Practice Problems for Pulse-Check 3

May 5, 2026

1. A formula is in **Conjunctive Normal Form (CNF)** if it is:

   (A) An OR of ANDs of literals
   (B) An AND of ORs of literals
   (C) An AND of implications
   (D) A single clause with no negations

2. In a CNF formula, a **clause** is satisfied if:

   (A) All literals in the clause are True
   (B) At least one literal in the clause is True
   (C) An even number of literals are True
   (D) The clause contains at most 3 literals

3. A CNF formula is satisfiable if:

   (A) At least one clause is satisfied
   (B) At least half the clauses are satisfied
   (C) There exists an assignment that satisfies ALL clauses simultaneously
   (D) Every assignment satisfies at least one clause

4. The implication $A \Rightarrow B$ is logically equivalent to:

   (A) $A \wedge B$
   (B) $\neg A \vee B$
   (C) $A \vee \neg B$
   (D) $\neg A \wedge B$

5. To encode "exactly one of $x_1, x_2, \ldots, x_n$ is True" in SAT, we need:

(A) At-least-one: `Or`$(x_1, \ldots, x_n)$ only

(B) At-most-one: $(\neg x_i \vee \neg x_j)$ for all pairs $i < j$, only

(C) Both at-least-one AND at-most-one constraints

(D) A single clause containing all variables

6. In the Sudoku encoding from lecture, how many **Boolean variables** are needed (one per cell-digit assignment for a 9×9 grid)?

   (A) 81

   (B) 162

   (C) 729

   (D) 6561

7. After Z3 returns `sat` and gives you a model, you want to find a *different* solution. You add a **blocking clause** that:

   (A) Removes all variables from the solver

   (B) Negates the conjunction of the current assignment (at least one variable must differ)

   (C) Adds a new variable for each existing variable

   (D) Resets the solver to its initial state

8. In DPLL, **unit propagation** fires when:

   (A) A clause has all literals assigned False

   (B) A clause has exactly one unassigned literal (all others are False)

   (C) A clause has more than 3 literals

   (D) A variable appears only positively in the formula

9. If a clause becomes empty (all literals are False) during DPLL, this means:

   (A) The formula is satisfiable

   (B) We found a solution

   (C) A conflict has occurred; we must backtrack

   (D) We should apply pure literal elimination

10. **Pure literal elimination** removes a variable x from the formula when:

    (A) x appears in every clause

(B) x appears only in one clause

(C) x appears only positively (or only negatively) across all remaining clauses

(D) x is the last unassigned variable

11. CDCL improves over DPLL by:

(A) Trying variables in a random order

(B) Using floating point arithmetic

(C) Learning new clauses from conflicts and backjumping non-chronologically

(D) Converting the problem to an LP relaxation

12. In CDCL, when a conflict is detected, the solver:

(A) Restarts from scratch

(B) Analyzes the implication graph to find a learned clause, then backjumps

(C) Removes the conflicting clause from the formula

(D) Doubles the number of variables

13. The VSIDS heuristic used in CDCL decides which variable to branch on by:

(A) Choosing the variable that appears in the most clauses

(B) Choosing the variable with the highest activity score (bumped during conflicts, decayed over time)

(C) Choosing variables in alphabetical order

(D) Choosing the variable with the smallest index

14. A modern CDCL solver can typically handle SAT instances with up to:

(A) 20 variables

(B) 1,000 variables

(C) 1,000,000+ variables (structured instances)

(D) Any number of variables in polynomial time

15. SMT stands for:

(A) Satisfiability Modulo Types

(B) Satisfiability Modulo Theories

(C) Symbolic Math Transformation

(D) Structural Model Testing

16. In the theory of **Linear Integer Arithmetic (LIA)**, which of the following is NOT a valid constraint?

    (A) $x + 2y \leq 10$

    (B) $3x - y = 7$

    (C) $x \cdot y \leq 5$

    (D) $x \geq 0$

17. The **congruence axiom** in EUF (Equality with Uninterpreted Functions) states:

    (A) $f(a) = f(b)$ always

    (B) If $a = b$, then $f(a) = f(b)$

    (C) If $f(a) = f(b)$, then $a = b$

    (D) f must be injective

18. In the Array theory, what does $\texttt{Select}(\texttt{Store}(a, i, v), i)$ equal?

    (A) $a[i]$ (the original value)

    (B) v (the stored value)

    (C) i (the index)

    (D) Undefined

19. In the **lazy** (DPLL(T)) approach to SMT, the SAT solver and theory solver interact by:

    (A) The SAT solver handles everything, ignoring theory

    (B) The theory solver converts the entire problem to pure boolean SAT first

    (C) The SAT solver proposes assignments; the theory solver checks consistency and returns conflict clauses if inconsistent

    (D) The theory solver runs first and passes the result to the SAT solver

20. In the **eager** (bit-blasting) approach to SMT with bitvectors:

    (A) All arithmetic is approximated by floating point

    (B) Integer variables are expanded into individual boolean bits and operations are encoded as boolean circuits

    (C) The solver uses branch and bound

    (D) Variables are treated as real numbers

21. For encoding "at most k of n Bools are True," the naive pairwise approach requires $\binom{n}{k+1}$ clauses. The Sequential Counter (Sinz 2005) encoding uses:

    (A) $O(n!)$ auxiliary variables

    (B) $O(nk)$ auxiliary variables and clauses

    (C) No auxiliary variables

    (D) $O(2^n)$ clauses

22. **Symmetry breaking** in SAT/SMT refers to:

    (A) Breaking the formula into smaller sub-problems

    (B) Adding constraints that eliminate equivalent solutions (e.g., fixing canonical orderings)

    (C) Removing all negative literals

    (D) Converting the problem from SAT to LP

23. To prove that two circuits C_1 and C_2 are equivalent for ALL inputs using Z3, you:

    (A) Check `sat` for $C_1(\vec{x}) = C_2(\vec{x})$ and verify the model

    (B) Check `sat` for $C_1(\vec{x}) \neq C_2(\vec{x})$; if `unsat`, they are equivalent

    (C) Enumerate all 2^n inputs and compare outputs

    (D) Check if both circuits have the same number of gates

24. The famous Java `Arrays.binarySearch` bug was caused by:

    (A) An off-by-one error in the loop bounds

    (B) Integer overflow in `(low + high) / 2` when `low + high > Integer.MAX_VALUE`

    (C) Using floating point instead of integers

    (D) Forgetting to sort the array first

25. Which Z3 sort (type) would you use to detect an integer overflow bug in 32-bit arithmetic?

    (A) `Bool`

    (B) `Int` (mathematical integers, unbounded)

    (C) `BitVec(32)` (fixed-width, wraps on overflow)

    (D) `Real`

Solutions:

1. (B)
2. (B)
3. (C)
4. (B)
5. (C)
6. (C)
7. (B)
8. (B)
9. (C)
10. (C)
11. (C)
12. (B)
13. (B)
14. (C)
15. (B)
16. (C)
17. (B)
18. (B)
19. (C)
20. (B)
21. (B)
22. (B)
23. (B)
24. (B)
25. (C)