CS156: The Calculus of Computation

Zohar Manna

Winter 2010

It is reasonable to hope that the relationship between **computation** *and* **mathematical logic** *will be as fruitful in the next century as that between* **analysis** *and* **physics** *in the last. The development of this relationship demands a concern for both applications and mathematical elegance.*

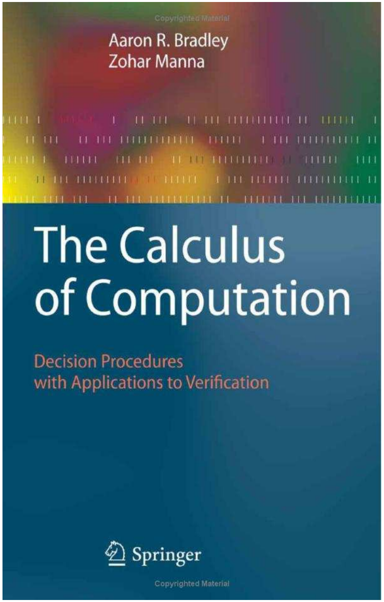
John McCarthy
A Basis for a Mathematical Theory of Computation, 1963

Textbook

The Calculus of Computation:
Decision Procedures with
Applications to Verification

by
Aaron Bradley
Zohar Manna

Springer 2007


Copyrighted Material
Aaron R. Bradley
Zohar Manna
The Calculus of Computation
Decision Procedures with Applications to Verification
Springer
Copyrighted Material

Topics: Overview

1. First-Order logic
2. Specification and verification
3. Satisfiability decision procedures

Part I: Foundations

1. Propositional Logic
2. First-Order Logic
3. First-Order Theories
4. Induction
5. Program Correctness: Mechanics
   Inductive assertion method, Ranking function method

Part II: Decision Procedures

7. Quantified Linear Arithmetic
   Quantifier elimination for integers and rationals
8. Quantifier-Free Linear Arithmetic
   Linear programming for rationals
9. Quantifier-Free Equality and Data Structures
10. Combining Decision Procedures
    Nelson-Oppen combination method
11. Arrays
    More than quantifier-free fragment

CS156: The Calculus of Computation

Zohar Manna

Winter 2010

Motivation

Motivation I

Decision Procedures are algorithms to decide formulae.
These formulae can arise

- in software verification.
- in hardware verification

Consider the following program:

```
for
  @ $\ell \leq i \leq u \land (rv \leftrightarrow \exists j.\, \ell \leq j < i \land a[j] = e)$
  (int $i := \ell$; $i \leq u$; $i := i + 1$) {
  if ($a[i] = e$) $rv :=$ true;
}
```

How can we decide whether the formula is a loop invariant?

Motivation II

Prove: (Path 1)

```
assume $\ell \leq i \leq u \land (rv \leftrightarrow \exists j.\ \ell \leq j < i \land a[j] = e)$
assume $i \leq u$
assume $a[i] = e$
$rv$ := true;
$i := i + 1$
@ $\ell \leq i \leq u \land (rv \leftrightarrow \exists j.\ \ell \leq j < i \land a[j] = e)$
```

Motivation III

Path 2:

```
assume $\ell \leq i \leq u \wedge (rv \leftrightarrow \exists j.\ \ell \leq j < i \wedge a[j] = e)$
assume $i \leq u$
assume $a[i] \neq e$
$i := i + 1$
@ $\ell \leq i \leq u \wedge (rv \leftrightarrow \exists j.\ \ell \leq j < i \wedge a[j] = e)$
```

Each path generates a Verification Condition (VC). We have to prove that each VC holds (valid).

Motivation IV

The VC for path 1 is computed by substitution:

```
assume $\ell \le i \le u \wedge (rv \leftrightarrow \exists j.\ \ell \le j < i \wedge a[j] = e)$
assume $i \le u$
assume $a[i] = e$
$rv$ := true;
$i := i + 1$
@ $\ell \le i \le u \wedge (rv \leftrightarrow \exists j.\ \ell \le j < i \wedge a[j] = e)$
```

Substituting $\top$ for rv and $i+1$ for i, the postcondition (denoted by the @ symbol) holds if and only if the VC:

$$\begin{aligned} &\ell \le i \le u \wedge (rv \leftrightarrow \exists j.\ \ell \le j < i \wedge a[j] = e) \wedge i \le u \wedge a[i] = e \\ \rightarrow\ &\ell \le i+1 \le u \wedge (\top \leftrightarrow \exists j.\ \ell \le j < i+1 \wedge a[j] = e) \end{aligned}$$

holds.

Motivation V

We need an algorithm that decides whether this formula holds. If the formula does not hold, the algorithm should give a counterexample; e.g.,

$$\ell = 0, i = 1, u = 1, rv = \mathit{false}, a[0] = 0, a[1] = 1, e = 1.$$

We will discuss such algorithms in later lectures.

CS156: The Calculus of Computation

Zohar Manna

Winter 2010

Chapter 1: Propositional Logic (PL)

Propositional Logic (PL)

PL Syntax

Atom	truth symbols $\top$ ("true") and $\bot$ ("false")
	propositional variables $P, Q, R, P_1, Q_1, R_1, \ldots$
Literal	atom α or its negation $\neg\alpha$
Formula	literal or application of a logical connective to formulae F, F_1, F_2

$\neg F$	"not"	(negation)
$F_1 \wedge F_2$	"and"	(conjunction)
$F_1 \vee F_2$	"or"	(disjunction)
$F_1 \rightarrow F_2$	"implies"	(implication)
$F_1 \leftrightarrow F_2$	"if and only if"	(iff)

Example:

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 of PL)

Formula 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

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

Example:

$F : P \land Q \;\rightarrow\; P \lor \neg Q$
$I : \{P \mapsto \text{true}, Q \mapsto \text{false}\}$ i.e., $I[P] = \text{true}, I[Q] = \text{false}$

P	Q	$\neg Q$	$P \land Q$	$P \lor \neg Q$	F
1	0	1	0	1	1

$1 = \text{true}$ $0 = \text{false}$

F evaluates to true under I; i.e., $I[F] = \text{true}$.

Inductive Definition of PL's Semantics

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

<u>Base Case</u>:

$I \models \top \qquad I \not\models \bot$

$I \models P$ iff $I[P] = \text{true}$; i.e., P is true under I
$I \not\models P$ iff $I[P] = \text{false}$

<u>Inductive Case</u>:

$I \models \neg F$ iff $I \not\models F$
$I \models F_1 \wedge F_2$ iff $I \models F_1$ and $I \models F_2$
$I \models F_1 \vee F_2$ iff $I \models F_1$ or $I \models F_2$ (or both)
$I \models F_1 \rightarrow F_2$ iff $I \models F_1$ implies $I \models F_2$
$I \models F_1 \leftrightarrow F_2$ iff, $I \models F_1$ and $I \models F_2$,
or $I \not\models F_1$ and $I \not\models F_2$

<u>Note</u>:

$I \models F_1 \rightarrow F_2$ iff $I \not\models F_1$ or $I \models F_2$.
$I \not\models F_1 \rightarrow F_2$ iff $I \models F_1$ and $I \not\models F_2$.
$I \not\models F_1 \vee F_2$ iff $I \not\models F_1$ and $I \not\models F_2$.

Example of Inductive Reasoning:

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

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

1.	I	$\models$	P	since $I[P] = \text{true}$	
2.	I	$\not\models$	Q	since $I[Q] = \text{false}$	
3.	I	$\models$	$\neg Q$	by 2 and $\neg$	
4.	I	$\not\models$	$P \wedge Q$	by 2 and $\wedge$	
5.	I	$\models$	$P \vee \neg Q$	by 1 and $\vee$	
6.	I	$\models$	F	by 4 and $\rightarrow$	Why?

Thus, F is true under I.

Note: steps 1, 3, and 5 are nonessential.

Satisfiability and Validity

F satisfiable iff there exists an interpretation I such that $I \models F$.
F valid iff for all interpretations I, $I \models F$.

$$\boxed{F \text{ is valid iff } \neg F \text{ is unsatisfiable}}$$

Goal: devise an algorithm to decide validity or unsatisfiability of formula F.

Method 1: Truth Tables

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

P	Q	$P \wedge Q$	$\neg Q$	$P \vee \neg Q$	F
0	0	0	1	1	1
0	1	0	0	0	1
1	0	0	1	1	1
1	1	1	0	1	1

Thus F is valid.

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

P	Q	$P \vee Q$	$P \wedge Q$	F	
0	0	0	0	1	$\leftarrow$ satisfying I
0	1	1	0	0	$\leftarrow$ falsifying I
1	0	1	0	0	
1	1	1	1	1	

Thus F is satisfiable, but invalid.

Method 2: Semantic Argument

- Assume F is not valid and I a falsifying interpretation:
  $I \not\models F$
- Apply proof rules.
- If no contradiction reached and no more rules applicable,
  F is invalid.
- If in every branch of proof a contradiction reached,
  F is valid.

Proof Rules for Semantic Arguments I

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

$$\frac{I \models F \wedge G}{\begin{array}{l} I \models F \\ I \models G \end{array}} \ \leftarrow\text{and} \qquad \frac{I \not\models F \wedge G}{I \not\models F \quad | \quad I \not\models G}$$

$$\nwarrow\text{or}$$

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

Proof Rules for Semantic Arguments II

$$\frac{I \models F \rightarrow G}{I \not\models F \quad | \quad I \models G}$$

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

$$\frac{I \models F \leftrightarrow G}{I \models F \wedge G \quad | \quad I \not\models F \vee G}$$

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

$$\frac{\begin{array}{l} I \models F \\ I \not\models F \end{array}}{I \models \bot}$$

Example: Prove

$$F: \; P \wedge Q \;\rightarrow\; P \vee \neg Q \quad \text{is valid.}$$

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	$\models$	$\bot$	4 and 5 are contradictory

Thus F is valid.

Example: Prove

$$F : (P \to Q) \land (Q \to R) \to (P \to R) \quad \text{is valid.}$$

Let's assume that F is not valid.

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

6.	$I \models P \rightarrow Q$	2 and $\wedge$
7.	$I \models Q \rightarrow R$	2 and $\wedge$
8a.	$I \not\models P$	6 and $\rightarrow$ (case a)
9a.	$I \models \bot$	4 and 8
8b.	$I \models Q$	6 and $\rightarrow$ (case b)
9ba.	$I \not\models Q$	7 and $\rightarrow$ (subcase ba)
10ba.	$I \models \bot$	8b and 9ba
9bb.	$I \models R$	7 and $\rightarrow$ (subcase bb)
10bb.	$I \models \bot$	5 and 9bb
9b.	$I \models \bot$	10ba and 10bb
8.	$I \models \bot$	9a and 9b

Our assumption is contradictory in all cases, so F is valid.

Example 3: Is

$$F: \; P \lor Q \;\rightarrow\; P \land Q$$

valid? Assume F is not valid:

1.	$I \not\models P \lor Q \;\rightarrow\; P \land Q$	assumption
2.	$I \models P \lor Q$	1 and $\rightarrow$
3.	$I \not\models P \land Q$	1 and $\rightarrow$
4a.	$I \models P$	2, $\lor$ (case a)
5aa.	$I \not\models P$	3, $\lor$ (subcase aa)
6aa.	$I \models \bot$	4a, 5aa
5ab.	$I \not\models Q$	3, $\lor$ (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.	$I \models \perp$	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):

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

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

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, that is, both $F_1 \rightarrow F_2$ and $F_2 \rightarrow F_1$ are valid.

F_1 entails F_2 ($F_1 \Rightarrow F_2$)

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

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

Example: Show

$$P \rightarrow Q \Leftrightarrow \neg P \vee Q$$

i.e.

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

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

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

$I \models P \rightarrow Q$

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

$I \not\models P \rightarrow Q$

Derive contradictions in both cases.

Normal Forms

1. Negation Normal Form (NNF)

   $\neg, \wedge, \vee$ are the only boolean connectives allowed.

   Negations may occur only in literals of the form $\neg P$.

   To transform F into equivalent F' in NNF, apply the following template equivalences recursively (and left-to-right):

$$\neg\neg F_1 \Leftrightarrow F_1 \qquad \neg\top \Leftrightarrow \bot \qquad \neg\bot \Leftrightarrow \top$$

$$\left.\begin{array}{l} \neg(F_1 \wedge F_2) \Leftrightarrow \neg F_1 \vee \neg F_2 \\ \neg(F_1 \vee F_2) \Leftrightarrow \neg F_1 \wedge \neg F_2 \end{array}\right\}\text{De Morgan's Law}$$

$$F_1 \rightarrow F_2 \Leftrightarrow \neg F_1 \vee F_2$$

$$F_1 \leftrightarrow F_2 \Leftrightarrow (F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1)$$

   "Complete" syntactic restriction: every F has an equivalent F' in NNF.

Example: Convert

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

to NNF.

$$\begin{aligned} F' &: \neg(\neg P \vee \neg(P \wedge Q)) && \rightarrow \\ F'' &: \neg\neg P \wedge \neg\neg(P \wedge Q) && \text{De Morgan's Law} \\ F''' &: P \wedge P \wedge Q && \neg\neg \end{aligned}$$

F''' is equivalent to F ($F''' \Leftrightarrow F$) and is in NNF.

2. Disjunctive Normal Form (DNF)

Disjunction of conjunctions of literals

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

To convert F into equivalent F' in DNF,
transform F into NNF and then
use the following template equivalences (left-to-right):

$$\left.\begin{array}{lcl} (F_1 \vee F_2) \wedge F_3 & \Leftrightarrow & (F_1 \wedge F_3) \vee (F_2 \wedge F_3) \\ F_1 \wedge (F_2 \vee F_3) & \Leftrightarrow & (F_1 \wedge F_2) \vee (F_1 \wedge F_3) \end{array}\right\} \textit{dist}$$

Note: formulae can grow exponentially as the distributivity laws are applied.

Example: Convert

$$F: \ (Q_1 \vee \neg\neg Q_2) \wedge (\neg R_1 \ \rightarrow \ R_2)$$

into equivalent DNF

$$\begin{aligned} F' &: (Q_1 \vee Q_2) \wedge (R_1 \vee R_2) && \text{in NNF} \\ F'' &: (Q_1 \wedge (R_1 \vee R_2)) \vee (Q_2 \wedge (R_1 \vee R_2)) && \text{dist} \\ F''' &: (Q_1 \wedge R_1) \vee (Q_1 \wedge R_2) \vee (Q_2 \wedge R_1) \vee (Q_2 \wedge R_2) && \text{dist} \end{aligned}$$

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

3. Conjunctive Normal Form (CNF)

Conjunction of disjunctions of literals

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

To convert F into equivalent F' in CNF,
transform F into NNF and then
use the following template equivalences (left-to-right):

$$\begin{array}{rcl} (F_1 \wedge F_2) \vee F_3 & \Leftrightarrow & (F_1 \vee F_3) \wedge (F_2 \vee F_3) \\ F_1 \vee (F_2 \wedge F_3) & \Leftrightarrow & (F_1 \vee F_2) \wedge (F_1 \vee F_3) \end{array}$$

A disjunction of literals is called a clause.

Example: Convert

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

to an equivalent formula F' in CNF.

First get rid of $\leftrightarrow$:

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

Now replace $\rightarrow$ with $\lor$:

$$F_2 : (\neg P \lor (\neg Q \lor R)) \land (\neg(\neg Q \lor R) \lor P)$$

Drop unnecessary parentheses and apply De Morgan's Law:

$$F_3 : (\neg P \lor \neg Q \lor R) \land ((\neg\neg Q \land \neg R) \lor P)$$

Simplify double negation (now in NNF):

$$F_4 : (\neg P \lor \neg Q \lor R) \land ((Q \land \neg R) \lor P)$$

Distribute disjunction over conjunction (now in CNF):

$$F' : (\neg P \lor \neg Q \lor R) \land (Q \lor P) \land (\neg R \lor P)$$

Equisatisfiability

Definition

F and F' are *equisatisfiable*, iff

$$F \text{ is satisfiable if and only if } F' \text{ is satisfiable}$$

Every formula is equisatifiable to either $\top$ or $\bot$.

Goal: Decide satisfiability of PL formula F

Step 1: Convert F to equisatisfiable formula F' in CNF

Step 2: Decide satisfiability of formula F' in CNF

Step 1: Convert F to equisatisfiable formula F' in CNF I

There is an *efficient conversion* of F to F' where

- F' is in CNF and
- F and F' are equisatisfiable

Note: efficient means polynomial in the size of F.

Basic Idea:

- Introduce a new variable P_G for every subformula G of F, unless G is already an atom.

Step 1: Convert F to equisatisfiable formula F' in CNF II

- For each subformula

  $$G : \; G_1 \circ G_2,$$

  produce a small formula

  $$P_G \;\leftrightarrow\; P_{G_1} \circ P_{G_2}.$$

  Here $\circ$ denotes an arbitrary connective ($\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$); if the connective is $\neg$, G_1 should be ignored.

Step 1: Convert F to equisatisfiable formula F' in CNF III

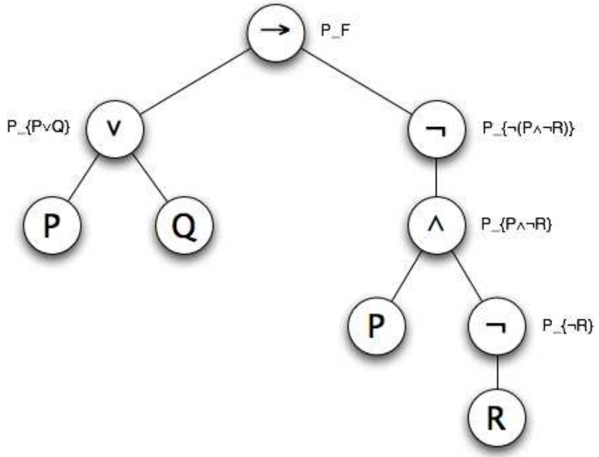


Figure: Parse tree for $F : P \lor Q \ \rightarrow \ \neg(P \land \neg R)$

Step 1: Convert F to equisatisfiable formula F' in CNF IV

- Convert each of these (small) formulae separately to an equivalent CNF formula

$$\mathrm{CNF}(P_G \leftrightarrow P_{G_1} \circ P_{G_2}) \ .$$

Let S_F be the set of all non-atom subformulae G of F (including F itself). The formula

$$P_F \wedge \bigwedge_{G \in S_F} CNF(P_G \leftrightarrow P_{G_1} \circ P_{G_2})$$

is equisatisfiable to F. (Why?)

The number of subformulae is linear in the size of F.
The time to convert one small formula is constant!

Example: CNF I

Convert

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

to an equisatisfiable formula in CNF.

Introduce new variables: P_F, $P_{P\vee Q}$, $P_{P\wedge\neg R}$, $P_{\neg R}$.

Create new formulae and convert them to equivalent formulae in CNF separately:

- $F_1 = \text{CNF}(P_F \ \leftrightarrow\ (P_{P\vee Q} \ \rightarrow\ P_{P\wedge\neg R}))$:

$$(\neg P_F \vee \neg P_{P\vee Q} \vee P_{P\wedge\neg R}) \wedge (P_F \vee P_{P\vee Q}) \wedge (P_F \vee \neg P_{P\wedge\neg R})$$

- $F_2 = \text{CNF}(P_{P\vee Q} \ \leftrightarrow\ P \vee Q)$:

$$(\neg P_{P\vee Q} \vee P \vee Q) \wedge (P_{P\vee Q} \vee \neg P) \wedge (P_{P\vee Q} \vee \neg Q)$$

Example: CNF II

- $F_3 = \text{CNF}(P_{P\wedge\neg R} \;\leftrightarrow\; P \wedge P_{\neg R})$:

$$(\neg P_{P\wedge\neg R} \vee P) \wedge (\neg P_{P\wedge\neg R} \vee P_{\neg R}) \wedge (P_{P\wedge\neg R} \vee \neg P \vee \neg P_{\neg R})$$

- $F_4 = \text{CNF}(P_{\neg R} \;\leftrightarrow\; \neg R)$:

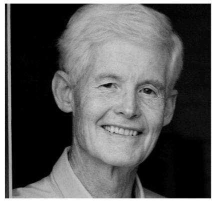
$$(\neg P_{\neg R} \vee \neg R) \wedge (P_{\neg R} \vee R)$$

$P_F \wedge F_1 \wedge F_2 \wedge F_3 \wedge F_4$ is in CNF and equisatisfiable to F.

The Boolean Satisfiability problem

A bit of history

Cook



Levin

Karp

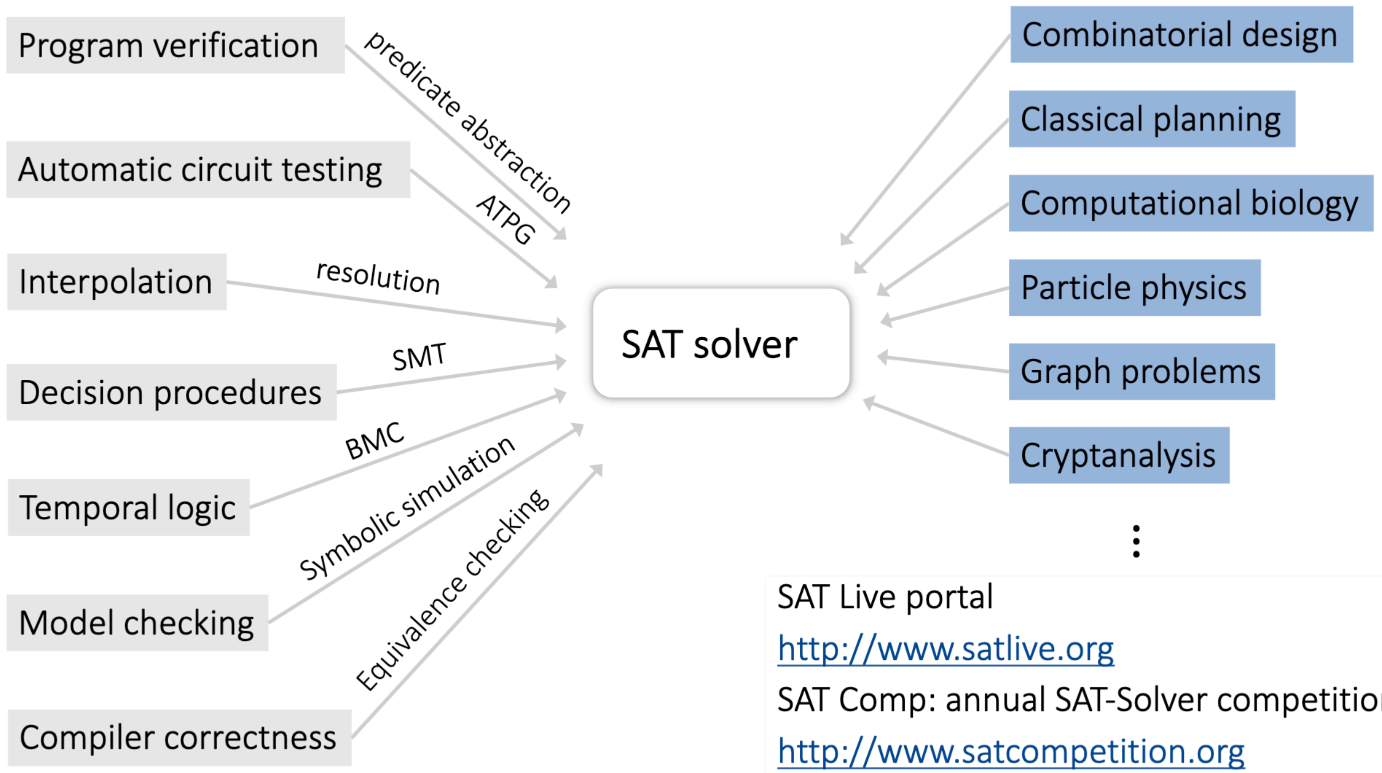
The SAT problem
For F in CNF, exists $I : I \models F$?

First NP-complete problem!

Cook-Levin Theorem:
SAT is NP-complete

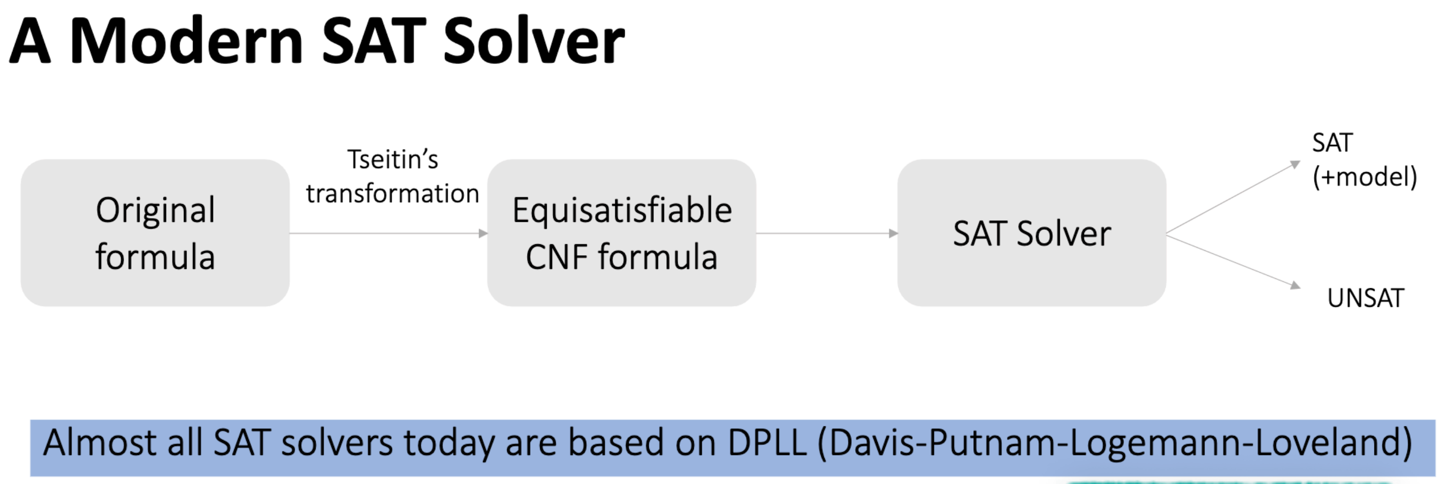
Cook, *The complexity of theorem proving procedures*, 1971

Karp, *Reducibility among combinatorial problems*, 1972



SAT Live portal
http://www.satlive.org
SAT Comp: annual SAT-Solver competition
http://www.satcompetition.org

A Modern SAT Solver

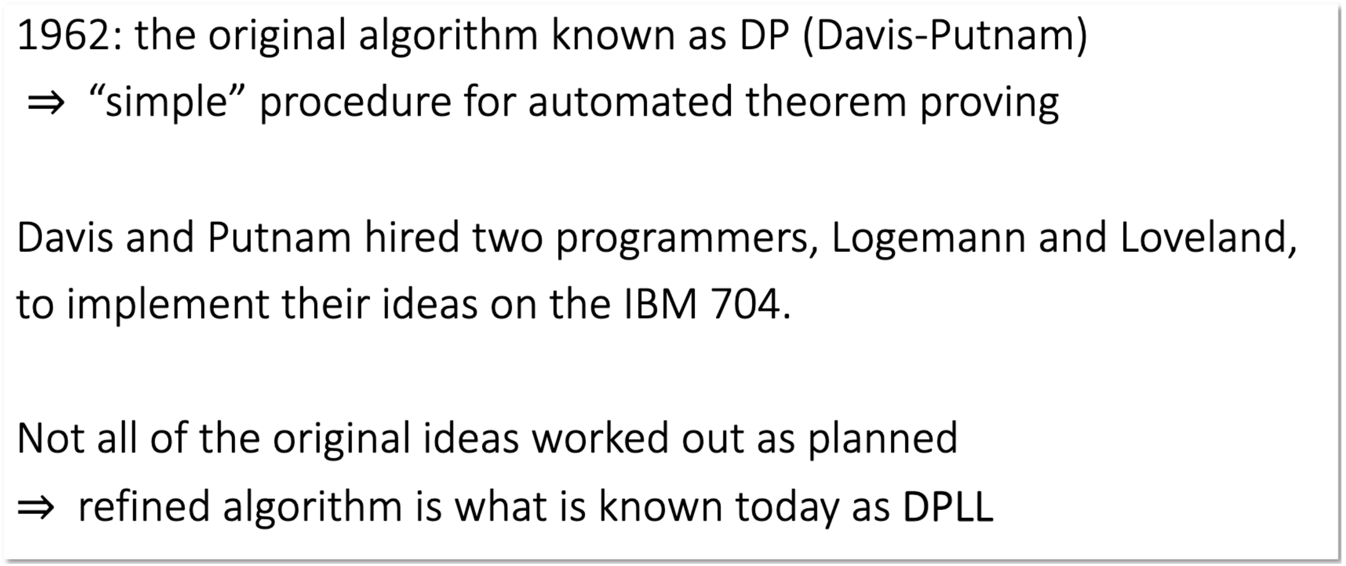


A Modern SAT Solver

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

Step 2: Decide the satisfiability of PL formula F' in CNF

Boolean Constraint Propagation (BCP)

If a clause contains one literal ℓ,

Set ℓ to $\top$:	$\cdots \wedge \overset{\top}{\cancel{\ell}} \wedge \cdots$
Remove all clauses containing ℓ:	$\cdots \wedge \cancel{(\cdots \vee \ell \vee \cdots)} \wedge \cdots$
Remove $\neg\ell$ in all clauses:	$\cdots \wedge (\cdots \vee \cancel{\neg\ell} \vee \cdots) \wedge \cdots$

based on the unit resolution

$$\frac{\ell \qquad \neg\ell \vee C}{C} \quad \leftarrow \text{clause}$$

Pure Literal Propagation (PLP)

If P occurs only positive (without negation), set it to $\top$.
If P occurs only negative set it to $\bot$.
Then do the simplifications as in Boolean Constraint Propagation

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

Decides the satisfiability of PL formulae in CNF

Decision Procedure DPLL: Given F in CNF

```
let rec DPLL $F$ =
  let $F'$ = BCP $F$ in
  let $F''$ = PLP $F'$ in
  if $F'' = \top$ then true
  else if $F'' = \bot$ then false
  else
     let $P$ = CHOOSE vars($F''$) in
     (DPLL $F''\{P \mapsto \top\}$) $\vee$ (DPLL $F''\{P \mapsto \bot\}$)
```

Simplification

Simplify according to the template equivalences (left-to-right) [exercise 1.2]

$$\begin{array}{ccc} \neg\bot \;\Leftrightarrow\; \top & \neg\top \;\Leftrightarrow\; \bot & \neg\neg F \;\Leftrightarrow\; F \\ F \land \top \;\Leftrightarrow\; F & F \land \bot \;\Leftrightarrow\; \bot & \cdots \\ F \lor \top \;\Leftrightarrow\; \top & F \lor \bot \;\Leftrightarrow\; F & \cdots \end{array}$$

Example I

Consider

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

<u>Branching on Q</u>

On the first branch, we have

$$F\{Q \mapsto \top\}: \ (R) \wedge (\neg R) \wedge (P \vee \neg R).$$

By unit resolution,

$$\frac{R \qquad (\neg R)}{\bot},$$

so $F\{Q \mapsto \top\} = \bot \Rightarrow$ false.

Example II

Recall

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

On the other branch, we have

$$F\{Q \ \mapsto \ \bot\}:\ (\neg P \vee R).$$

Furthermore, by PLP,

$$F\{Q \ \mapsto \ \bot,\ R \ \mapsto \ \top\} \ = \ \top \ \Rightarrow \ \text{true}$$

Thus F is satisfiable with satisfying interpretation

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

or

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

Example

$$F : (\neg P \lor Q \lor R) \land (\neg Q \lor R) \land (\neg Q \lor \neg R) \land (P \lor \neg Q \lor \neg R)$$

$$\begin{tikzcd}
& F \arrow[dl, dash, "Q \mapsto \top"] \arrow[dr, dash, "Q \mapsto \bot"'] & \\
(R) \land (\neg R) \land (P \lor \neg R) \arrow[d, dash] & & (\neg P \lor R) \arrow[d, dash, "R \mapsto \top"] \\
\bot & & \top
\end{tikzcd}$$

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

(No matter what P is)