 \vee \neg x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee x_4) \wedge T \wedge x_1' \wedge \neg x_2' \wedge \neg x_3' \wedge \neg x_4'.$$

The query is **UNSAT**, so the clause is pushed:

$$R_2 = (x_1 \vee \neg x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee x_4).$$

Now $R_1 = R_2$: two equal frames, so the algorithm **terminates**. The common frame is an inductive invariant that excludes E; it is exactly J from (b).

Graduate Students Only: Continuation of Problem 3 (5 points)

(d) We want to describe an alternative set of error states for the bitvector transition system of Problem 3. A configuration of bits is an error state exactly when the 1 bits form **one nonempty contiguous block**. The block may have any positive length and may touch either end. For example, 0011100 belongs to the error states, while 0100100 and 0000000 do not.

Give a propositional formula $G(x_1, \ldots, x_n)$ encoding exactly the error states.

Answer.

$$G(x) = \left(\bigvee_{i=1}^{n} x_i \right) \wedge \bigwedge_{1 \le i < j < k \le n} \big((x_i \wedge x_k) \Rightarrow x_j \big).$$

Empty page if you wish to skip the table in Problem 3(c) and do your own structure for the solution.