# LECTURE 18

Rest of this course: Focus on SAT solving
Prove theoretical results related to algorithms for SAT solving, focusing on CDCL/resolution
[Except for guest lecture on Gröbner bases]

Let's start with an overview

The idea of resolution-based SAT solving is from 1960s
[DP60, DLL62, Robinson 65]

However, "modern" SAT solving took off when efficient clause learning discovered
[Silva & Sakallah '96, Bayardo & Schrag '97]
Gave rise to new SAT solvers
[Moskewicz et al '01] — zMMZZM, chaff
and later solvers

Want to develop rigorous understanding
- Why solvers so good
- What are limitations

Different approaches (incomplete, biased survey)

(a) Proof cplx
Capture power of SAT solvers in terms of pr/syst
[Beame, Kautz, Sabharwal 2004]
[Pipatsrisawat & Darwiche 2011] (conference 2009)*

*) Ingo to cover this paper

(6) Model solvers

↗ Idealized SAT solvers, typically using nondeterminism to apply techniques optimally.

[Same references.]

*) will cover today

[Atserias, Fichte & Thurley '11]* conf 09 avoids nondet, but gets more limited results

High-level:

(a) gives lower bounds on performance
E.g. never better than resolution
intractable even for perfect strategy choices

(b) gives upper bounds
There are (sometimes) good strategies to be found.

Nondeterminism in (6) is probably inherent, in view of [Alekhnovich-Razborov '08] (conf '01) saying that resolution probably is <u>not</u> automatizable.

Today: [AFT '11]

Overview: 1) Gives general, theoretical model of CDCL solvers, with certain assumptions that we will discuss (most are reasonable, one or two less realistic but probably inherent — see open problems later)

2) Show that in this model, if $W(F \vdash \bot) \leq W$, then CDCL probably runs in time $n^{O(W)}$ ($n$ = # vars)

Open problem: Is this tight? I.e., are there formulas s.t. $W(F \vdash \bot) = W$ but $\mathcal{L}(F \vdash \bot) \geq n^{W}$ (or $n^{W - O(1)}$, or $n^{\Omega(W)}$)?

Proof idea:

Suppose $\exists$ width-$w$ refutation $\{C_1, C_2, \ldots, C_L\}$

Would like to argue that CDCL solver learns clauses $C_i$ in refutation one by one. If so, finally learns contradiction at top level ⇒ done.

Does not quite work for [AFT '11]

Instead, learn enough to be able to imply/conclude each $C_i$

Key concept: Solver ABSORBS clauses $C_i$ one by one.

Follow-up work on CSP

[Jeavons & Petke 2010]

SAT solvers can efficiently simulate so-called local consistency techniques for general CSPs.

---

Set-up

$0$ = false  $1$ = true

$x^b$ literal s.t. $x = b$ makes it true.

STATE: — denoted $S, T$

sequence of assignments $(x_1 = b_1, x_2 = b_2, \ldots)$

all variables distinct

some marked as DECISIONS $x_i \overset{d}{=} b_i$

$x_i$ DECISION VARIABLE

Otherwise IMPLIED ASSIGNMENT

DECISION LEVEL of $x_i \overset{d}{=} b_i$ = #decisions in $j = 1, \ldots, i$

Identify state and its partial assignment

CDCL model

Current state $S$ (at start $= \emptyset$)

Current set of clauses $\mathcal{D}$ (at start $= F$)

4 modes

Default

Conflict : Learn clause

Unit : Unit Propagate

Decision: Pick variable to branch on

DEFAULT If $S$ sets $Vars(\mathbb{D})$ and satisfies $\mathbb{D}$,
output SAT & stop
else if $\bot \in \mathbb{D}|_S$
move to CONFLICT
else if unit clause in $\mathbb{D}|_S$
move to UNIT
else
move to DECISION

CONFLICT
Apply LEARNING SCHEME to learn new clause
$C$ and add to $\mathbb{D}$
If $C = \bot$, output UNSAT & stop
else
apply RESTART POLICY deciding one of
a) set $S = \emptyset$, move to DEFAULT (restart)
b) remove assignments from tail of $S$
until $C|_{S'} \neq \bot$, new state $S'$,
move to UNIT

UNIT For any $x^b \in \mathbb{D}|_S$, add $x = b$ to $S$
and move to DEFAULT

DECISION Apply DECISION STRATEGY to find
$x \overset{d}{=} b$, add this to $S$, move to DEFAULT

Learning scheme should always add
$C$ s.t. $\mathcal{D} \models C$, and $C|_S = \bot$,
and at most one variable of max decision level

$\Rightarrow$ Never learn same clause twice

$\Rightarrow$ Solver terminates (# clauses finite)

So-called ASSERTING CLAUSES satisfy this
[Zhang, Madigan, Moskewicz & Malik '01]
[Also recall Matti's lecture]

so in particular this is always possible.

Plain vanilla DPLL (or actually DLL):
- Never learn new clause
- Never restart
- BACKTRACKING
- Assignments removed up to latest decision assignment
  $x = 0$, which is replaced by $x = 1-0$.

Modern SAT solvers
- Do learn clauses
- Do restart (fairly frequently)
- Backtrack on literal as determined by learned clause
  NON-CHRONOLOGICAL BACKTRACKING

COMPLETE ROUND STARTED WITH $\mathbb{D}$

Start in default mode, $S = \emptyset$,
run until a clause is falsified or all variables set.
Represent as its sequence of states $S_0 = \emptyset, S_1, \ldots, S_m$

ROUND

Initial segment of complete round $S_0, \ldots, S_t$ up to state where either

a) $\bot \in \mathbb{D} \restriction_{S_t}$ ← conclusive round

b) No unit clause in $\mathbb{D} \restriction_{S_t}$ ← inconclusive round

(since no clause can be learned)

ASSUMPTIONS

RESTART POLICY Should dictate restarts often enough

Any bounded # conflicts in between restarts

Comment Some solvers restart very aggressively after almost each conflict — is OK

Popular sequence Luby 1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 1, 1, 2, 4, 8

seems to be excluded. $S_{-1}$

Assumption seems ~~OK~~ reasonable.

This lecture assume restarts after every conflict.

DECISION STRATEGY

No assumptions, except that constant fraction of (complex) rounds should have totally random decisions.

Comment: In theory, could implement this
Know of no SAT solver that does.
Fairly easy to argue this assumption is necessary

Assumption does not correspond to practice, but is not outrageous.

This lecture assume all decisions random.

LEARNING SCHEME

Assume that asserting clause is learned.
clearly OK assumption

More details

$S_0, \ldots, S_r$ conclusive round $x_i \overset{d}{=} 6_i$ ith ~~decision~~ assignment.

Define clauses $A_i$ by reverse induction

1) $A_{r+1}$ any clause $\in \mathbb{D}$ s.t $A_{r+1} \restriction_{S_r} = 1$
2) If $x_i \overset{d}{=} 6_i$ decision $A_r = A_{r+1}$
3) Else if $x_i = a_i$ implied fix some $B_i \in \mathbb{D}$
s.t. $B_i \restriction_{S_{i-1}} = x_i^{6_i}$

Let $A_i = Res(A_{i+1}, B_i, x)$ if resolvable
$A_i = A_{i+1}$ otherwise.

Call $A_i$ CONFLICT CLAUSES

Two examples

AFT-DECISION  Add $A_1$

Not hard to check that $A_1$ is asserting. Every literal in $A_1$ is negation of some decision literal in $S_\sigma$

This lecture assume for simplicity this is used

1UIP (First unique implication point)

Add $A_i$ s.t. $i \leq \sigma$ maximal subject to condition that $A_i$ asserting.

Learned clauses

No learned clauses are ever forgotten

Comment Clearly unrealistic — time-out in just a few seconds

Open problem Is this assumption necessary? Can it be relaxed? How much?

AFT memoir that in view of [BN '08, BN '11] probably assumption cannot be removed entirely

Open problem Since the clauses needed are all small width, would it be OK to learn only small-width clauses?

Not true in general, in view of
[Ben-Sasson & Johannsen '10]
But maybe if the proof is narrow... ?

## ANALYSIS OF THE CDCL ALGORITHM

$R, R'$ rounds (represented as sequences of states)

$\mathbb{D}, \mathbb{D}'$ sets of clauses

$A, B, C$ clauses

$R'$ SUBSUMES $R$ if $R \subseteq R'$ (viewed as partial truth value assignments ignoring decision marks) ($R'$ extends $R$ as a trail)

$R'$ & $R$ AGREE on $C$ if assign the same values to $Vars(C)$ (in $\{0, 1, *\}$)

$R$ BRANCHES in $C$ if all decision variables $\subseteq Vars(C)$

(Notions depend only on $Vars(C)$, but notationally convenient to use clauses)

Next: two important, but technical, lemmas
High level: "Inconclusive rounds are robust wrt order of assignments"

any inconclusive round subsumes any other round that agrees on its decision assignments.

LEMMA 1 (C any clause; $\mathcal{D} \subseteq \mathcal{D}'$, $R'$ inconclusive round started with $\mathcal{D}'$)
Suppose $\mathcal{D} \subseteq \mathcal{D}'$, $R'$ inconclusive round) ~~started with $\mathcal{D}$~~
Then $\forall$ $R$ started with $\mathcal{D}$ that branches in $C$ and agrees with $R'$ on $C$, $R'$ subsumes $R$.

Proof By induction over $R = (S_0, \ldots, S_t)$. Prove: Every assignment in $S_i$ made in $R'$. Base case empty. If $R$ makes decision, its on Vars(C) and agrees with $R'$ by assumption.
Implied assignment $x = b$. Then

$$\exists A \in \mathcal{D} \quad A|_{S_{i-1}} = x\,b$$

Then $R'$ has had to set $x = b$ since it is inconclusive (no conflict) and a round (no further unit prop possible). □

One can find an $R$ as in Lem 1 (i.e. lemma not vacuous)

LEMMA 2 (C any clause; $\mathcal{D} \subseteq \mathcal{D}'$, $R$ inconclusive started with $\mathcal{D}'$)
Then $\exists$ inconclusive $R$ started with $\mathcal{D}$ that branches in $C$, agrees with $R'$ on $C$, and is subsumed by $R$

(Skip proof to save time.)