Lecture Overview

- Finish Resolution in Propositional logics
- Satisfiability problems
- WalkSAT
- Encoding Example
Full Propositional Logics

DEFs.

Literal: an atom or a negation of an atom

Clause: is a disjunction of literals

Conjunctive Normal Form (CNF): a conjunction of clauses

INFERRENCE:

• Convert all formulas in KB and \( \neg \alpha \) in CNF
• Apply Resolution Procedure

\[ p \lor q \lor \neg r \]

\[ (p \lor (q \lor r)) \lor (\neg q \lor p) \]

\[ p \lor q \lor \neg q \lor r \rightarrow p \lor r \]

\[ \text{KB} \vdash \alpha \]

\[ \text{KB} \vdash \neg \alpha \]
Resolution Deduction step

Resolution: inference rule for CNF: sound and complete! *

\((A \lor B \lor C)\)
\((\neg A)\)

\[\therefore (B \lor C)\]

\[\text{“If A or B or C is true, but not A, then B or C must be true.”}\]

\((A \lor B \lor C)\)
\((\neg A \lor D \lor E)\)

\[\therefore (B \lor C \lor D \lor E)\]

\[\text{“If A is false then B or C must be true, or if A is true then D or E must be true, hence since A is either true or false, B or C or D or E must be true.”}\]

\((A \lor B)\)
\((\neg A \lor B)\)

\[\therefore (B \lor B) \equiv B\]

\[\text{Simplification}\]

CPSC 322, Lecture 19
Resolution Algorithm

- The resolution algorithm tries to prove: \( KB \models \alpha \)
- \( KB \land \neg \alpha \) is converted in CNF
- Resolution is applied to each pair of clauses with complementary literals
- Resulting clauses are added to the set (if not already there)

- Process continues until one of two things can happen:

  1. Two clauses resolve in the empty clause. i.e. query is entailed

\[
P \models \neg \neg P \rightarrow \emptyset \quad \therefore KB \models \alpha \rightarrow KB \models \alpha
\]

  2. No new clauses can be added: We find no contradiction, there is a model that satisfies the sentence and hence we cannot entail the query.

\[
KB \models \alpha \models KB \not\models \alpha
\]
Resolution example

\[ KB = (A \iff (B \lor C)) \land \neg A \]

\[ \alpha = \neg B \]

True!

False in all worlds
Resolution algorithm

Proof by contradiction, i.e., show $KB \land \neg \alpha$ unsatisfiable

function PL-RESOLUTION($KB$, $\alpha$) returns true or false
inputs: $KB$, the knowledge base, a sentence in propositional logic
$\alpha$, the query,

$clauses \leftarrow$ the set of clauses in the CNF representation of $KB \land \neg \alpha$
$new \leftarrow \{\}$
loop do
  for each $C_i, C_j$ in $clauses$ do
    $resolvents \leftarrow$ PL-RESOLVE($C_i, C_j$)
    if $resolvents$ contains the empty clause then return true
    $new \leftarrow new \cup resolvents$
    if $new \subseteq clauses$ then return false
  $clauses \leftarrow clauses \cup new$

; no new clauses were created
Lecture Overview

• Finish Resolution in Propositional logics
• Satisfiability problems
• WalkSAT
• Hardness of SAT
• Encoding Example
Satisfiability problems

Consider a CNF sentence, e.g.,

\[(\neg D \lor \neg B \lor C) \land (B \lor \neg A \lor \neg C) \land (\neg C \lor \neg B \lor E) \land (E \lor \neg D \lor B) \land (B \lor E \lor \neg C)\]

Is there an interpretation in which this sentence is true (i.e., that is a model of this sentence)?

Many combinatorial problems can be reduced to checking the satisfiability of propositional sentences (example later)
How can we solve a SAT problem?

Consider a CNF sentence, e.g.,

$$(\neg D \lor \neg B \lor C) \land (A \lor C) \land (\neg C \lor \neg B \lor E) \land (E \lor \neg D \lor B) \land (B \lor E \lor \neg C)$$

*Each clause can be seen as a constraint that reduces the number of interpretations that can be models*

*Eg* $(A \lor C)$ eliminates interpretations in which $A=F$ and $C=F$

So SAT is a Constraint Satisfaction Problem: Find a possible world that is satisfying all the constraints (here all the clauses)
(Stochastic) Local Search Algorithms can be used for this task!

**Evaluation Function:** number of unsatisfied clauses

**WalkSat:** One of the simplest and most effective algorithms:

Start from a randomly generated interpretation
- Pick randomly an unsatisfied clause
- Pick a proposition/atom to flip (randomly 1 or 2)
  1. Randomly
  2. To minimize # of unsatisfied clauses
WalkSAT: Example

\((\neg D \lor \neg B \lor C) \land (A \lor C) \land (\neg C \lor \neg B) \land (E \lor \neg D \lor B) \land (B \lor C)\)

A B C D E
0 0 0 1 0
0 1 0 1 0

pick randomly unsatisfied clause

assume \(E \lor \neg D \lor B\)

pick randomly \(\lor\) or 2

assume 2

\(\neg B \Rightarrow B = 1\)

flip \(E\) # unsat. 2

flip \(D\) # unsat. 2

flip \(B\) # unsat. 2
Pseudocode for WalkSAT

function WALKSAT(clauses, p, max-flips) returns a satisfying model or failure
inputs: clauses, a set of clauses in propositional logic
         p, the probability of choosing to do a “random walk” move
         max-flips, number of flips allowed before giving up

pw ← a random assignment of true/false to the symbols in clauses

for i = 1 to max-flips do
    if pw satisfies clauses then return pw

    clause ← a randomly selected clause from clauses that is false in
              pw

    with probability p flip the value in clause of a randomly selected symbol

    return failure

1 else flip whichever symbol in clause maximizes the number of satisfied clauses

pw = possible world / interpretation
The **WalkSAT** algorithm

If it returns failure after it tries $max$-flips times, what can we say?

A. The sentence is unsatisfiable

B. Nothing

C. The sentence is satisfiable

Typically most useful when we expect a solution to exist
WalkSAT: Example

\((\neg D \lor \neg B \lor C) \land (A \lor C) \land (\neg C \lor \neg B) \land (E \lor \neg D \lor B) \land (B \lor C)\)

- Pick a randomly unsatisfied clause.
  - Assume \((E \lor \neg D \lor B)\)
  - Pick randomly \(1\) or \(2\)
  - Assume \(2\)
  - Flip \(B\) \(\Rightarrow\) \(B = 1\)

A B C D E
0 0 0 1 0
0 1 0 1 0

CPSC 422, Lecture 21
Hard satisfiability problems

Consider *random* 3-CNF sentences. e.g.,

\[(\neg D \lor \neg B \lor C) \land (B \lor \neg A \lor \neg C) \land (\neg C \lor \neg B \lor E) \land (E \lor \neg D \lor B) \land (B \lor E \lor \neg C)\]

\(m = \text{number of clauses (5)}\)
\(n = \text{number of symbols (5)}\)

• Under constrained problems:
  ✓ Relatively few clauses constraining the variables
  ✓ Tend to be easy
  
  E.g. For the above problem 16 of 32 possible assignments are solutions
    – (so 2 random guesses will work on average)
Hard satisfiability problems

What makes a problem hard?

• Increase the number of clauses while keeping the number of symbols fixed
• Problem is more constrained, fewer solutions

• You can investigate this experimentally….
P(satisfiable) for random 3-CNF sentences, $n = 50$

- Hard problems seem to cluster near $m/n = 4.3$ (critical point)

$m = \text{number of clauses}$

$n = \text{number of symbols}$
Lecture Overview

- Finish Resolution in Propositional logics
- Satisfiability problems
- WalkSAT
- Encoding Example
Encoding the Latin Square Problem in Propositional Logic

In combinatorics and in experimental design, a Latin square is an $n \times n$ array filled with $n$ different symbols, each occurring exactly once in each row and exactly once in each column. Here is an example:

\begin{array}{ccc}
A & B & C \\
C & A & B \\
B & C & A \\
\end{array}

Here is another one:
Encoding Latin Square in Propositional Logic: Propositions

Variables must be binary! (They must be propositions)
Each variables represents a color assigned to a cell.
Assume colors are encoded as integers

\[ x_{ijk} \in \{0,1\} \]

Assuming colors are encoded as follows
(black, 1) (red, 2) (blue, 3) (green, 4) (purple, 5)

\[ x_{233} = 0 \]

True or false, ie. 0 or 1 with respect to the interpretation represented by the picture?

How many vars/propositions overall? \( n^3 \)
Encoding Latin Square in Propositional Logic: Clauses

- Some color must be assigned to each cell (clause of length n);

\[ \forall_{ij} (x_{ij1} \lor x_{ij2} \ldots x_{ijn}) \quad \forall_{ik} (x_{ilk} \lor x_{i2k} \ldots x_{ink}) \]

A.

- No color is repeated in the same row (sets of negative binary clauses);

\[ \forall_{ik} (\neg x_{ilk} \lor \neg x_{i2k}) \land (\neg x_{ilk} \lor \neg x_{i3k}) \ldots (\neg x_{ilk} \lor \neg x_{ink}) \ldots (\neg x_{ink} \lor \neg x_{i(n-1)k}) \]

B.

How many clauses?
Encoding Latin Square in Propositional Logic: Clauses

- Some color must be assigned to each cell (clause of length n);

$$\forall ij (x_{ij1} \lor x_{ij2} \ldots x_{ijn}) \quad \forall ik (x_{ilk} \lor x_{i2k} \ldots x_{ink})$$

A.

- No color is repeated in the same row (sets of negative binary clauses);

$$\forall ik (\neg x_{ilk} \lor \neg x_{i2k}) \ldots (\neg x_{ilk} \lor \neg x_{i3k}) \ldots (\neg x_{ilk} \lor \neg x_{ink}) \ldots (\neg x_{i(n-1)k} \lor \neg x_{ink})$$

B. 

$$\frac{n^2 \cdot n \cdot (n-1)}{2} = O(n^4)$$

How many clauses?
Encoding Latin Square Problems in Propositional Logic: FULL MODEL

Variables: \( x_{ijk} \) cell \( i,j \) has color \( k; \ i,j,k=1,2, ..., n. \ x_{ijk} \in \{0,1\} \)

Each variables represents a color assigned to a cell.

Clauses: \( O(n^4) \)

- Some color must be assigned to each cell (clause of length \( n \));
  \[ \forall_{ij} (x_{ij1} \lor x_{ij2} \cdots x_{ijn}) \]

- No color is repeated in the same row (sets of negative binary clauses);
  \[ \forall_{ik} (\neg x_{i1k} \lor \neg x_{i2k}) \land (\neg x_{i1k} \lor \neg x_{i3k}) \cdots (\neg x_{i1k} \lor \neg x_{ink}) \]
  \[ \cdots (\neg x_{in(n-1)k} \lor \neg x_{ink}) \]

- No color is repeated in the same column (sets of negative binary clauses);
  \[ \forall_{jk} (\neg x_{1jk} \lor \neg x_{2jk}) \land (\neg x_{1jk} \lor \neg x_{3jk}) \cdots (\neg x_{1jk} \lor \neg x_{njk}) \]
  \[ \cdots (\neg x_{njk} \lor \neg x_{(n-1)jk}) \]

\[ \text{same as for rows} \]
Logics in AI: Similar slide to the one for planning

- Propositional Definite Clause Logics
- Semantics and Proof Theory
- Satisfiability Testing (SAT)

- Propositional Logics
- First-Order Logics

- Description Logics
- Production Systems

- Ontologies
- Cognitive Architectures

- Semantic Web
- Video Games

- Information Extraction
- Summarization

- Hardware Verification
- Product Configuration
- Tutoring Systems

You will know some applications.
Relationships between different Logics

(better with colors)

**First Order Logic**
\[ \forall X \exists Y p(X,Y) \iff \neg q(Y) \]
- \[ p(a_1,a_2) \]
- \[ \neg q(a_3) \]

**Propositional Logic**
\[ \neg(p \lor q) \rightarrow (r \lor s \lor \top) \]
- \[ p, r \]

**Datalog**
- \[ p(X) \leftarrow q(X) \land r(X,Y) \]
- \[ r(X,Y) \leftarrow s(Y) \]
- \[ s(a_1), q(a_2) \]

**PDDL**
- \[ p \leftarrow s \land f \]
- \[ r \leftarrow s \land q \land p \]
- \[ r \]
- \[ p \]
Learning Goals for today’s class

You can:

• Specify, Trace and Debug the resolution proof procedure for propositional logics
• Specify, Trace and Debug WalkSat
• Explain differences between Proposition Logic and First Order Logic
Announcements

Midterm

• Avg 72    Max 103    Min 13

• If score below 70 need to very seriously revise all the material covered so far

• You can pick up a printout of the solutions along with your midterm
Next class Mon

- First Order Logic
- Extensions of FOL

- Assignment-3 will be posted next week!