Lecture 2

Propositional Logic & SAT

Zvonimir Rakamarić
University of Utah

Announcements

- Posted Homework 1
  - Due on Friday morning next week

- Propositional logic: Chapter 1 of our textbook
  - You can download it for free as a PDF

Syntax of Propositional Logic (PL)

truth_symbol ::= $\top$ (true), $\bot$ (false)

variable ::= p, q, r, …

atom ::= truth_symbol | variable

literal ::= atom | $\neg$atom

formula ::= literal |
$\neg$formula |
formula $\wedge$ formula |
formula $\vee$ formula |
formula $\rightarrow$ formula |
formula $\leftrightarrow$ formula

Examples of PL Formulae

$F : \top$

$F : \mathrm{p}$

$F : \neg p$

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

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

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

$F : p \leftrightarrow (q \rightarrow r)$

Semantics

- Semantics provides *meaning* to a formula
  - Defines mechanism for evaluating a formula
  - Formula evaluates to truth values *true*/1 and *false*/0
- Formula F evaluated in two steps
  1) Interpretation I assigns truth values to propositional variables
     $I : \{p \mapsto \mathit{false}, q \mapsto \mathit{true} \ldots\}$
  2) Compute truth value of F based on I using e.g. truth table
- formula F + interpretation I = truth value

Notation

- Let F be a formula and I an interpretation…
- $I[F]$ denotes evaluation of F under I
- If $I[F] = \mathit{true}$ then we say that
  - F is true in I
  - I satisfies F
  - I is a model of F

  and write $I \models F$
- If $I[F] = \mathit{false}$ we write $I \not\models F$

Example

$F : (p \land q) \rightarrow (p \lor \neg q)$

$I : \{p \mapsto 1, q \mapsto 0\}$

(i.e., $I[\,p\,] = 1$, $I[\,q\,] = 0$)

p	q	$\neg q$	$p \land q$	$p \lor \neg q$	F
1	0	1	0	1	1

F evaluates to *true* under I or $I[\,F\,] = \mathit{true}$ or $I \vDash F$...

Satisfiability and Validity

- F is <u>satisfiable</u> iff (if and only if) there exists I such that $I \vDash F$
  - Otherwise, F is unsatisfiable
- F is <u>valid</u> iff for all I, $I \vDash F$
  - Otherwise, F is invalid
- We write $\vDash F$ if F is valid
- Duality between satisfiablity and validity:

  F is valid iff $\neg F$ is unsatisfiable

  Note: only holds if logic is closed under negation

Equivalence

- Two formulae F_1 and F_2 are <u>equivalent</u>, denoted by $F_1 \Leftrightarrow F_2$, iff they have the same models

Decision Procedure for Satisfiability

- Algorithm that in some finite amount of computation decides if given PL formula *F* is satisfiable
  - NP-complete problem
- Modern decision procedures for PL formulae are called *SAT solvers*
- Naïve approach
  - Enumerate truth table
- Modern SAT solvers
  - DPLL algorithm
    - Davis-Putnam-Logemann-Loveland
  - Operates on Conjunctive Normal Form (CNF)

Normal Forms

- Negation Normal Form (NNF)
  - Only allows $\neg$, $\wedge$, $\vee$
  - Negation only in literals
- Disjunctive Normal Form (DNF)
  - Disjunction of conjunction of literals:

$$\bigvee_i \bigwedge_j l_{i,j}$$

- Conjunctive Normal Form (CNF)
  - Conjunction of disjunction of literals:

$$\bigwedge_i \bigvee_j l_{i,j}$$

Negation Normal Form

To transform F into F' in NNF recursively apply the following equivalences:

$$\neg\neg F_1 \Leftrightarrow F_1$$

$$\neg\top \Leftrightarrow \bot$$

$$\neg\bot \Leftrightarrow \top$$

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

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

$$F_1 \rightarrow F_2 \Leftrightarrow \neg F_1 \vee F_2$$

$$F_1 \leftrightarrow F_2 \Leftrightarrow (F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1)$$

Example

$$F : p \leftrightarrow (q \rightarrow r)$$

Conjunctive Normal Form

To transform F into F' in CNF first transform F into NNF and then recursively apply the following equivalences:

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

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

(Note: a disjunction of literals is called a clause.)

Example

$F : p \leftrightarrow (q \rightarrow r)$

Exponential Blow-Up

- Such a naïve transformation can blow-up exponentially (in formula size) for some formulae
  - For example: transforming from DNF into CNF

Tseitin Transformation [1968]

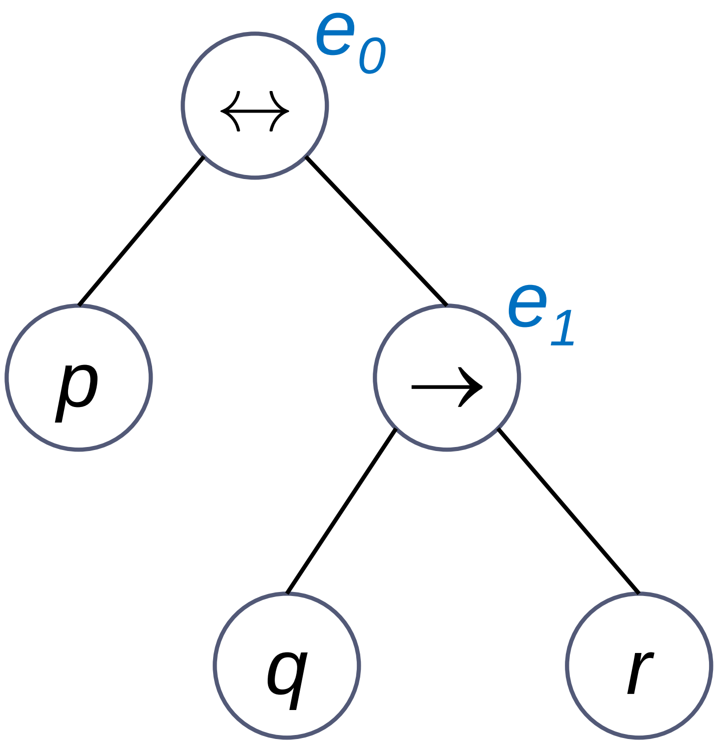
- Used in practice
  - No exponential blow-up
  - CNF formula size is linear wrt original formula
- Does not produce an equivalent CNF
- However, given *F*, the following holds for the computed CNF *F'*:
  - *F'* is equisatisfiable to *F*
  - Every model of *F'* can be translated (i.e., projected) to a model of *F*
  - Every model of *F* can be translated (i.e., completed) to a model of *F'*
- No model is lost or added in the conversion

Tseitin Transformation – Main Idea

- Introduce a fresh variable e_i for every subformula G_i of F
  - e_i represents the truth value of G_i
- Assert that every e_i and G_i pair are equivalent
  - Assertions expressed as CNF
- Conjoin all such assertions in the end

Example

$$F : p \leftrightarrow (q \rightarrow r)$$



SAT Solver Input Format

Based around DIMACS

```
c
c start with comments
c
p cnf 5 3
1 -5 4 0
-1 5 3 4 0
-3 -4 0
```

Using a SAT Solver

- Graph coloring
  - Given a graph and K colors, decide if each vertex can be assigned a color so that no two adjacent vertices have the same color

- How to solve using SAT?

Classical DPLL

- Searching for a model *M* for a given CNF formula *F*
  - Incrementally try to build a model *M*
  - Maintain state during search
- State is a pair *M* | *F*
  - *F* is a set of clauses and it doesn't change during search
  - *M* is a sequence of literals
    - No literals appear twice and no contradiction
    - Order does matter
    - Decision literals marked with l^d

Abstract Transition System

- Contains a set of rules of the form

$$M \mid F \Rightarrow M' \mid F'$$

denoting that search can move from state $M \mid F$ to state $M' \mid F'$

DPLL Rules – Extending M

- Propagate

$M \mid G, C \vee l \;\Rightarrow\; M, l \mid G, C \vee l$

if $M \models \neg C$ and l not in M

- Decide

$M \mid F \;\Rightarrow\; M, l^d \mid F$

if l or $\neg l$ in F and l not in M

DPLL Rules – Adjusting M

- Fail

$M \mid G,C \;\Longrightarrow\; \mathit{fail}$

if $M \models \neg C$ and M contains no decision literals

- Backtrack

$M,l^d,N \mid G,C \;\Longrightarrow\; M,\neg l \mid G,C$

if $M,l^d,N \models \neg C$ and N contains no decision literals

- Propagate

$M \mid G, C \vee l \;\Rightarrow\; M, l \mid G, C \vee l$

if $M \models \neg C$ and l not in M

- Decide

$M \mid F \;\Rightarrow\; M, l^d \mid F$

if l or $\neg l$ in F and l not in M

- Fail

$M \mid G, C \;\Rightarrow\; \mathit{fail}$

if $M \models \neg C$ and M contains no decision literals

- Backtrack

$M, l^d, N \mid G, C \;\Rightarrow\; M, \neg l \mid G, C$

if $M, l^d, N \models \neg C$ and N contains no decision literals

DPLL Example 1

$\emptyset \quad \| \quad \neg p \vee q \vee r,\ p,\ \neg q \vee r,\ \neg q \vee \neg r,\ q \vee r,\ q \vee \neg r$

DPLL Example 2

$\emptyset \quad \| \quad \neg p \vee q,\ \neg r \vee s,\ \neg t \vee \neg u,\ u \vee \neg t \vee \neg q$

Modern SAT Solvers

- DPLL + improvements
  - Backjumping
  - Dynamic variable ordering
  - Learning conflict clauses
  - Random restarts
  - …

Next Lecture

- First-order logic