CS6410 Software Verification

#2. Overview, Normal Forms and SAT

Ashish Mishra

with Inputs from Roopsha Samantha's course at Purdue

Overview: Module I : Fundamentals

- Propositional Logic: Theory, Implementation and Applications
- First-Order Logic
- First-Order Theories
- Satisfiability Modulo Theories
- Program Semantics and Specifications
- Induction
- Dafny Verifier (https://dafny.org/)

Overview: Module II : Model Checking

- Transition Systems
- Model Checking
- Model Checking Theory
- Model Checking Applications:
  - CBMC (https://www.cprover.org/cbmc)

Overview: Module II : Abstract Interpretation

- Static and Dynamic Program Analysis
- Abstract Interpretation Theory
- Abstract Interpretation Application
  - Static DataFlow Analysis Frameworks:
    - Soot for Java (http://soot-oss.github.io/soot)
    - NASA IKOS for C/C++ (https://github.com/NASA-SW-VnV/ikos)
  - Dynamic Analysis Framework (https://github.com/analysis-tools-dev)
    - Jalangi : JavaScript
    - KLEE, Valgrind C/C++

Example : Recap PL formula

formula $F : (P \wedge Q) \rightarrow (\top \vee \neg Q)$
atoms: P, Q, $\top$
literals: P, Q, $\top$, $\neg Q$
subformulae: P, Q, $\top$, $\neg Q$, $P \wedge Q$, $\top \vee \neg Q$, F
abbreviation

$$F : P \wedge Q \rightarrow \top \vee \neg Q$$

PL Semantics (Meaning)

Sentence F + Interpretation I = Truth value (true, false)

Interpretation

$$I : \{P \mapsto \text{true}, Q \mapsto \text{false}, \cdots\}$$

Evaluation of F under I:

F	$\neg F$
0	1
1	0

where 0 corresponds to value false
1 true

$I \models F$ if F evaluates to true under I
$I \not\models F$ false

F_1	F_2	$F_1 \wedge F_2$	$F_1 \vee F_2$	$F_1 \rightarrow F_2$	$F_1 \leftrightarrow F_2$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

Satisfying and Falsifying Interpretations

PL Semantics (Inductive definitions)

Base Case:

$$I \models \top$$

$$I \not\models \bot$$

$$I \models P \quad \text{iff} \quad I[P] = \text{true}$$

$$I \not\models P \quad \text{iff} \quad I[P] = \text{false}$$

Inductive Case:

$$I \models \neg F \quad \text{iff } I \not\models F$$

$$I \models F_1 \wedge F_2 \quad \text{iff } I \models F_1 \text{ and } I \models F_2$$

$$I \models F_1 \vee F_2 \quad \text{iff } I \models F_1 \text{ or } I \models F_2$$

$$I \models F_1 \rightarrow F_2 \quad \text{iff, if } I \models F_1 \text{ then } I \models F_2$$

$$I \models F_1 \leftrightarrow F_2 \quad \text{iff, } I \models F_1 \text{ and } I \models F_2, \text{ or } I \not\models F_1 \text{ and } I \not\models F_2$$

Note:

$$I \not\models F_1 \rightarrow F_2 \quad \text{iff} \quad I \models F_1 \text{ and } I \not\models F_2$$

Satisfiability and Validity

F is satisfiable iff there exists $I : I \vDash F$

F is valid iff for all $I : I \vDash F$

Duality:
F is valid iff $\neg F$ is unsatisfiable

Procedure for deciding satisfiability *or* validity suffices!

Method 2: Semantic Argument

Proof by contradiction:

1. Assume F is not valid
2. Apply proof rules
3. Contradiction (i.e, $\bot$) along every branch of proof tree $\Rightarrow F$ is valid
4. Otherwise, F is not valid

A bit of an overhead for PL
Applicable to first-order logic

Proof rules

$$\frac{I \models \neg F}{I \not\models F} \qquad \frac{I \not\models \neg F}{I \models F}$$

$$\frac{I \models F \wedge G}{\begin{array}{l} I \models F \\ I \models G \end{array}} \leftarrow \text{and} \qquad \frac{I \not\models F \wedge G}{I \not\models F \mid I \not\models G} \nwarrow \text{or}$$

$$\frac{I \models F \vee G}{I \models F \mid I \models G} \qquad \frac{I \not\models F \vee G}{\begin{array}{l} I \not\models F \\ I \not\models G \end{array}}$$

$$\frac{I \models F \rightarrow G}{I \not\models F \mid I \models G} \qquad \frac{I \not\models F \rightarrow G}{\begin{array}{l} I \models F \\ I \not\models G \end{array}}$$

$$\frac{I \models F \leftrightarrow G}{I \models F \wedge G \mid I \not\models F \vee G} \qquad \frac{\begin{array}{l} I \models F \\ I \not\models F \end{array}}{I \models \bot}$$

$$\frac{I \not\models F \leftrightarrow G}{I \models F \wedge \neg G \mid I \models \neg F \wedge G}$$

Example

To Prove $F: P \wedge Q \rightarrow P \vee \neg Q$ is valid.

Let's assume that F is not valid and that I is a falsifying interpretation.

1.	I	$\not\models$	$P \wedge Q \rightarrow P \vee \neg Q$	assumption
2.	I	$\models$	$P \wedge Q$	1 and $\rightarrow$
3.	I	$\not\models$	$P \vee \neg Q$	1 and $\rightarrow$
4.	I	$\models$	P	2 and $\wedge$
5.	I	$\not\models$	P	3 and $\vee$
6.	I	$\models$	$\perp$	4 and 5 are contradictory

Thus F is valid

Example 2

To Prove

$F: (P \rightarrow Q) \wedge (Q \rightarrow R) \rightarrow (P \rightarrow R)$ is valid.

Example 2

Let's assume that F is not valid.

1.	I	$\not\models$	F	assumption
2.	I	$\models$	$(P \rightarrow Q) \wedge (Q \rightarrow R)$	1 and $\rightarrow$
3.	I	$\not\models$	$P \rightarrow R$	1 and $\rightarrow$
4.	I	$\models$	P	3 and $\rightarrow$
5.	I	$\not\models$	R	3 and $\rightarrow$
6.	I	$\models$	$P \rightarrow Q$	2 and of $\wedge$
7.	I	$\models$	$Q \rightarrow R$	2 and of $\wedge$

Example 2

Two cases from 6

$$\begin{array}{llll} 8a. & I & \not\models & P \quad \text{6 and } \rightarrow \\ 9a. & I & \models & \bot \quad \text{4 and 8a are contradictory} \end{array}$$

and

$$8b. \quad I \quad \models \quad Q \qquad \text{6 and } \rightarrow$$

Two cases from 7

$$\begin{array}{llll} 9ba. & I & \not\models & Q \quad \text{7 and } \rightarrow \\ 10ba. & I & \models & \bot \quad \text{8b and 9ba are contradictory} \end{array}$$

and

$$\begin{array}{llll} 9bb. & I & \models & R \quad \text{7 and } \rightarrow \\ 10bb. & I & \models & \bot \quad \text{5 and 9bb are contradictory} \end{array}$$

Our assumption is incorrect in all cases — F is valid.

Example 3: Is

$$F: \; P \vee Q \; \rightarrow \; P \wedge Q$$

valid? Assume F is not valid:

1.	$I \not\models P \vee Q \rightarrow P \wedge Q$	assumption
2.	$I \models P \vee Q$	1 and $\rightarrow$
3.	$I \not\models P \wedge Q$	1 and $\rightarrow$
4a.	$I \models P$	2, $\vee$ (case a)
5aa.	$I \not\models P$	3, $\vee$ (subcase aa)
6aa.	$I \models \bot$	4a, 5aa
5ab.	$I \not\models Q$	3, $\vee$ (subcase ab)
6ab.	?	
5a.	?	

4b.		$I \models Q$	2, $\vee$ (case b)
5ba.		$I \not\models P$	3, $\vee$ (subcase ba)
6ba.		?	
5bb.		$I \not\models Q$	3, $\vee$ (subcase bb)
6bb.		$I \models \bot$	4b, 5bb
5b.	?		
5.	?		

We cannot derive a contradiction in both cases (4a and 4b), so we cannot prove that F is valid. To demonstrate that F is not valid, however, we must find a falsifying interpretation (here are two):

$I_1 : \{P \mapsto \text{true}, Q \mapsto \text{false}\}$ $I_2 : \{Q \mapsto \text{true}, P \mapsto \text{false}\}$

Note: we have to derive a contradiction in all cases for F to be valid!

Semantic judgements, Equivalence

F_1 and F_2 are equivalent ($F_1 \Leftrightarrow F_2$)

iff for all interpretations I, $I \models F_1 \leftrightarrow F_2$

To prove $F_1 \Leftrightarrow F_2$ show $F_1 \leftrightarrow F_2$ is valid.

F_1 implies F_2 ($F_1 \Rightarrow F_2$)

iff for all interpretations I, $I \models F_1 \rightarrow F_2$

$F_1 \Leftrightarrow F_2$ and $F_1 \Rightarrow F_2$ are not formulae!

Homework

Example: Show

$$P \to Q \Leftrightarrow \neg P \vee Q$$

i.e.

$$F : (P \to Q) \leftrightarrow (\neg P \vee Q) \textit{ is valid}.$$

Assume F is not valid, then we have two cases:

Case a: $I \not\models \neg P \vee Q$,

$I \models P \to Q$

Case b: $I \models \neg P \vee Q$,

$I \not\models P \to Q$

Derive contradictions in both cases.

Normal Forms

Normal Forms

- A normal form for a logic is a syntactical restriction such that for every formula in the logic, there is an equivalent formula in the normal form.

- Three useful normal forms for propositional logic:
  - Negation Normal Form (NNF)
  - Disjunctive Normal Form (DNF)
  - Conjunctive Normal Form (CNF)

Negation Normal Form (NNF)

Atom	T , ⊥ , propositional variables
Literal	Atom | ¬Atom
Formula	Literal | Formula op Formula
op	∨ | ∧

The only logical connectives are ¬, ∧, ∨

Negations appear only in literals

Conversion to NNF:

Eliminate → and ↔

"Push negations in" using DeMorgan's Laws:

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

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

Example: Convert $F : \neg(P \rightarrow \neg(P \wedge Q))$ to NNF

Example: Convert $F : \neg(P \rightarrow \neg(P \wedge Q))$ to NNF

$F' : \neg(\neg P \vee \neg(P \wedge Q))$ → to ∨

$F'' : \neg\neg P \wedge \neg\neg(P \wedge Q)$ De Morgan's Law

$F''' : P \wedge P \wedge Q$ ¬¬

F ′′′ is equivalent to F (F ′′′ ⇔ F) and is

Disjunctive Normal Form (DNF)

Atom	⊤ , ⊥ , propositional variables
Literal	Atom | ¬Atom
Disjunct	Literal ∧ Disjunct
Formula	Disjunct ∨ Formula

Disjunction of conjunctions of literals

$$\bigvee_i \bigwedge_j \ell_{i,j} \quad \text{for literals } \ell_{i,j}$$

Conversion to DNF:

First convert to NNF

Distribute ∧ over ∨

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

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

Deciding satisfiability of DNF formulas is trivial
Why not convert all PL formulas to DNF for SAT solving?
Exponential blow-up of formula size in DNF conversion!

Example

Example: Convert

$$F : (Q_1 \vee \neg\neg Q_2) \wedge (\neg R_1 \rightarrow R_2) \text{ into DNF}$$

$$F' : (Q_1 \vee Q_2) \wedge (R_1 \vee R_2) \quad \text{in NNF}$$

$$F'' : (Q_1 \wedge (R_1 \vee R_2)) \vee (Q_2 \wedge (R_1 \vee R_2)) \quad \text{dist}$$

$$F''' : (Q_1 \wedge R_1) \vee (Q_1 \wedge R_2) \vee (Q_2 \wedge R_1) \vee (Q_2 \wedge R_2) \quad \text{dist}$$

F ′′′ is equivalent to F (F ′′′ $\Leftrightarrow$ F) and
is in DNF

Conjunctive Normal Form (CNF)

Atom	T , ⊥ , propositional variables
Literal	Atom | ¬Atom
Clause	Literal ∨ Clause
Formula	Clause ∧ Formula

Conjunction of disjunctions of literals

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

Conversion to CNF:

First convert to NNF

Distribute ∨ over ∧

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

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

Deciding satisfiability of CNF formulas is not trivial
CNF conversion must also exhibit an exponential blow-up of formula size
Yet, almost all SAT solvers convert to CNF first before solving. Why?

Natural representation because in practice, many formulas arise from multiple constraints that must hold *simultaneously* (AND).

Example: Convert

$$F : P \leftrightarrow (Q \rightarrow R)$$

to an equivalent formula F' in CNF.

First get rid of $\leftrightarrow$:

$$F_1 : (P \rightarrow (Q \rightarrow R)) \wedge ((Q \rightarrow R) \rightarrow P)$$

Now replace $\rightarrow$ with $\vee$:

$$F_2 : (\neg P \vee (\neg Q \vee R)) \wedge (\neg(\neg Q \vee R) \vee P)$$

Drop unnecessary parentheses and apply De Morgan's Law:

$$F_3 : (\neg P \vee \neg Q \vee R) \wedge ((\neg\neg Q \wedge \neg R) \vee P)$$

Simplify double negation (now in NNF):

$$F_4 : (\neg P \vee \neg Q \vee R) \wedge ((Q \wedge \neg R) \vee P)$$

Distribute disjunction over conjunction (now in CNF):

$$F' : (\neg P \vee \neg Q \vee R) \wedge (Q \vee P) \wedge (\neg R \vee P)$$

Potential Problem with CNF: Size blowup

Distributivity will duplicate entire subformulas

Can happen repeatedly: $(p_1 \wedge p_2 \wedge p_3) \vee (q_1 \wedge q_2 \wedge q_3) =$
$(p_1 \vee (q_1 \wedge q_2 \wedge q_3)) \wedge (p_2 \vee (q_1 \wedge q_2 \wedge q_3)) \wedge (p_3 \vee (q_1 \wedge q_2 \wedge q_3))$
$= (p_1 \vee q_1) \wedge (p_1 \vee q_2) \wedge (p_1 \vee q_3)$
$\wedge (p_2 \vee q_1) \wedge (p_2 \vee q_2) \wedge (p_2 \vee q_3)$
$\wedge (p_3 \vee q_1) \wedge (p_3 \vee q_2) \wedge (p_3 \vee q_3)$

Worst-case blowup? : exponential!

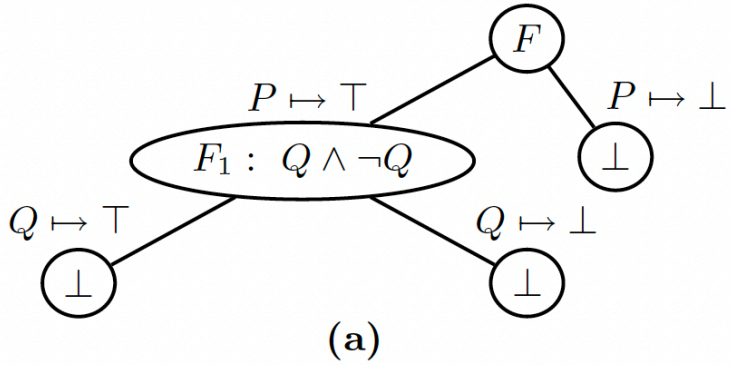
Can't use this transformation for subsequent algorithms (e.g., satisfiability checking) if resulting formula is inefficiently large (possibly too large to represent/process).

Decision Procedure for SAT

```
let rec SAT F =
  if F = ⊤ then true
  else if F = ⊥ then false
  else
    let P = CHOOSE vars(F) in
    (SAT F{P ↦ ⊤}) ∨ (SAT F{P ↦ ⊥})
```

(Choose a var to branch)

Example



(a)

$$F : \ (P \rightarrow Q) \wedge P \wedge \neg Q \ .$$

$$F\{P \mapsto \top\} : \ (\top \ \rightarrow \ Q) \wedge \top \wedge \neg Q \ ,$$

$$F_1 : \ Q \wedge \neg Q$$

$$F_1\{Q \mapsto \top\} \quad \text{and} \quad F_1\{Q \mapsto \bot\}$$

Both reduce to False

$$F\{P \mapsto \bot\} : \ (\bot \ \rightarrow \ Q) \wedge \bot \wedge \neg Q \ ,$$

Reduces to False

UNSAT

Equisatisfiability and Tseitin's Transformation

Two formulas F_1 and F_2 are equisatisfiable iff:
F_1 is satisfiable iff F_2 is satisfiable

Note that equisatisfiability is a much weaker notion than equivalence, but is adequate for checking satisfiability.

to decide the satisfiability of F,
we need only
examine a formula F′ such that F and F′ are equisatisfiable.

Tseitin's transformation converts any PL formula F_1 to equisatisfiable formula F_2 in CNF with only a linear increase in size

Tseitin's Transformation

1. Introduce an auxiliary variable rep(G) for each subformula $G = G_1\ op\ G_2$ of formula F_1
2. Constrain auxiliary variable to be equivalent to subformula: rep(G) ↔ rep(G_1) op rep(G_2)
3. Convert equivalence constraint to CNF: **CNF**(rep(G) ↔ rep(G_1) op rep(G_2))
4. Let F_2 be rep(F) ∧ $\bigwedge_G$ **CNF**(rep(G) ↔ rep(G_1) op rep(G_2)). Check if F_2 is satisfiable.

F_1 and F_2 are equisatisfiable!

Size of each equivalence constraint is bounded by a constant
This restricts the size of F_2 to be linear in the size of F_1: $|F_2| = 30.|F_1| + 2$

Tseitin Transformation: Example

Add numbered proposition for each operator:

$(a \overset{1}{\wedge} \neg b) \vee \neg(c \overset{2}{\wedge} d)$

no need to number negations

nor top-level operator $(\ldots) \vee (\ldots)$

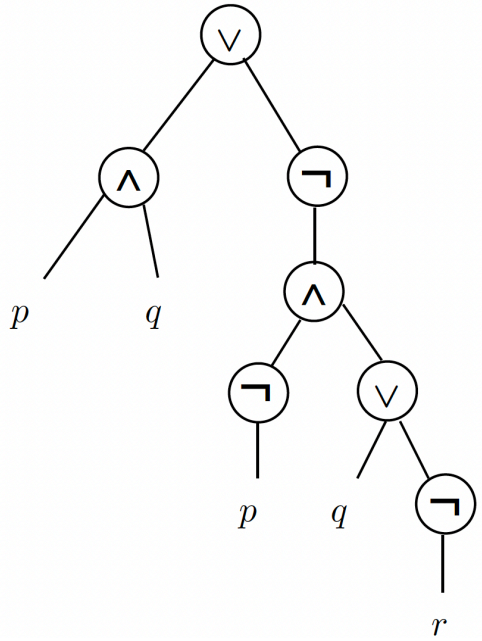
New propositions: $p_1 \leftrightarrow a \overset{1}{\wedge} \neg b, \quad p_2 \leftrightarrow c \overset{2}{\wedge} d$.

Rewrite equivalences for new propositions in CNF,
conjunct with top-level operator of formula:

	overall formula
$(p_1 \vee \neg p_2)$	
$\wedge\,(\neg a \vee b \vee p_1) \wedge (a \vee \neg p_1) \wedge (\neg b \vee \neg p_1)$	$p_1 \leftrightarrow a \wedge \neg b$
$\wedge\,(\neg c \vee \neg d \vee p_2) \wedge (c \vee \neg p_2) \wedge (d \vee \neg p_2)$	$p_2 \leftrightarrow c \wedge d$

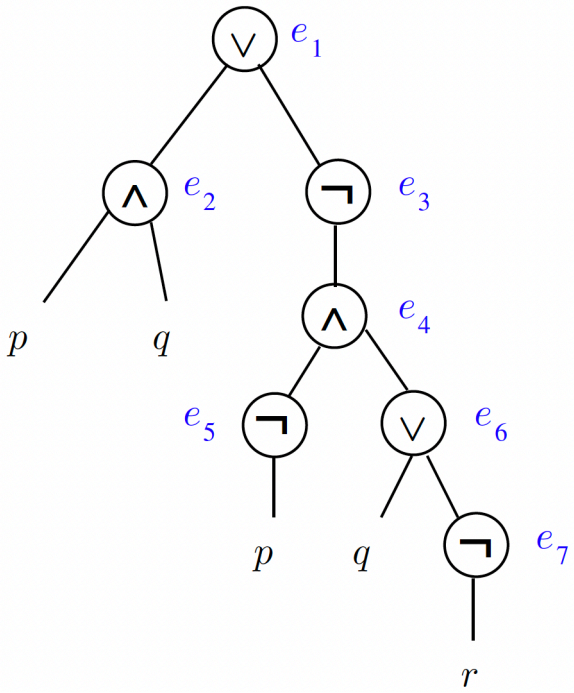
Example

Let F be $(p \wedge q) \vee \neg(\ \neg p \wedge (q \vee \neg r)\ )$



Example

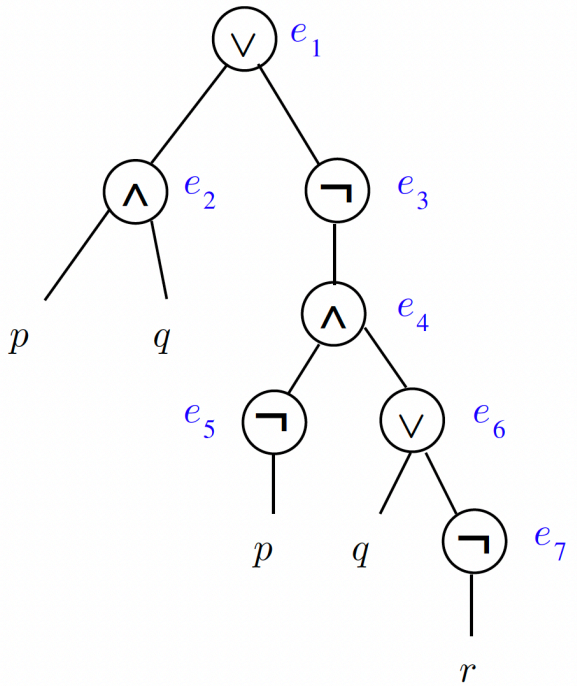
Let F be $(p \wedge q) \vee \neg(\ \neg p \wedge (q \vee \neg r)\ )$



- e_1
- $e_1 \leftrightarrow e_2 \vee e_3$
- $e_2 \leftrightarrow p \wedge q$
- $e_3 \leftrightarrow \neg e_4$
- $e_4 \leftrightarrow e_5 \wedge e_6$
- $e_5 \leftrightarrow \neg p$
- $e_6 \leftrightarrow q \vee \neg e_7$
- $e_7 \leftrightarrow \neg r$

Example

Let F be $(p \wedge q) \vee \neg(\ \neg p \wedge (q \vee \neg r)\ )$

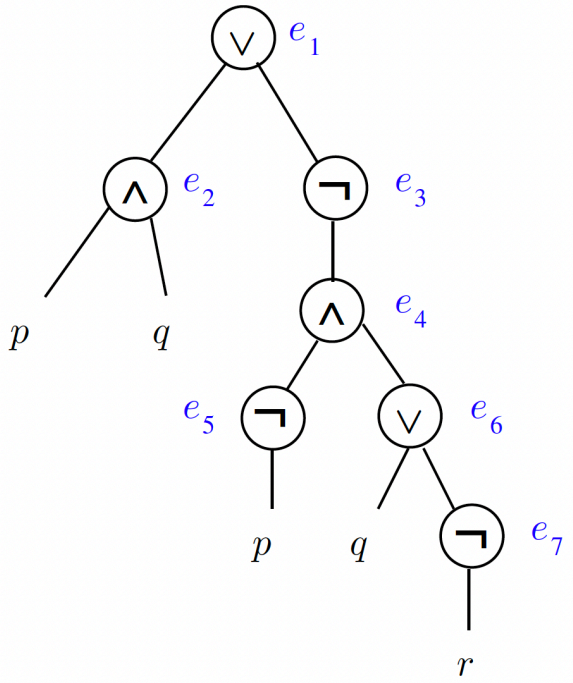


- e_1
- $e_1 \leftrightarrow e_2 \vee e_3$

  $\neg e_1 \vee e_2 \vee e_3$
  $\neg e_2 \vee e_1$
  $\neg e_3 \vee e_1$
- $e_2 \leftrightarrow p \wedge q$
- $e_3 \leftrightarrow \neg e_4$
- $e_4 \leftrightarrow e_5 \wedge e_6$
- $e_5 \leftrightarrow \neg p$
- $e_6 \leftrightarrow q \vee \neg e_7$
- $e_7 \leftrightarrow \neg r$

Example

Let F be $(p \wedge q) \vee \neg(\ \neg p \wedge (q \vee \neg r)\ )$



- e_1
- $e_1 \leftrightarrow e_2 \vee e_3$
  $\neg e_1 \vee e_2 \vee e_3$
  $\neg e_2 \vee e_1$
  $\neg e_3 \vee e_1$
- $e_2 \leftrightarrow p \wedge q$
  $\neg p \vee \neg q \vee e_2$
  $\neg e_2 \vee p$
  $\neg e_2 \vee q$
- $e_3 \leftrightarrow \neg e_4$
- $e_4 \leftrightarrow e_5 \wedge e_6$
- $e_5 \leftrightarrow \neg p$
- $e_6 \leftrightarrow q \vee \neg e_7$
- $e_7 \leftrightarrow \neg r$

What do we get?

A new formula with more propositions than the original one
NOT an equivalent formula

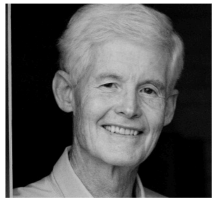
New formula is *satisfiable iff the original is satisfiable*
we call it *equisatisfiable*)

Size of resulting formula: *linear* in original size
good for use in satisfiability checking

The Boolean Satisfiability problem

A bit of history

Cook



Levin

Karp

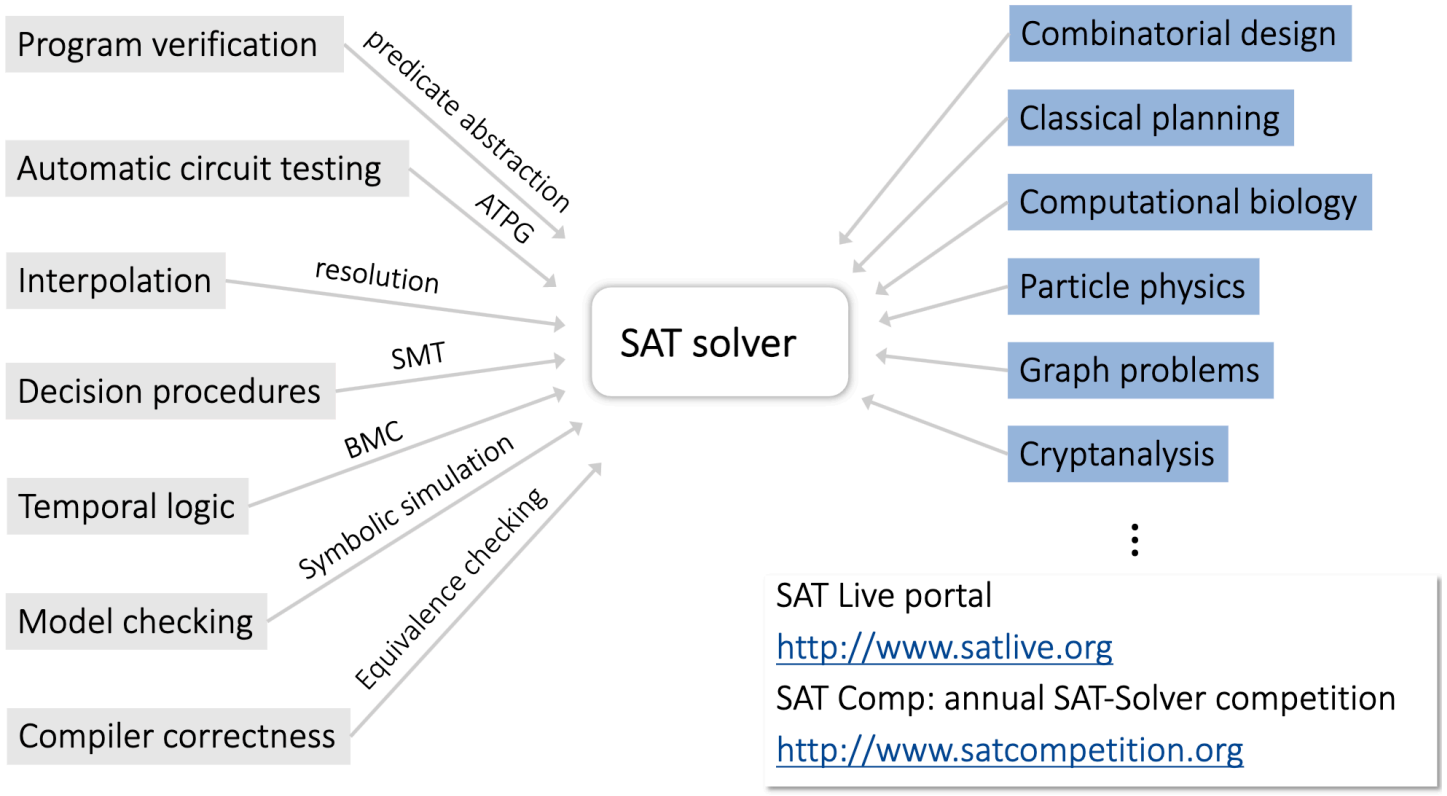
The SAT problem
For F in CNF, exists $I : I \vDash F$?

First NP-complete problem!

Cook-Levin Theorem:
SAT is NP-complete

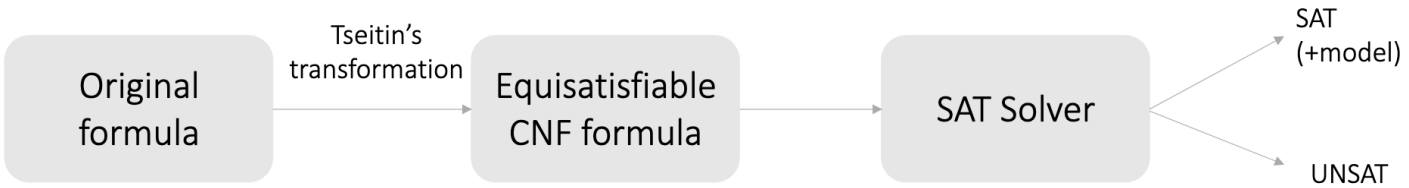
Cook, *The complexity of theorem proving procedures*, 1971

Karp, *Reducibility among combinatorial problems*, 1972


Program verification
Automatic circuit testing
Interpolation
Decision procedures
Temporal logic
Model checking
Compiler correctness
predicate abstraction
ATPG
resolution
SMT
BMC
Symbolic simulation
Equivalence checking
SAT solver
Combinatorial design
Classical planning
Computational biology
Particle physics
Graph problems
Cryptanalysis
SAT Live portal
http://www.satlive.org
SAT Comp: annual SAT-Solver competition
http://www.satcompetition.org

A Modern SAT Solver

A Modern SAT Solver



Almost all SAT solvers today are based on DPLL (Davis-Putnam-Logemann-Loveland)

These algorithms are also called "Decision Procedures"

History Again

1962: the original algorithm known as DP (Davis-Putnam)
⇒ “simple” procedure for automated theorem proving

Davis and Putnam hired two programmers, Logemann and Loveland, to implement their ideas on the IBM 704.

Not all of the original ideas worked out as planned
⇒ refined algorithm is what is known today as DPLL

DPLL Insight

Two distinct approaches for the Boolean satisfiability problem

- Search
  - Find satisfying assignment by searching through all possible assignments
  - Example: truth table
- Deduction
  - Deduce new facts from set of known facts, i.e, application of proof rules
  - Example: semantic argument method
- DPLL combines search and deduction in a very effective way!

- Deductive principle underlying DPLL is **propositional resolution**
- Resolution can only be applied to formulas in CNF
- SAT solvers convert formulas to CNF to be able to perform resolution