Lecture 10

Pete Manolios
Northeastern

Short History

- Ancients: Logic invented as a scientific field of study by Aristotle (380-322 B.C.)
  - Categorical logic, quantifiers, 2-valued, *satisfiability, validity, …*
- Medieval Logicians: early ideas of mechanization, eg, Lull (1232-1315)
- Leibniz (1646-1716): calculus ratiocinator, a kind of calculating machine
- Stanhope (1753-1816): first machine to solve logic problems
- Boole (1815-1864): Boolean algebra
- Frege (1848-1925): Concept notation, basis for modern formal logic
- Russell & Whitehead, Godel, Herbrand, Pierce, Tarski, …
- Shannon (1940): Boolean logic to minimize circuits
- Davis & Putnam (1958): DP algorithm, DPLL (1962)BDDs (Lee 1959), …, ROBDDs (Bryant 1986, …)
- Bryant, Clarke, Emerson & McMillan received the 1998 Paris Kanellakis Award for "their invention of 'symbolic model checking', a method of formally checking system designs widely used in the computer hardware industry."
- CDCL: decision heuristics, backjumping, learning/forgetting, restarts, pre/in-processing, …

DP SAT Algorithm

- Davis Putnam (1960)
- Input: CNF formula
- Output: SAT/UNSAT
- Idea: apply three rules until
  - Derive the empty clause: UNSAT (identity of $\vee$ is false)
  - No clauses remain: SAT (identity of $\wedge$ is true)
- Three “rules”
  - Pure literal rule (affirmative-negative rule)
  - Unit resolution rule (unit propagation, BCP, 1-literal rule)
  - Resolution (Called consensus, also used for logic minimization)

Pure Literal Rule

- Given F, a set of clauses, and literal ℓ such
  - ℓ appears in F
  - $\neg\ell$ does not appear in F
  - remove all clauses containing ℓ
- Equisatisfiable because we can make ℓ true
- Notice that this always simplifies F
- Modern SAT solvers tend to not use the rule (efficiency)

Boolean Constraint Propagation

Unit resolution rule:

$$\frac{C, \neg\ell \qquad \ell}{C}$$

- BCP: given a set of clauses including $\{\ell\}$
  - remove all other clauses containing ℓ (subsumption)
  - remove all occurrences of $\neg\ell$ in clauses (unit resolution)
  - repeat until a fixpoint is reached

Resolution

Resolution rule:

$$\frac{C, v \qquad D, \neg v}{C, D} \quad \neg v, v \notin C, D$$

Resolution rule:

$$\frac{C_i, p \qquad D_i, \neg p}{C_i, D_i} \quad \neg p \notin C_i \in P, p \notin D_i \in N$$

- Soundness of rule: above line implies below line
- If below line is SAT, so is above line (w/ side conditions)
- Given literal *p*, set of clauses *S*, let *P* be the clauses in *S* that contain *p* only positively and let *N* be the clauses that contain *p* only negatively. Let *E* be the rest of the clauses. Then *S* is SAT iff *S'* is SAT, where *S'*= *E* U the set of all *p*-resolvents of *P* and *N*.
- Proof: If *A* is an assignment for *S*, then if *A*(*p*)=true, all clauses in *N*, with ¬*p* removed are satisfied, so each *p*-resolvent is satisfied. Similarly if *A*(*p*)=false. If *A* is an assignment for *S*', then it satisfies all *Ci* or all *Di*: suppose it doesn't satisfy *Ck*, then it must satisfy all *Di*. If it satisfies all *Ci*, let *A'*(*p*)=false, else *A'*(*p*)=true and *A'*(*x*)=*A*(*x*) otherwise.

Resolution Example

Resolution rule:

$$\frac{C, v \qquad D, \neg v}{C, D}$$

C, D are clauses, ¬v∉C and v∉D

Given literal *p*, set of clauses *S*, let *P* be the clauses in *S* that contain *p* only positively and let *N* be the clauses that contain *p* only negatively. Let *E* be the rest of the clauses. Then *S* is SAT iff *S'* is SAT, where *S'*= *E* U the set of all *p*-resolvents of *P* and *N*.

$\{\{\neg p, q, r, s\}, \{p, \neg q, s\}, \{\neg p, \neg q, r, \neg s\}, \{p, \neg r, \neg s\}, \{\neg p, \neg q, \neg r\}, \{p, q\}, \{\neg p, \neg q, s\}\}$

Resolve on *q*

$\{\neg p, p, r, s\}$

$\{\{p, \neg r, \neg s\}, \{\neg p, r, s\}, \{p, s\}\}$

Notice that clauses that contain a literal and its negation can be thrown away. Why?

Resolution Example

Resolution rule:

$$\frac{C, v \qquad D, \neg v}{C, D}$$ C, D are clauses, ¬v∉C and v∉D

Given literal *p*, set of clauses *S*, let *P* be the clauses in *S* that contain *p* only positively and let *N* be the clauses that contain *p* only negatively. Let *E* be the rest of the clauses. Then *S* is SAT iff *S'* is SAT, where *S'*= *E* U the set of all *p*-resolvents of *P* and *N*.

$\{\{\neg p, q, r, s\}, \{p, \neg q, s\}, \{\neg p, \neg q, r, \neg s\}, \{p, \neg r, \neg s\}, \{\neg p, \neg q, \neg r\}, \{p, q\}, \{\neg p, \neg q, s\}\}$

Resolve on *q* $\{\neg p, p, r, s\}$

$\{\{p, \neg r, \neg s\}, \{\neg p, r, s\}, \{p, s\}\}$

Notice that clauses that contain a literal and its negation can be thrown away. Why?

Resolve on *r*

$\{\{p, s\}\}$ Sat, resolve on *p* to get {} or use pure literal rule

How do we generate a satisfying assignment? Next homework

DP SAT Algorithm

- Input: CNF formula, Output: SAT/UNSAT
- Base case: empty clause: UNSAT
- Base case: no clauses: SAT
  - Apply these two rules until fixpoint
    - Pure literal rule
    - BCP
  - Choose var, say x, perform all possible resolutions, remove trivial clauses and clauses containing x
  - Repeat
- Existentially quantify variables, one at a time
- Problem: space blow-up

Defdata, Macros, History DEMO