LAST LECTURE

- Propositional Logic
  - Syntax and Semantics
  - Two methods for Satisfiability/Validity
  - Truth Table-based method
  - Semantic Argument-based method
- Is the semantic argument method complete?
- What is the time complexity of the semantic argument method?

DECISION PROCEDURES FOR SAT

- We will go through the DPLL algorithm.
  - Davis-Putnam-Logemann-Loveland Algorithm
  - Combines truth table and deductive approaches
  - Requires formulae in Conjunctive Normal Form (CNF)
  - Forms the basis of modern SAT solvers

NORMAL FORMS

- A Normal Form of a formula F is another equivalent formula F′ which obeys some syntactic restrictions.
- Three important normal forms:
  - Negation Normal Form (NNF): Should use only $\neg$, $\wedge$, $\vee$ as the logical connectives, and $\neg$ should only be applied to literals
  - Disjunctive Normal Form (DNF): Should be a disjunction of conjunction of literals
  - Conjunctive Normal Form (CNF): Should be a conjunction of disjunction of literals

CONJUNCTIVE NORMAL FORM

- A conjunction of disjunction of literals

$$\bigwedge_i \bigvee_j \ell_{i,j} \quad \text{for literals } \ell_{i,j}$$

- Each inner disjunct is also called a clause
- Is every formula in CNF also in NNF?

CNF CONVERSION

- We can use distribution of $\vee$ over $\wedge$ to obtain formula in CNF
  - $F_1 \vee (F_2 \wedge F_3) \Leftrightarrow (F_1 \vee F_2) \wedge (F_1 \vee F_3)$
  - Causes exponential blowup!
- Tseitin's transformation algorithm can be used to obtain an equisatisfiable CNF formula linear in size
  - BM Chapter 1

TRUTH TABLE BASED METHOD

Decision Procedure for Satisfiability:
Returns true if F is SAT, false if F is UNSAT

```
SAT(F){
   if (F = True) return true;
   if (F = False) return false;
   p = Choose VARS(F);
   return SAT(F[True/p]) ∨ SAT(F[False/p]);
}
```

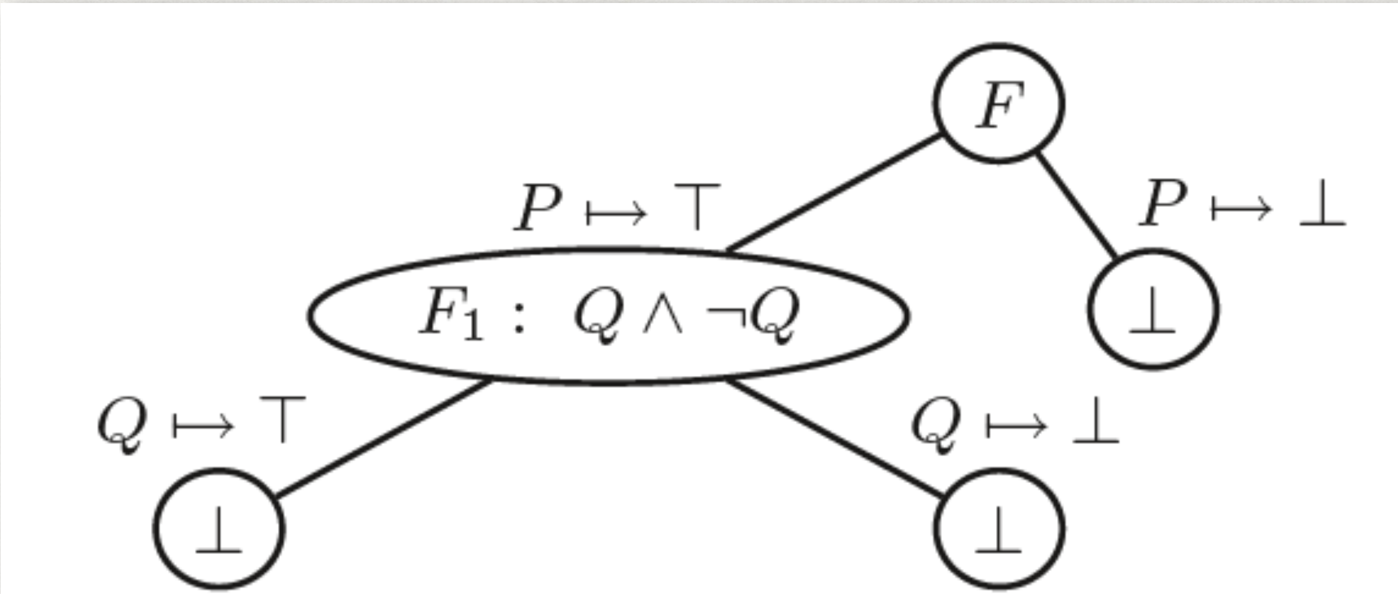
F[G/P] : G REPLACES EVERY OCCURRENCE OF P IN F, THEN SIMPLIFY

SIMPLIFICATION

- Following equivalences can be used to simplify:
  - $F \wedge \bot \Leftrightarrow \bot$
  - $F \wedge \top \Leftrightarrow F$
  - $F \vee \bot \Leftrightarrow F$
  - $F \vee \top \Leftrightarrow \top$

EXAMPLE

- SAT($(P \rightarrow Q) \wedge P \wedge \neg Q$)
- $F = (\neg P \vee Q) \wedge P \wedge \neg Q$
- $F[\top / P] \triangleq (\bot \vee Q) \wedge \top \wedge \neg Q \equiv Q \wedge \neg Q$



SAT MAY SAVE BRANCHING ON SOME OCCASIONS DUE TO SIMPLIFICATION

DEDUCTION: CLAUSAL RESOLUTION

$$\frac{I \vDash p \vee F \qquad I \vDash \neg p \vee G}{I \vDash F \vee G}$$

[CLAUSAL RESOLUTION]

- Given a CNF Formula $F = C_1, C_2, \ldots C_n$, if C' is a resolvent deduced from F, then $F' = C_1, C_2, \ldots, C_n, C'$ is equivalent to F.
- Example: $F = (\neg P \vee Q) \wedge P \wedge \neg Q$
  - Resolvent: Q
  - $F' = (\neg P \vee Q) \wedge P \wedge \neg Q \wedge Q \equiv \bot$

DEDUCTION: UNIT RESOLUTION

$$\frac{I \vDash p \qquad I \vDash \neg p \vee F}{I \vDash F} \quad \text{[UNIT RESOLUTION]}$$

- In Unit Resolution, the resolvent replaces the original clause.
  - Can this be done in clausal resolution? Can the resolvent replace any of the original clauses?

BOOLEAN CONSTRAINT PROPAGATION (BCP)

$$I \vDash p \wedge (\neg p \vee q) \wedge (r \vee \neg q \vee s)$$

BOOLEAN CONSTRAINT PROPAGATION (BCP)

$$\frac{I \models p \wedge (\neg p \vee q) \wedge (r \vee \neg q \vee s)}{I \models q \wedge (r \vee \neg q \vee s)}$$

[UNIT RESOLUTION]

BOOLEAN CONSTRAINT PROPAGATION (BCP)

$$\frac{\dfrac{I \models p \wedge (\neg p \vee q) \wedge (r \vee \neg q \vee s)}{I \models q \wedge (r \vee \neg q \vee s)}}{I \models r \vee s}$$

[UNIT RESOLUTION]

BOOLEAN CONSTRAINT PROPAGATION (BCP)

$$\frac{\dfrac{I \vDash p \wedge (\neg p \vee q) \wedge (r \vee \neg q \vee s)}{I \vDash q \wedge (r \vee \neg q \vee s)}}{I \vDash r \vee s}$$

[UNIT RESOLUTION]

FIND A SATISFYING INTERPRETATION

PURE LITERAL PROPAGATION (PLP)

- If a variable appears only positively or negatively in a formula, then all clauses containing the variable can be removed.
  - p appears positively if every $p-$literal is just p
  - p appears negatively if every $p-$literal is $\neg p$
- Removing such clauses from F results in a equisatisfiable formula F'
  - Why?
  - Are F and F' equivalent?

DPLL

Decision Procedure for Satisfiability:
Returns true if F is SAT, false if F is UNSAT

```
SAT(F){

  F’ = PLP(F);

  F’’ = BCP(F’);

  if (F’’ = True) return true;

  if (F’’ = False) return false;

  p = Choose VARS(F’’);

  return SAT(F’’[True/p]) ∨ SAT(F’’[False/p]);

}
```

EXAMPLE

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

- SAT(F)
  - No PLP or BCP.
  - q ← CHOOSE.
  - F[True/q] = $r \wedge \neg r \wedge (p \vee \neg r)$
- SAT(F[True/q])
  - After PLP: $r \wedge \neg r$
  - After BCP: False
  - Return False and backtrack to previous call

```
SAT(F){

  F' = PLP(F);

  F'' = BCP(F');

  if (F'' = True) return true;

  if (F'' = False) return
false;

  p = Choose VARS(F);

  return SAT(F''[True/p]) ∨
  SAT(F''[False/p]);

}
```

EXAMPLE

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

- SAT(F)
  - No PLP or BCP.
  - q ← CHOOSE.
  -

```
SAT(F){

  F' = PLP(F);

  F'' = BCP(F');

  if (F'' = True) return true;

  if (F'' = False) return
false;

  p = Choose VARS(F);

  return SAT(F''[True/p]) ∨
  SAT(F''[False/p]);

}
```

EXAMPLE

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

- SAT(F)
  - No PLP or BCP.
  - q ← CHOOSE.
  - F[False/q] = $\neg p \vee r$

```
SAT(F){
  F' = PLP(F);
  F'' = BCP(F');
  if (F'' = True) return true;
  if (F'' = False) return
false;
  p = Choose VARS(F);
  return SAT(F''[True/p]) ∨
  SAT(F''[False/p]);
}
```

EXAMPLE

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

- SAT(F)
  - No PLP or BCP.
  - q ← CHOOSE.
  - F[False/q] = $\neg p \vee r$
- SAT(F[False/q])
  -

```
SAT(F){

  F' = PLP(F);

  F'' = BCP(F');

  if (F'' = True) return true;

  if (F'' = False) return
false;

  p = Choose VARS(F);

  return SAT(F''[True/p]) ∨
  SAT(F''[False/p]);

}
```

EXAMPLE

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

- SAT(F)
  - No PLP or BCP.
  - q ← CHOOSE.
  - F[False/q] = $\neg p \vee r$
- SAT(F[False/q])
  - After PLP: True
  - Satisfiable!

```
SAT(F){
  F' = PLP(F);
  F'' = BCP(F');
  if (F'' = True) return true;
  if (F'' = False) return false;
  p = Choose VARS(F);
  return SAT(F''[True/p]) ∨ SAT(F''[False/p]);
}
```

DPLL IS JUST THE STARTING POINT!

- Modern SAT solvers use a variety of approaches to further improve the performance
  - Non-chronological back tracking
  - Conflict-driven clause learning (CDCL)
  - Heuristics to CHOOSE appropriate variables and assignments
- Current SAT solvers can solve problems with millions of clauses in reasonable amount of time on average.