Propositional Logic
Normal Forms

CS560: Reasoning About Programs

Roopsha Samanta

PURDUE
UNIVERSITY

Partly based on slides by Aaron Bradley

Roadmap

Previously

- Course overview
- Program synthesis overview

Today

- Review of propositional logic
- Normal forms for propositional logic

Propositional logic (PL) syntax

Atom	truth symbols	⊤ ("true") and ⊥ ("false")
	propositional variables	p, q, r, p_1, q_1
Literal	atom α or its negation $\neg\alpha$	
Formula	literal or application of a logical connective to F, F_1, F_2	

$\neg F$	"not"	(negation)
$F_1 \vee F_2$	"or"	(disjunction)
$F_1 \wedge F_2$	"and"	(conjunction)
$F_1 \rightarrow F_2$	"implies"	(implication)
$F_1 \leftrightarrow F_2$	"if and only if"	(iff)

PL semantics

Interpretation I : mapping of each propositional variable to a truth value

$$I\colon \{\, p \mapsto \top, q \mapsto \bot, \ldots \,\}$$

Satisfying interpretation : F evaluates to $\top$ under I, written $I \vDash F$

Falsifying interpretation : F evaluates to $\bot$ under I, written $I \nvDash F$

$\phi : (p \wedge q) \to (p \vee \neg q)$ $p \to q$

$I : \{p \mapsto T, q \mapsto \perp\}$

p	q	$p \wedge q$	$\neg q$	$p \vee \neg q$	ϕ
T	$\perp$	$\perp$	T	T	T

PL semantics: inductive definition

Base Cases:

$I \vDash \top$
$I \nvDash \bot$
$I \vDash p$ iff $I[p] = \top$
$I \nvDash p$ iff $I[p] = \bot$

Inductive Cases:

$I \vDash \neg F$ iff $I \nvDash F$
$I \vDash F_1 \vee F_2$ iff $I \vDash F_1$ or $I \vDash F_2$
$I \vDash F_1 \wedge F_2$ iff $I \vDash F_1$ and $I \vDash F_2$
$I \vDash F_1 \rightarrow F_2$ iff $I \nvDash F_1$ or $I \vDash F_2$
$I \vDash F_1 \leftrightarrow F_2$ iff $I \vDash F_1$ and $I \vDash F_2$, or,
$I \nvDash F_1$ and $I \nvDash F_2$

$p \wedge q \Rightarrow \underline{(p \vee \neg q)}$

$I : p \mapsto T, \; q \mapsto F$

1. $I \models p$ $\qquad I[p] = T$
2. $I \not\models q$ $\qquad I[q] = F$
3. $I \models \neg q$ $\qquad$ 2: and "$\neg$"
4. $I \not\models p \wedge q$ $\qquad$ 2 and "$\wedge$"
5. $I \models p \vee \neg q$ $\qquad$ 3/1 and "$\vee$"
6. $I \models \phi$ $\qquad$ 4, 5, "$\rightarrow$"

Satisfiability and Validity

F is **satisfiable** iff there exists $I : I \vDash F$

F is **valid** iff for all $I : I \vDash F$

Duality:
F is valid iff $\neg F$ is unsatisfiable

Procedure for deciding satisfiability *or* validity suffices!

Deciding satisfiability/validity

- SAT solvers! (next lecture)

- Basic techniques
  - **Truth table method**: search-based
  - **Semantic argument method**: deductive technique

- SAT solvers combine search and deduction

Truth table method

1. Enumerate all interpretations
2. Search for satisfying interpretation

Brute-force!
Impractical (2^{n} interpretations)
Can’t be used if domain is not finite, e.g., for first-order logic

$$F: P \wedge Q \to P \vee \neg Q$$

$\boldsymbol{P}$	$\boldsymbol{Q}$	$\boldsymbol{P \wedge Q}$	$\boldsymbol{\neg Q}$	$\boldsymbol{P \vee \neg Q}$	$\boldsymbol{F}$
$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\top$
$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\top$
$\top$	$\bot$	$\bot$	$\top$	$\top$	$\top$
$\top$	$\top$	$\top$	$\bot$	$\top$	$\top$

Semantic argument (decide validity)

Proof by contradiction:

1. Assume F is not valid
2. Apply proof rules
3. Contradiction (i.e, ⊥) along every branch of proof tree ⇒ F is valid
4. Otherwise, F is not valid

A bit of an overhead for PL
Applicable to first-order logic

$$\frac{I \vDash \neg F}{I \nvDash F} \qquad \frac{I \nvDash \neg F}{I \vDash F}$$

$$\text{(and)}\ \frac{I \vDash F \wedge G}{\begin{array}{c} I \vDash F \\ I \vDash G \end{array}} \qquad \frac{I \nvDash F \wedge G}{I \nvDash F \mid I \nvDash G}\ \text{(or)}$$

$$\frac{I \vDash F \vee G}{I \vDash F \mid I \vDash G} \qquad \frac{I \nvDash F \vee G}{\begin{array}{c} I \nvDash F \\ I \nvDash G \end{array}}$$

$$\frac{I \vDash F \rightarrow G}{I \nvDash F \mid I \vDash G} \qquad \frac{I \nvDash F \rightarrow G}{\begin{array}{c} I \vDash F \\ I \nvDash G \end{array}}$$

...

$$\frac{\begin{array}{c} I \vDash F \\ I \nvDash F \end{array}}{I \vDash \bot}$$

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

Assume F is not valid, $I \not\models F$

1. $I \not\models (p \wedge q) \rightarrow (p \vee \neg q)$ ass.
2. $I \models p \wedge q$ $1, \rightarrow$
3. $I \not\models p \vee \neg q$ $1, \rightarrow$
4. $I \models p$ $2, \wedge$
5. $I \not\models p$ $3, \vee$
6. $I \models \perp$ $4, 5, \perp$

Semantic judgements

F_1 and F_2 are **equivalent** ($F_1 \Leftrightarrow F_2$) iff for all $I, I \models F_1 \leftrightarrow F_2$

F_1 implies F_2 ($F_1 \Rightarrow F_2$) iff for all $I, I \models F_1 \rightarrow F_2$

A procedure for deciding satisfiability can decide equivalence and implication!

Normal Forms

A **normal form** for a logic is a syntactical restriction such that for every formula in the logic, there is an equivalent formula in the normal form

Three useful normal forms for propositional logic:

- Negation Normal Form (NNF)
- Disjunctive Normal Form (DNF)
- Conjunctive Normal Form (CNF)

Negation Normal Form (NNF)

Atom	T , ⊥ , propositional variables
Literal	Atom | ¬Atom
Formula	Literal | Formula op Formula
op	∨ | ∧

The only logical connectives are ¬, ∧, ∨

Negations appear only in literals

Conversion to NNF:

Eliminate → and ↔

"Push negations in" using **DeMorgan's Laws**:

$$\neg(F_1 \land F_2) \Leftrightarrow (\neg F_1 \lor \neg F_2)$$

$$\neg(F_1 \lor F_2) \Leftrightarrow (\neg F_1 \land \neg F_2)$$

Backus-Näur

$\neg p \lor \neg q \equiv \neg(p \land q)$

Disjunctive Normal Form (DNF)

Atom	T , ⊥ , propositional variables
Literal	Atom | ¬Atom
Disjunct	Literal ∧ Disjunct
Formula	Disjunct ∨ Formula

Disjunction of conjunction of literals

Conversion to DNF:

First convert to NNF

Distribute ∧ over ∨

$$((F_1 \lor F_2) \land F_3) \Leftrightarrow ((F_1 \land F_3) \lor (F_2 \land F_3))$$

$$(F_1 \land (F_2 \lor F_3)) \Leftrightarrow ((F_1 \land F_2) \lor (F_1 \land F_3))$$

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

?

Deciding satisfiability of DNF formulas is trivial
Why not convert all PL formulas to DNF for SAT solving?
Exponential blow-up of formula size in DNF conversion!

Conjunctive Normal Form (CNF)

Atom	T , ⊥ , propositional variables
Literal	Atom | ¬Atom
Clause	Literal ∨ Clause
Formula	Clause ∧ Formula

Conjunction of disjunction of literals

Conversion to CNF:

First convert to NNF

Distribute ∨ over ∧

$$((F_1 \wedge F_2) \vee F_3) \Leftrightarrow ((F_1 \vee F_3) \wedge (F_2 \vee F_3))$$

$$(F_1 \vee (F_2 \wedge F_3)) \Leftrightarrow ((F_1 \vee F_2) \wedge (F_1 \vee F_3))$$

Deciding satisfiability of CNF formulas is not trivial
CNF conversion must also exhibit an exponential blow-up of formula size
Yet, almost all SAT solvers convert to CNF first before solving. Why?

Equisatisfiability and Tseitin's Transformation

Two formulas F_1 and F_2 are **equisatisfiable** iff:
F_1 is satisfiable iff F_2 is satisfiable

Tseitin's transformation converts any PL formula F_1 to equisatisfiable formula F_2 in CNF with only a **linear** increase in size

Note that equisatisfiability is a much weaker notion than equivalence, but is adequate for checking satisfiability.

Tseitin's Transformation

1. Introduce an auxiliary variable rep(G) for each subformula $G = G_1\ op\ G_2$ of formula F_1
2. Constrain auxiliary variable to be equivalent to subformula: rep(G) $\leftrightarrow$ rep(G_1) op rep(G_2)
3. Convert equivalence constraint to CNF: **CNF**(rep(G) $\leftrightarrow$ rep(G_1) op rep(G_2))
4. Let F_2 be rep(F) $\wedge$ $\bigwedge_G$ **CNF**(rep(G) $\leftrightarrow$ rep(G_1) op rep(G_2)). Check if F_2 is satisfiable.

F_1 and F_2 are equisatisfiable!

Size of each equivalence constraint is bounded by a constant
This restricts the size of F_2 to be linear in the size of F_1: $|F_2| = 30.|F_1| + 2$

Summary

Today

- Review of propositional logic
- Normal forms for propositional logic

Next

- SAT Solving