CCES – College of Computer Engineering and Sciences

Prince Sattam Bin Abdulaziz University

CS600: Advanced AI

Propositional Logic

Chapter 4 — Syntax, Semantics, and Automated Inference

Dr. Khalil Chebil

Artificial Intelligence Course

Chapter Outline | What we will cover

Learning Objectives

- Master propositional **syntax**, **semantics**, and truth tables
- Apply **equivalences** and convert to **CNF**
- Understand **entailment**, **model checking**, and **resolution**
- Use **DPLL** for SAT and **chaining** for Horn clauses

Section 1

Foundations & Syntax

Why Logic in AI? | A formal language for knowledge & reasoning

Propositional (Boolean) logic is the foundation of knowledge representation and automated reasoning.

- **Declarative** — state *what* is true
- **Compositional** — build complex from simple
- **Unambiguous** — precise semantics
- **Inference** — derive new knowledge
- **Explainable** — transparent steps

Applications

- Expert systems & diagnosis
- Hardware / software verification
- Planning & scheduling
- Semantic web, knowledge graphs
- Circuit design & SAT

Atomic propositions $P, Q, R, \ldots$ each denote a statement that is *true* or *false*.

Symbol	Name	Meaning
$\neg P$	NOT	opposite of P
$P \wedge Q$	AND	both
$P \vee Q$	OR	either or both
$P \rightarrow Q$	IF-THEN	if P then Q
$P \leftrightarrow Q$	IFF	P exactly when Q

Well-formed formulas (WFF)

- Every atom is a WFF
- If α is a WFF, so is $\neg\alpha$
- If α, β are WFFs, so are $(\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta), (\alpha \leftrightarrow \beta)$

$\neg(P \wedge \neg Q)$ ✓ $\quad P \wedge \wedge Q$ ×

Section 2

Semantics & Truth Tables

Semantics: Models & Truth Tables | A model assigns a truth value to every symbol

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \to Q$	$P \leftrightarrow Q$
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Key insight

$P \to Q \equiv \neg P \vee Q$

Key insight

$P \leftrightarrow Q \equiv (P \to Q) \wedge (Q \to P)$

Logical Equivalences | Same truth value in all models: $\alpha \equiv \beta$

Core laws

- **Double negation**: $\neg\neg P \equiv P$
- **Commutative / Associative**
- **Distributive**:
  $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$

De Morgan's laws

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$
$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$$

Eliminations

$P \to Q \equiv \neg P \vee Q$
$P \leftrightarrow Q \equiv (P \to Q) \wedge (Q \to P)$

Three properties

- **Satisfiable**: true in *some* model
  e.g. $P \wedge Q$
- **Unsatisfiable**: false in *all*
  e.g. $P \wedge \neg P$
- **Valid** (tautology): true in *all*
  e.g. $P \vee \neg P$

Entailment $KB \models \alpha$

α is true in every model where KB is true.

$KB : P \rightarrow Q,\ P \quad \Rightarrow \quad KB \models Q$

Refutation principle

$KB \models \alpha \iff (KB \wedge \neg\alpha)$ is **unsatisfiable**.

Conjunctive Normal Form (CNF) | A conjunction of clauses of literals

Definitions

- **Literal**: P or $\neg P$
- **Clause**: disjunction of literals
- **CNF**: conjunction of clauses

$(P \vee Q) \wedge (\neg P \vee R)$ ✓
$(P \wedge Q) \vee R$ ×

Conversion algorithm

1. Eliminate $\rightarrow, \leftrightarrow$
2. Push $\neg$ inward (De Morgan)
3. Distribute $\vee$ over $\wedge$

Example — convert $\neg(P \rightarrow Q)$:

$$\neg(\neg P \vee Q) \;\Rightarrow\; \neg\neg P \wedge \neg Q \;\Rightarrow\; P \wedge \neg Q$$

Section 3

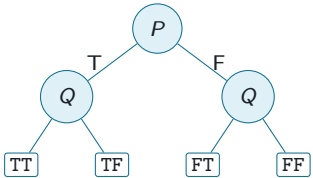
Inference: Model Checking, Resolution, DPLL

TT-Entails? checks $KB \models \alpha$ by trying every assignment of the n symbols.

- For each model: if KB true, require α true
- **Sound** and **complete**

Cost

$\mathcal{O}(2^n)$ — only practical for small formulas.



2^n leaves for n symbols

From two clauses with complementary literals, infer their resolvent:

$$\frac{\ell \vee A, \quad \neg \ell \vee B}{A \vee B}$$

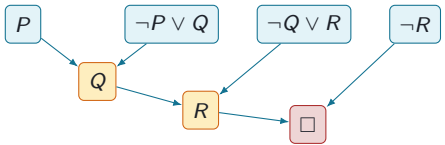
Example rule

$$\frac{P \vee Q, \quad \neg P \vee R}{Q \vee R}$$

Prove $KB \models \alpha$ by showing $KB \wedge \neg\alpha$ derives the **empty clause** $\square$.

Refutation: $KB \models R$

$KB = \{P \to Q,\ Q \to R,\ P\}$



Empty clause $\Rightarrow$ proved.

DPLL Algorithm for SAT | The engine behind modern SAT solvers

Backtracking search with powerful heuristics:

- **Early termination** — stop when decided
- **Pure symbol** — one polarity $\Rightarrow$ assign it
- **Unit clause** — single literal must be true
- **Splitting** — try both values, backtrack

Properties

- Much faster than plain enumeration
- Still exponential worst-case
- Foundation of **CDCL** SAT solvers

Unit propagation + pure literals prune huge parts of the search tree.

Section 4

Horn Clauses & Chaining

Horn Clauses | Restricted but tractable

A **Horn clause** has *at most one* positive literal.

- **Definite**: exactly one positive
  $\neg P \vee \neg Q \vee R \equiv (P \wedge Q \rightarrow R)$
- **Fact**: a single positive literal P
- **Goal**: no positive literals

Why they matter

- Many real KBs are naturally Horn
- Inference is **linear time**
- Foundation of **Prolog** / logic programming

Forward vs. Backward Chaining | Two ways to reason with Horn clauses

Forward (data-driven)

Start from facts, fire rules, derive *all* consequences until the query appears.

Backward (goal-driven)

Start from the query, recurse on rule premises until grounded in facts.

Aspect	Forward	Backward
Direction	Data-driven	Goal-driven
Strategy	Breadth-first	Depth-first
Space	More	Less
Best for	All facts	Specific query

Both complete for definite clauses; $\mathcal{O}(pn)$.

Section 5

Summary

Inference Methods Compared | Choose by problem shape

Method	Compl.	Sound	Complexity	Best for
Truth tables	Yes	Yes	$\mathcal{O}(2^n)$	Small formulas
Resolution	Yes*	Yes	Exponential	General CNF
DPLL	Yes	Yes	Exponential	SAT problems
Forward chain	Yes**	Yes	$\mathcal{O}(pn)$	Derive all facts
Backward chain	Yes**	Yes	$\mathcal{O}(pn)$	Specific queries

*refutation-complete **for Horn clauses

Chapter Summary | Key takeaways

Represent

- Syntax: propositions + connectives
- Semantics: models & truth tables
- **CNF** as the standard inference form

Reason

Model checking, **resolution**, **DPLL**, and chaining.

Limitations

Propositional logic cannot express **quantifiers**, **objects**, **relations**, or **functions** — it must enumerate every instance.

Looking ahead

Ch. 5: **First-Order Logic** adds quantifiers and relations.

Thank you!

Questions & Discussion

Next: Chapter 5 — First-Order Logic