Logic refresher and SAT

Propositional logic, first-order logic, and SAT

CMPUT 664 · Formal Verification

Today

1. Propositional logic: formulas and truth
2. First-order logic: objects, relations, and quantifiers
3. SAT solving: a high-level view

Propositional logic: building blocks

Variables: $p, q, r, \ldots$

Truth constants: $\top, \bot$

Connectives:

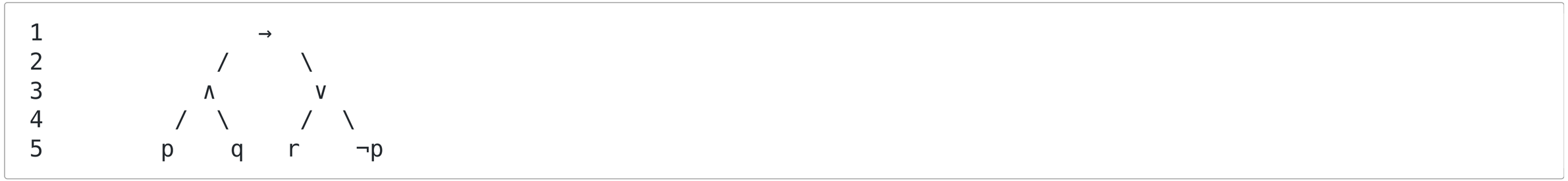
$$\neg F \qquad F_1 \wedge F_2 \qquad F_1 \vee F_2 \qquad F_1 \rightarrow F_2 \qquad F_1 \leftrightarrow F_2$$

A literal is a variable or its negation.

Read formulas structurally

$$(p \land q) \to (r \lor \neg p)$$

The outermost connective is $\to$.



An interpretation assigns truth values

For propositional variables $p_1, \ldots, p_n$:

$$I : \{p_1, \ldots, p_n\} \to \{\top, \bot\}$$

The connectives determine the value of every larger formula.

$$I \models F$$

means that F evaluates to true under I.

Implication is a conditional guarantee

$$p \to q \equiv \neg p \lor q$$

It is false in exactly one case:

p	q	$p \to q$
$\top$	$\bot$	$\bot$

The promise is broken only when p holds but q does not.

Evaluate this formula

$$F = (p \wedge q) \rightarrow (p \vee \neg q)$$

under

$$I = \{p \mapsto \top, q \mapsto \bot\}.$$

Does $I \models F$?

Models and countermodels

If $I \models F$, then I is a **model** of F.

If $I \not\models F$, then I is a **countermodel** of F.

For a claimed implication:

$$P \rightarrow Q$$

a countermodel makes P true and Q false.

Four possible outcomes

Outcomes	Meaning
Satisfiable	True under at least one interpretation
Unsatisfiable	True under no interpretation
Valid	True under every interpretation
Contingent	Satisfiable but not valid

These describe a formula across **all** interpretations.

Classify each formula

$$p \vee \neg p$$

$$p \wedge \neg p$$

$$p \rightarrow q$$

Valid, unsatisfiable, or contingent?

Syntax and semantics are different

Syntax: Which expressions are well formed?

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

Semantics: What does an expression mean under an interpretation?

$$I = \{p \mapsto \top, q \mapsto \bot, r \mapsto \top\}$$

Validity becomes satisfiability

$$F \text{ is valid} \quad \iff \quad \neg F \text{ is unsatisfiable}$$

To search for a flaw in a claim, ask for a model of its negation.

$$\text{prove } P \to Q \quad \rightsquigarrow \quad \text{solve } P \wedge \neg Q$$

Entailment

$$F_1, \ldots, F_n \models G$$

means every interpretation satisfying all assumptions also satisfies G.

Equivalently:

$$F_1 \wedge \cdots \wedge F_n \wedge \neg G$$

is unsatisfiable.

Conjunction: one interpretation; entailment: all interpretations

Conjunction combines formulas. In one interpretation I:

$$I \models F \wedge G \quad \iff \quad I \models F \text{ and } I \models G.$$

Entailment compares their meaning across every interpretation:

$$F \models G \quad \iff \quad \text{for every } I,\ I \models F \text{ implies } I \models G.$$

For example, $p \wedge q$ is true when $p = q = \top$. But $p \not\models q$, because another interpretation can make $p = \top$ and $q = \bot$.

> Conjunction asks whether both formulas hold here. Entailment asks whether truth of one guarantees the other everywhere.

Entailment means valid implication

$$F \to G$$

is a **formula**. It can be true in one interpretation and false in another.

$$F \models G$$

is a **semantic claim about every interpretation**.

The connection is:

$$F \models G \quad \iff \quad \models (F \to G).$$

For example, $p \to q$ is true when $p = \bot$ and $q = \bot$. But $p \not\models q$, because $p = \top$ and $q = \bot$ is a countermodel.

> Entailment says the implication is valid, not merely true in one case.

First-order logic

A review of objects, relationships, and quantifiers

What first-order logic adds

- A domain of objects
- Variables ranging over that domain
- Functions that produce objects
- Predicates that state facts about objects
- Quantifiers: ∀ and ∃

Propositional connectives remain unchanged.

A first-order language has a signature

Constants: `alice`, `file1`

Functions: $\text{owner}(r), \text{manager}(u)$

Predicates: $\text{Admin}(u), \text{Public}(r), \text{CanAccess}(u, r)$

Each function and predicate has a fixed arity.

Terms denote objects

Terms are built from variables, constants, and function applications:

$$u \qquad \text{alice} \qquad \text{owner}(r) \qquad \text{manager}(\text{owner}(r))$$

A predicate application is a formula—not a term:

$$\text{Admin}(u)$$

Formulas state claims

Atomic formulas apply predicates to terms:

$$\text{CanAccess}(u, r)$$

Larger formulas use connectives and quantifiers:

$$\forall u.\ \text{Admin}(u) \rightarrow \text{Trusted}(u)$$

$$\exists r.\ \text{Public}(r) \wedge \neg\text{Readable}(r)$$

Translate the policy

> Every administrator can access every public resource.

$$\forall u.\, \forall r.\ (\text{Admin}(u) \land \text{Public}(r)) \rightarrow \text{CanAccess}(u, r)$$

What changes if “every public resource” becomes “some public resource”?

Quantifier order matters

$$\forall u.\, \exists r.\ \mathrm{CanAccess}(u, r)$$

Every user can access some resource—possibly a different one.

$$\exists r.\, \forall u.\ \mathrm{CanAccess}(u, r)$$

There is one resource accessible to every user.

These claims are not equivalent.

Scope, free variables, bound variables

Consider:

$$\forall u.\ \mathrm{Admin}(u) \rightarrow \mathrm{CanAccess}(u, r)$$

- Both occurrences of u are bound by $\forall u$
- The occurrence of r is free
- A formula with no free variables is a sentence

Which claim does this open formula make?

A first-order interpretation supplies meaning

An interpretation chooses:

1. A nonempty domain D
2. An object in D for each constant
3. A function over D for each function symbol
4. A relation over D for each predicate symbol

Quantifiers range over the chosen domain.

Same syntax, different worlds

$$\forall x.\, \exists y.\ x < y$$

- Over the integers: true
- Over a finite set with a strict total order: false

The formula alone is not enough; the domain and symbol meanings matter.

Propositional logic vs. first-order logic

Propositional logic	First-order logic
Truth-valued variables	Objects and relations
Finite truth assignments	Interpretations over a domain
Limited expressiveness	Quantified, structural claims
Decidable by SAT solving	General validity is undecidable

More expressiveness changes what automation can guarantee.

SAT solving

Return to the finite Boolean world

The SAT problem

Input: a propositional formula F

Output:

- `SAT` and a satisfying assignment, or
- `UNSAT` if no satisfying assignment exists

SAT asks only whether **at least one** model exists.

Verification becomes a SAT query

To check whether a system can violate a property:

$$\text{System} \wedge \neg\text{Property}$$

SAT → counterexample

UNSAT → guarantee within the model

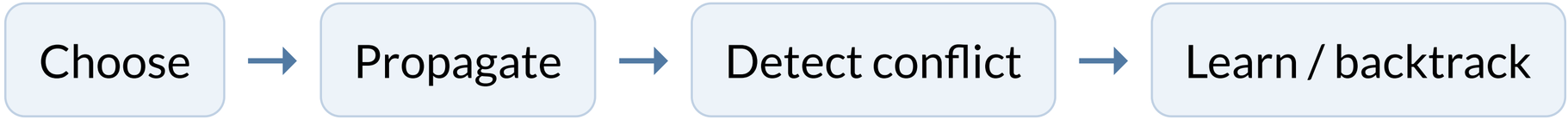
SAT solvers usually consume CNF

Conjunctive normal form is an AND of clauses:

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

- Each clause is an OR of literals
- The whole formula requires every clause to be true

What a modern solver does—conceptually



Then repeat until it finds a model or proves none exists.

> Search supplies possibilities; deduction eliminates them quickly.

Propagation: forced choices

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

The unit clause (p) forces $p = \top$.

Then:

$$q = \top \qquad \text{and then} \qquad r = \top$$

One assignment can trigger a chain of deductions.

Conflict: this branch cannot work

$$(p \vee q) \wedge (\neg p \vee q) \wedge (p \vee \neg q)$$

Suppose the solver tries $q = \bot$.

- First clause forces $p = \top$
- Second clause forces $p = \bot$
- Conflict

The solver must abandon—or learn from—this choice.

Learning avoids repeating mistakes

A conflict explains which earlier choices cannot coexist.

The solver records a new clause that blocks the same bad combination.

$$\text{conflict} \quad \rightsquigarrow \quad \text{new reusable constraint}$$

We will not derive learning algorithms today; retain the idea.

Why SAT solvers can be effective

- Propagation fixes many variables without branching
- Learned clauses summarize failed searches
- Heuristics focus on consequential variables
- Compact encodings expose useful structure

Worst-case complexity remains exponential.

Practical performance depends strongly on the problem and encoding.

What SAT gives us—and what it does not

SAT can provide:

- A concrete satisfying assignment
- An unsatisfiability result, often with checkable evidence
- A reusable engine for many finite reasoning problems

SAT does not decide unrestricted first-order logic directly.

Richer domains require encodings, specialized procedures, or SMT.

Next time

From SAT to SMT

What if our formulas contain integers, arrays, or functions?