Satisfiability

Introduction to Artificial Intelligence

COS302

Michael L. Littman

Fall 2001

Administration

Questions?

Types of Logics

Logic historically a hot topic in AI.

- Propositional logic: Boolean variables (simple)
- First-order logic: more advanced types, objects (expressive)

Book covers first-order logic.
Focus here on propositional.

Propositional Syntax

Formula:

- Constants: T, F
- Variables: $x_1,\ldots,x_n$.
- Negation: ~f (f formula)
- Literal: variable or its negation
- Grouping: (f) (f formula)
- Binary expressions next

Binary Expressions

Given formulae f and g:

- **Conjunction ("and"): fg**
- **Disjunction ("or"): f+g**
- **Implication: $f \rightarrow g$**
- **Equivalence: $f \leftrightarrow g$**

Truth Tables

x	y	xy	x+y	$x \leftrightarrow y$	$x \rightarrow y$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	F	T
F	F	F	F	T	T

Some Equivalences

Write xy in terms of + and ~:

- ~(~x+~y)

Write x$\leftrightarrow$y in terms of the others

- xy+(~x)(~y)

Write x$\rightarrow$y in terms of the others

- ~x+y
- ~(x(~y))

CNF

- Propositional logic syntax is pretty simple, but can be even simpler.
- Conjunctive normal form (CNF) is a conjunction of disjunction of literals (clauses).

(~x+w+v)(x+z+~y)(~w+~y+~v)(v+u+y)(x+~v+u)

Truth Table to CNF

1. Put negation of formula in DNF
   - For each “F” row in table, make a term equivalent to the corresponding assignment
2. Negate the negation
   - By DeMorgan’s Law, ands and ors swap and literals negate

CNF Example

Express $x \leftrightarrow y$ in CNF

1. **Two cases for "F": x=T, y=F and x=F, y=T**
2. **Negation in DNF: x(~y)+(~x)y**
3. **Negate it: (~x+y)(x+~y)**

It works!

Assignments & Models

- Assignment: Mapping of n variables to truth values
- u=F, v=T, w=F, x=T, y=F, z=T
- Satisfying assignment (model): Makes the formula evaluate to T
- (~x+w+v)(x+z+~y)(~w+~y+~v)(v+u+y)(x+~v+u)
- 64 assignments, 31 models.

Categories of Formulae

A Boolean formula can be:

- Valid (tautology): all assignments satisfying.
- Satisfiable: at least one assignment true.
- Unsatisfiable: none true.

Computational Problems

- Given a formula, determine if it is valid: reasoning, proof generation.
- Given a formula, determine if it is satisfiable (SAT): search.
  - ~valid(f) = satisfiable(~f)
- Both hard!

SAT as CSP

SAT is determining satisfiability of formula in CNF. Can be solved as a CSP!

$$(x+y)(\sim x+\sim y)$$

Variables are variables

Domain is T, F

Clauses are constraints

Generic CSP Algorithm

- If all values assigned and no constraints violated, done
- Apply consistency checking
- If deadend, backtrack
- Select variable to be assigned
- Select value for the variable
- Assign variable and recurse

Generic SAT Algorithm

- If all values assigned and no constraints violated, done
- Apply consistency checking
- If deadend, backtrack
- Select variable to be assigned
- Select value for the variable
- Assign variable and recurse

Generic SAT Algorithm

- If all values assigned and no clauses violated, done
- Apply consistency checking
- If deadend, backtrack
- Select variable to be assigned
- Select value for the variable
- Assign variable and recurse

Generic SAT Algorithm

- If all values assigned and no clauses violated, done
- Apply unit propagation
- If deadend, backtrack
- Select variable to be assigned
- Select value for the variable
- Assign variable and recurse

Generic SAT Algorithm

- If all values assigned and no clauses violated, done
- Apply unit propagation
- If unsatisfied clause, backtrack
- Select variable to be assigned
- Select value for the variable
- Assign variable and recurse

Pure Variables

(x+y+z)(x+~y+~w)(w+~z+y)

If x is a pure literal (never appears negated), then if there is a satisfying assignment with x=F, there must also be one with x=T.

So, we need only check one case (no branching).

Purification at Work

(~x+w+v)(x+z+~y)(~w+~y+~v)(v+u+y)(x+~v+u)

z=T

(~x+w+v)(~w+~y+~v)(v+u+y)(x+~v+u)

u=T

(~x+w+v)(~w+~y+~v)

x=F

(~w+~y+~v)

y=F

Formula satisfied

DPLL

Davis-Putnam-Logemann-Loveland (1962) basis of practical SAT algorithms

- Recursive: stop if SAT or UNSAT
- Unit propagation, recurse
- Purification, recurse
- Else, split and recurse on both

Splitting Heuristics

How choose a variable to split?

- Most occurrences
- In short clauses
- Lots more of one kind of literal than another

www.ee.princeton.edu/~chaff

DPLL Analysis

- n variables. Worst case?
- Split on a variable in a shortest clause.
- What if only k literals per clause (k-CNF)? Say, k=2?

Analysis of 2-CNF

Can be made to run in polynomial time.

Analysis of 3-CNF

(x+y+z)(~x+u+v)...

x=T: (u+v)...

u=T: ... (2 vars eliminated)

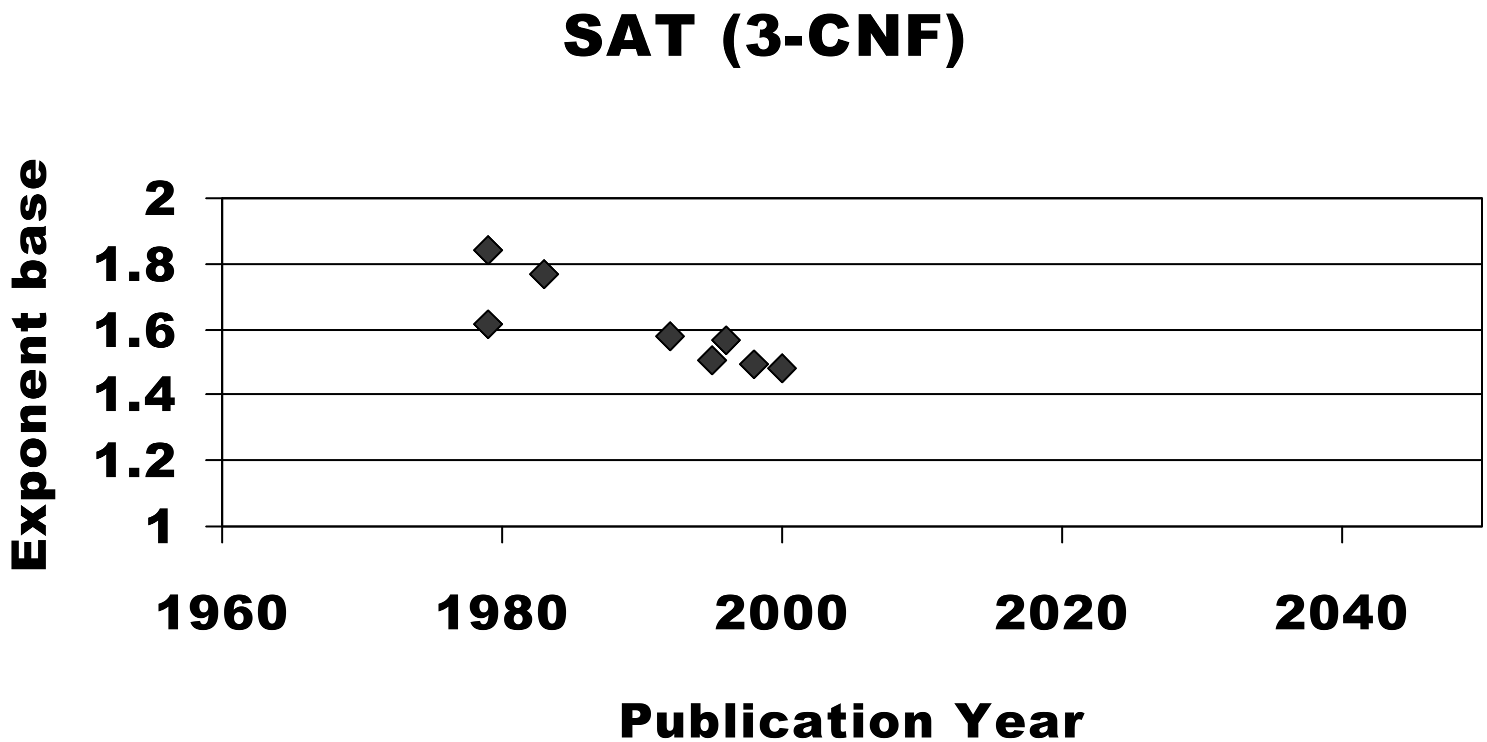
u=F, v=T: ... (3 vars eliminated)

x=F: (y+z)... (same idea)

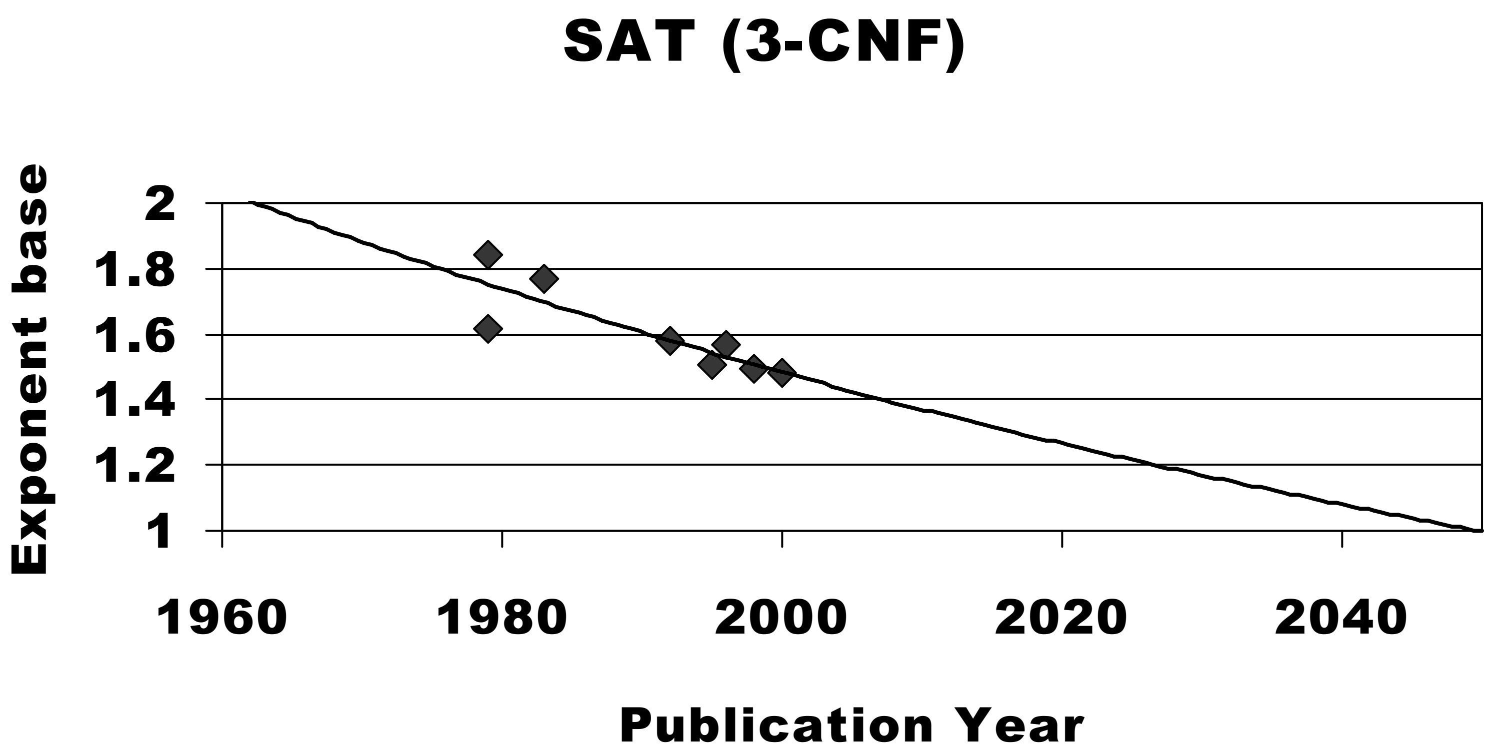
$R(n) \leq 2\,R(n-2) + 2\,R(n-3)$

$\approx 1.769^n$

Analysis Improvements



Analysis Improvements



PHP: Propositional Proof

Pigeonhole Principle:

- If you have n+1 pigeons and n holes and each pigeon is assigned a hole, then some hole contains at least 2 pigeons.

DPLL takes exponential time to prove validity.

3-CNF Conversion Ex.

$\sim(\sim(\sim x+y)\ z)$

Efficient procedure for creating an equivalent 3-CNF expression from an arbitrary propositional expression.

3-CNF Conversion

1. Add a variable for each binary operator in the expression.
2. Create a set of 3-CNF clauses for each of the derived variables.
3. Add a clause for the root node.

What to Learn

- Definition of SAT.
- How to make a CNF expression from a truth table.
- The DPLL algorithm.
- How to make a 3-CNF expression from an arbitrary expression.

Homework 3

1. Let f=~(x+ ~y(~x+z)). (a) Write out the truth table for f. (b) Convert the truth table to CNF. (c) Show the series of steps DPLL makes while solving the resulting formula. Assume variables chosen for splitting in the order x, y, z.
2. Using the same f from the first part, follow the 3-CNF conversion algorithm to create an equivalent 3-CNF formula.