The Proof-Search Problem (between bdd-width resolution and bdd-degree semi-algebraic proofs)

Albert Atserias
Universitat Politècnica de Catalunya
Barcelona, Spain

Satisfiability

Example:

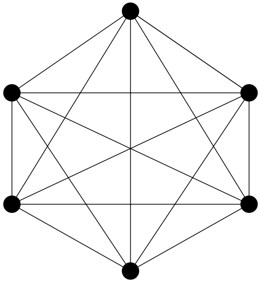
15 variables and 40 clauses

$x_1 \vee x_2 \vee x_6$	$x_1 \vee x_3 \vee x_7$	$x_1 \vee x_4 \vee x_8$	$x_1 \vee x_5 \vee x_9$
$x_2 \vee x_3 \vee x_{10}$	$x_2 \vee x_4 \vee x_{11}$	$x_2 \vee x_5 \vee x_{12}$	$x_3 \vee x_4 \vee x_{13}$
$x_3 \vee x_5 \vee x_{14}$	$x_4 \vee x_5 \vee x_{15}$	$x_6 \vee x_7 \vee x_{10}$	$x_6 \vee x_8 \vee x_{11}$
$x_6 \vee x_9 \vee x_{12}$	$x_7 \vee x_8 \vee x_{13}$	$x_7 \vee x_9 \vee x_{14}$	$x_8 \vee x_9 \vee x_{15}$
$x_{10} \vee x_{11} \vee x_{13}$	$x_{10} \vee x_{12} \vee x_{14}$	$x_{11} \vee x_{12} \vee x_{15}$	$x_{13} \vee x_{14} \vee x_{15}$
$\overline{x_1} \vee \overline{x_2} \vee \overline{x_6}$	$\overline{x_1} \vee \overline{x_3} \vee \overline{x_7}$	$\overline{x_1} \vee \overline{x_4} \vee \overline{x_8}$	$\overline{x_1} \vee \overline{x_5} \vee \overline{x_9}$
$\overline{x_2} \vee \overline{x_3} \vee \overline{x_{10}}$	$\overline{x_2} \vee \overline{x_4} \vee \overline{x_{11}}$	$\overline{x_2} \vee \overline{x_5} \vee \overline{x_{12}}$	$\overline{x_3} \vee \overline{x_4} \vee \overline{x_{13}}$
$\overline{x_3} \vee \overline{x_5} \vee \overline{x_{14}}$	$\overline{x_4} \vee \overline{x_5} \vee \overline{x_{15}}$	$\overline{x_6} \vee \overline{x_7} \vee \overline{x_{10}}$	$\overline{x_6} \vee \overline{x_8} \vee \overline{x_{11}}$
$\overline{x_6} \vee \overline{x_9} \vee \overline{x_{12}}$	$\overline{x_7} \vee \overline{x_8} \vee \overline{x_{13}}$	$\overline{x_7} \vee \overline{x_9} \vee \overline{x_{14}}$	$\overline{x_8} \vee \overline{x_9} \vee \overline{x_{15}}$
$\overline{x_{10}} \vee \overline{x_{11}} \vee \overline{x_{13}}$	$\overline{x_{10}} \vee \overline{x_{12}} \vee \overline{x_{14}}$	$\overline{x_{11}} \vee \overline{x_{12}} \vee \overline{x_{15}}$	$\overline{x_{13}} \vee \overline{x_{14}} \vee \overline{x_{15}}$

Satisfiability

Example:

$$R(3,3) \leq 6$$



In every party of six,
either three of them are mutual friends,
or three of them are mutual strangers.

Part I

PROPOSITIONAL PROOF COMPLEXITY

Proof systems

Definition:

A proof system for $A \subseteq \Sigma^*$ is a binary relation $R \subseteq \Sigma^* \times \Sigma^*$ s.t.:

- $x \in A \Rightarrow \exists y \in \Sigma^* \ \ ((x, y) \in R)$,
- $x \notin A \Rightarrow \forall y \in \Sigma^* \ \ ((x, y) \notin R)$,

and

- $(x, y) \overset{?}{\in} R$ decidable in time $\mathrm{poly}(|x| + |y|)$.

Proof systems

Terminology:

- If $(x, y) \in R$, then y is an R-proof that $x \in A$,

Proof systems

Terminology:

- If $(x, y) \in R$, then y is an R-proof that $x \in A$,
- For x in A, let $c_R(x) = \min\{|y| : y \text{ is an } R\text{-proof that } x \in A\}$.

Proof systems

Terminology:

- If $(x, y) \in R$, then y is an R-proof that $x \in A$,
- For x in A, let $c_R(x) = \min\{|y| : y$ is an R-proof that $x \in A\}$.

Definition:
A proof system R for A is polynomially-bounded if

$$c_R(x) \leq \mathrm{poly}(|x|),$$

for $x \in A$.

Polynomial simulation

Definition:
Given proof systems R_1 and R_2 for A,

$$R_1 \; \leq^p \; R_2$$

if there exist f computable in polynomial-time such that:

$$(x, y) \in R_1 \Rightarrow (x, f(y)) \in R_2.$$

Resolution and Frege Proof Systems

Cut rule (Resolution):

$$\frac{A \vee C \qquad B \vee \overline{C}}{A \vee B}.$$

Resolution and Frege Proof Systems

Cut rule (Resolution):

$$\frac{A \vee C \qquad B \vee \overline{C}}{A \vee B}.$$

Rest of rules of inference (Frege):

$$\frac{}{A \vee \overline{A}} \qquad \frac{A}{A \vee B} \qquad \frac{A \vee C \qquad B \vee D}{A \vee B \vee (C \wedge D)}.$$

Resolution and Frege Proof Systems

Cut rule (Resolution):

$$\frac{A \vee C \qquad B \vee \overline{C}}{A \vee B}.$$

Rest of rules of inference (Frege):

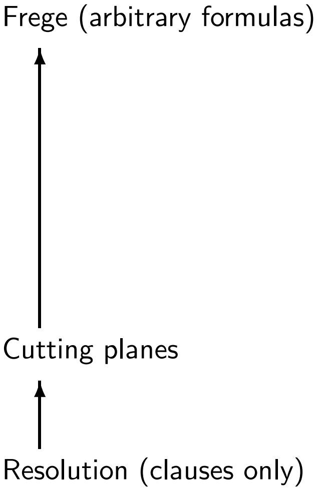
$$\frac{}{A \vee \overline{A}} \qquad \frac{A}{A \vee B} \qquad \frac{A \vee C \qquad B \vee D}{A \vee B \vee (C \wedge D)}.$$

Proof that $C_1 \wedge \ldots \wedge C_m \in \mathrm{UNSAT}$:

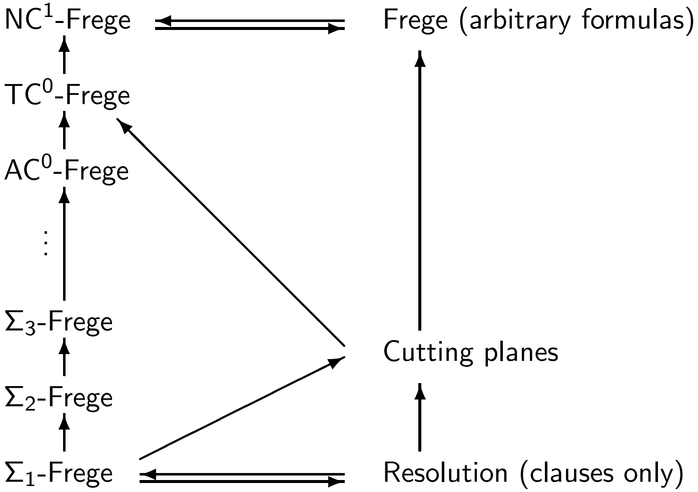
$$C_1, \ldots, C_m, F_1, \ldots, F_i, \ldots, F_j, \ldots, F_k, \ldots, \emptyset$$

Hierarchy of proof systems

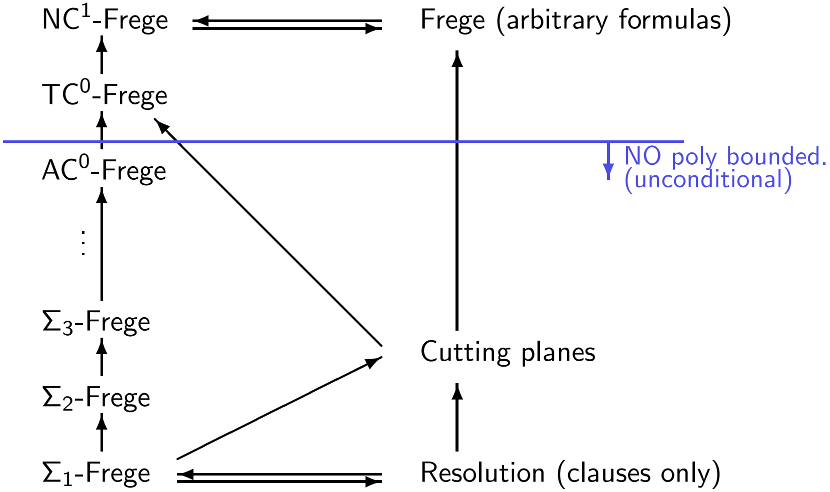
Hierarchy of proof systems



Hierarchy of proof systems



Hierarchy of proof systems



Proof search

Definition:
The proof search problem for a proof system R for A is:

Given $x \in A$,
find some $y \in \Sigma^*$ (any $y \in \Sigma^*$)
such that $(x, y) \in R$.

Proof search

Definition:
The proof search problem for a proof system R for A is:

$$\text{Given } x \in A, \\ \text{find some } y \in \Sigma^* \text{ (any } y \in \Sigma^*) \\ \text{such that } (x, y) \in R.$$

Definition [Bonet-Pitassi-Raz]:
A proof system R for A is automatizable if the proof search problem for R is solvable in time $\mathrm{poly}(|x| + c_R(x))$.

An easier task

Definition

The weak proof search problem for a proof system R for A is:

$$\text{Given } x \in \Sigma^* \text{ and a size parameter } s \in \mathbb{N},$$
$$\text{if } c_P(x) \leq s, \text{ say YES,}$$
$$\text{if } c_P(x) = \infty, \text{ say NO.}$$

An easier task

Definition
The weak proof search problem for a proof system R for A is:

$$\text{Given } x \in \Sigma^* \text{ and a size parameter } s \in \mathbb{N}, \\ \text{if } c_P(x) \le s, \text{ say YES,} \\ \text{if } c_P(x) = \infty, \text{ say NO.}$$

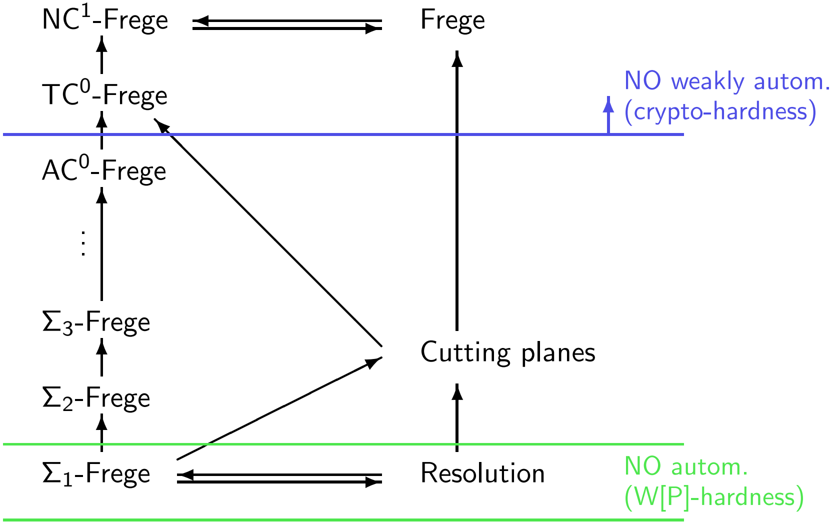
Definition [Razborov] [Pudlak]
A proof system R for A is weakly automatizable if the weak proof search problem for R is solvable in time $\mathrm{poly}(|x| + s)$.

Some known results

Theorems [Bonet-Pitassi-Raz] [Alekhnovich-Razborov]

1. Weak automatizability of Frege is crypto-hard.
2. Automatizability of Resolution is W[P]-hard.

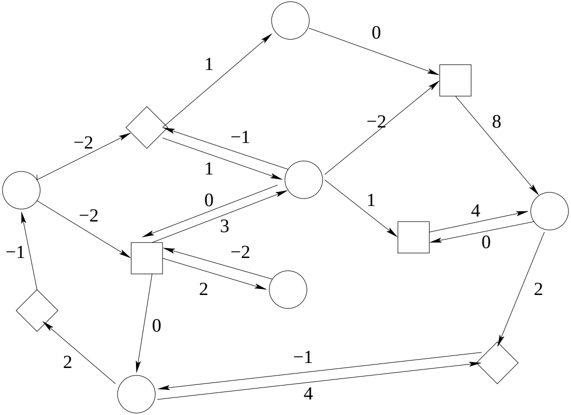
Status of the question



Part II

MEAN-PAYOFF STOCHASTIC GAMES

Mean-payoff games



Box: player max.
Diamond: player min.
Circle: random (nature).

Mean-payoff stochastic games

A mean-payoff stochastic game is given by:

- Game graph $G = (V, E)$: finite directed graph.
- Partition: $V = V_{\max} \cup V_{\min} \cup V_{\text{avg}}$.
- Weights on edges: $w : E \to \mathbb{Z}$.

Mean-payoff stochastic games

A mean-payoff stochastic game is given by:

- Game graph $G = (V, E)$: finite directed graph.
- Partition: $V = V_{\max} \cup V_{\min} \cup V_{\text{avg}}$.
- Weights on edges: $w : E \to \mathbb{Z}$.

Goals of players:

$$\max/\min \, \mathbb{E}\left[\lim_{t\to\infty} \frac{1}{t} \sum_{i=0}^{t} w(v_{i-1}, v_i)\right]$$

(simplifying issues: lim vs. lim sup or lim inf, measurability, etc.).

Four types of games

Mean-payoff stochastic games [Shapley 1953]:

No restrictions.

Simple stochastic games [Condon]:

All weights are 0 except at one $+1$-sink and one -1-sink.

Mean-payoff games [Ehrenfeucht-Mycielski]:

There are no random nodes.

Parity games [Emerson-Jutla]:

There are no random nodes and
all weights outgoing node i are $(-1)^i \cdot (|V|+1)^i$.

Complexity of the games

Definition

The MPSG-problem is:

Given a game graph,
does player max have a strategy
securing value ≥ 0?

Complexity of the games

Definition
The MPSG-problem is:

Given a game graph,
does player max have a strategy
securing value ≥ 0?

Theorem [C, EM, EJ, Zwick-Paterson]

1. PG $\leq_m^p$ MPG $\leq_m^p$ SSG $\leq_m^p$ MPSG.
2. All four versions are in NP $\cap$ co-NP.

Complexity of the games

Definition
The MPSG-problem is:

Given a game graph,
does player max have a strategy
securing value ≥ 0?

Theorem [C, EM, EJ, Zwick-Paterson]

1. PG $\leq_m^p$ MPG $\leq_m^p$ SSG $\leq_m^p$ MPSG.
2. All four versions are in NP $\cap$ co-NP.

Open problems

Membership in P is unknown.
Any kind of hardness is unknown.

Back to the proof-search problem

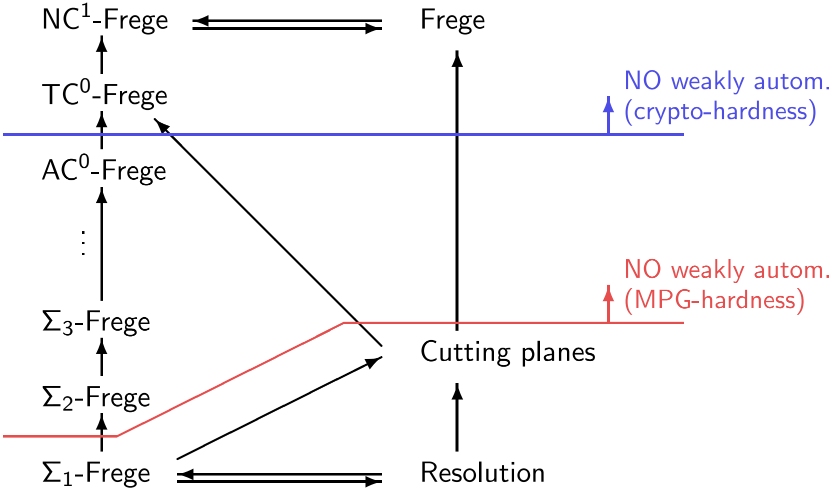
Theorem [A.-Maneva]
There is a polynomial time algorithm

$$\text{MPG instance } G \mapsto \text{CNF formula } F$$

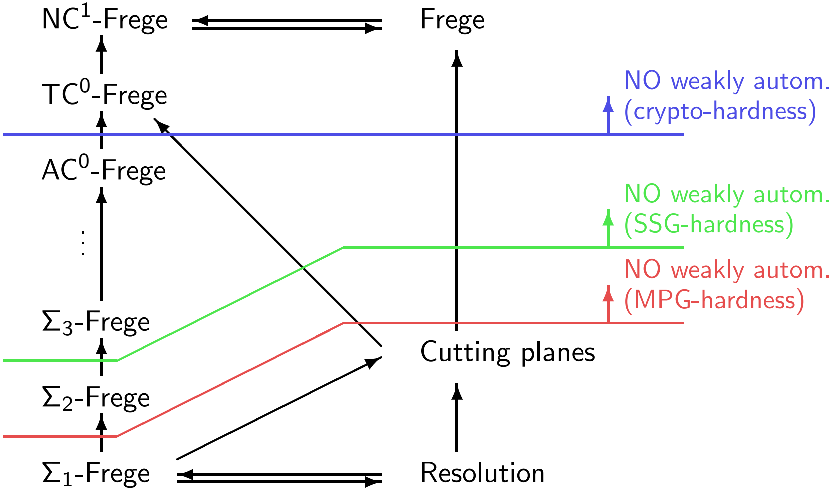
so that:

1. If max wins G, then F is satisfiable.
2. If min wins G, then F has poly-size Σ_2-refutation.

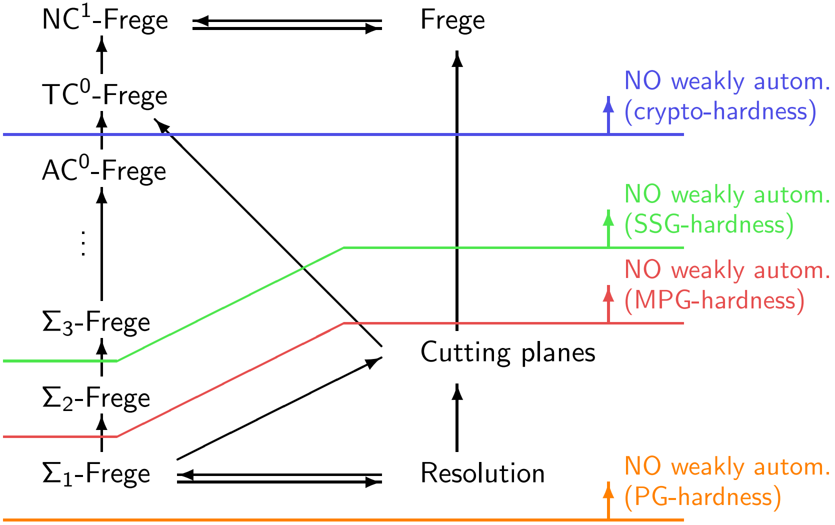
Status of the question



Status of the question



Status of the question



Part III

BOUNDED-WIDTH RESOLUTION

Bounded-width resolution

Definition

1. The width of a clause is its number of literals.
2. The width of a refutation is the width of its widest clause.

Bounded-width resolution

Definition

1. The width of a clause is its number of literals.
2. The width of a refutation is the width of its widest clause.

Facts

1. The number of clauses of width at most k is $O(n^k)$.
2. If F has a refutation of width k, then it has one of size $O(n^k)$.

Facts

1. Width-2 resolution is complete for 2CNFs.
2. Width-k resolution is complete for CNFs of tree-width $< k$.
3. Bounded-width resolution simulates typical constraint propagation techniques.

Some structure

Theorem [Ben-Sasson-Wigderson]
If an n-variable 3-CNF formula has a resolution refutation of size s, then it also has one of width $O(\sqrt{n \log s})$.

Some structure

Theorem [Ben-Sasson-Wigderson]
If an n-variable 3-CNF formula has a resolution refutation of size s, then it also has one of width $O(\sqrt{n \log s})$.

Corollary

The proof-search problem for resolution for n-variable 3CNFs
can be solved in time $n^{O(\sqrt{n \log s})}$,
where s is the smallest refutation-size.

Note:

If $s = \mathrm{poly}(n)$, this is subexponential of type $2^{n^{0.51}}$

Bounded-width proofs and SAT-solving

Question:

How do state-of-the-art SAT-solvers compare to bounded-width?

Bounded-width proofs and SAT-solving

Question:

How do state-of-the-art SAT-solvers compare to bounded-width?

Rest of this section [A.-Fichte-Thurley]

If CDCL is allowed enough random decisions and restarts,
then it simulates width-k resolution in time $O(n^{2k})$ w.h.p.

CDCL Algorithms

Algorithm A:

Let α be the empty list

DEFAULT:
if α satisfies F: return YES
if α falsifies F: go to CONFLICT
if $F|_\alpha$ contains a unit-clause: go to UNIT
go to DECIDE

CDCL Algorithms

Algorithm A:

Let α be the empty list

DEFAULT:

if α satisfies F: return YES
if α falsifies F: go to CONFLICT
if $F|_\alpha$ contains a unit-clause: go to UNIT
go to DECIDE

UNIT:

choose unit-clause x^a in $F|_\alpha$
append $x = a$ to α, go to DEFAULT

CDCL Algorithms

Algorithm A:

Let α be the empty list

DEFAULT:

if α satisfies F: return YES
if α falsifies F: go to CONFLICT
if $F|_\alpha$ contains a unit-clause: go to UNIT
go to DECIDE

UNIT:

choose unit-clause x^a in $F|_\alpha$
append $x = a$ to α, go to DEFAULT

DECIDE:

choose x in $V \setminus \mathrm{Dom}(\alpha)$ and a in $\{0, 1\}$
append $x \stackrel{\mathrm{d}}{=} a$ to α, go to DEFAULT

CDCL Algorithms (continued)

Algorithm A:

CONFLICT:

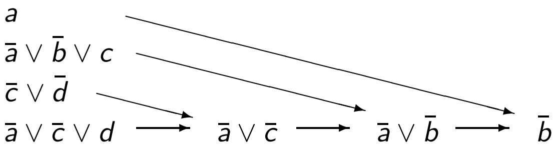
add new C to F with $F \models C$ and $C|_\alpha = \emptyset$
if C is the empty clause: return NO
remove assignments from the tail of α while $C|_\alpha = \emptyset$
go to DEFAULT

How is the new clause found?

Example:

$$F = a \wedge (\bar{a} \vee \bar{b} \vee c) \wedge (\bar{c} \vee \bar{d}) \wedge (\bar{a} \vee \bar{c} \vee d)$$

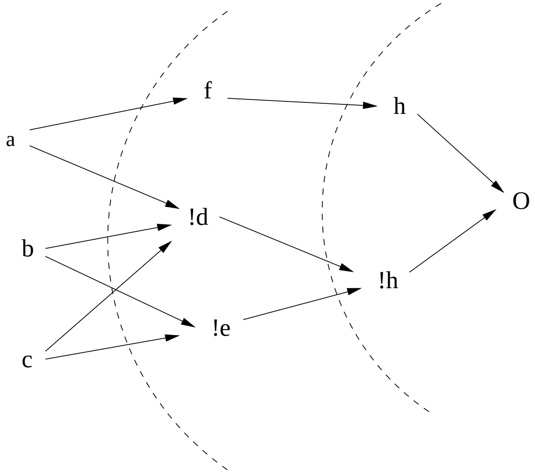
UNIT:	$a = 1$	due to a
DECIDE:	$b \stackrel{d}{=} 1$	choice
UNIT:	$c = 1$	due to $\bar{a} \vee \bar{b} \vee c$
UNIT:	$d = 0$	due to $\bar{c} \vee \bar{d}$
CONFLICT:		due to $\bar{a} \vee \bar{c} \vee d$.



Add (or learn) $\bar{b}$.

How is the new clause found?

Cuts in a conflict graph:



Add "occasional" restarts

Algorithm A:

CONFLICT:

add new C to F with $F \models C$ and $C|_\alpha = \emptyset$

if C is the empty clause: return NO

choose whether to restart (with current F) or continue

remove assignments from the tail of α while $C|_\alpha = \emptyset$

go to DEFAULT

Add “systematic” restarts

Algorithm A:

CONFLICT:

add new C to F with $F \models C$ and $C|_\alpha = \emptyset$
if C is the empty clause: return NO
restart (with current F)

Choice strategy under analysis

Learning scheme:

- Any asserting scheme [Marques-Silva-Sakallah].
- Particular case: DECISION scheme, 1UIP scheme, etc.

Restart policy:

- Any policy that allows any controlled number of conflicts between restarts.
- Particular case: restart at every conflict.

Decision strategy:

- Any strategy that allows a controlled number of rounds of arbitrary decisions between rounds of totally random ones.
- Particular case: totally random decisions all the time.

Rounds of the algorithm

A round is a sequence

$$\text{UNIT}^*(,\text{DECIDE},\text{UNIT}^*)^*$$

where each UNIT^* is maximal.

Rounds of the algorithm

A round is a sequence

$$\text{UNIT}^*(, \text{DECIDE}, \text{UNIT}^*)^*$$

where each UNIT^* is maximal.

A conclusive round is one where CONFLICT would be next.

Rounds of the algorithm

A round is a sequence

$$\text{UNIT}^*(, \text{DECIDE}, \text{UNIT}^*)^*$$

where each UNIT* is maximal.

A conclusive round is one where CONFLICT would be next.
An inconclusive round is one where CONFLICT would not be next.

Clause absorption

Let F be a set of clauses.
Let C be a clause.
Let R be an inconclusive round started with F.

Clause absorption

Let F be a set of clauses.
Let C be a clause.
Let R be an inconclusive round started with F.

Fact

If C belongs to F and R falsifies all literals of C but one,
then R satisfies the remaining one.

Clause absorption

Let F be a set of clauses.
Let C be a clause.
Let R be an inconclusive round started with F.

Fact

If C belongs to F and R falsifies all literals of C but one, then R satisfies the remaining one.

Definition

F absorbs C if every inconclusive round that falsifies all literals of C but one, satisfies the remaining one.

Example and non-example

Let

$$F = (a \vee \bar{b}) \wedge (b \vee c) \wedge (\bar{a} \vee \bar{b} \vee d \vee e).$$

Example and non-example

Let

$$F = (a \vee \bar{b}) \wedge (b \vee c) \wedge (\bar{a} \vee \bar{b} \vee d \vee e).$$

Note: F absorbs $a \vee c$. Why?

Example and non-example

Let

$$F = (a \vee \bar{b}) \wedge (b \vee c) \wedge (\bar{a} \vee \bar{b} \vee d \vee e).$$

Note: F absorbs $a \vee c$. Why?
$a = 0$ implies $b = 0$ and $b = 0$ implies $c = 1$, by UNIT;

Example and non-example

Let

$$F = (a \vee \bar{b}) \wedge (b \vee c) \wedge (\bar{a} \vee \bar{b} \vee d \vee e).$$

Note: F absorbs $a \vee c$. Why?
$a = 0$ implies $b = 0$ and $b = 0$ implies $c = 1$, by UNIT;
$c = 0$ implies $b = 1$ and $b = 1$ implies $a = 1$, by UNIT.

Example and non-example

Let

$$F = (a \vee \bar{b}) \wedge (b \vee c) \wedge (\bar{a} \vee \bar{b} \vee d \vee e).$$

Note: F absorbs $a \vee c$. Why?
$a = 0$ implies $b = 0$ and $b = 0$ implies $c = 1$, by UNIT;
$c = 0$ implies $b = 1$ and $b = 1$ implies $a = 1$, by UNIT.

Note: F does not absorb $\bar{b} \vee d \vee e$. Why?

Example and non-example

Let

$$F = (a \vee \bar{b}) \wedge (b \vee c) \wedge (\bar{a} \vee \bar{b} \vee d \vee e).$$

Note: F absorbs $a \vee c$. Why?
$a = 0$ implies $b = 0$ and $b = 0$ implies $c = 1$, by UNIT;
$c = 0$ implies $b = 1$ and $b = 1$ implies $a = 1$, by UNIT.

Note: F does not absorb $\bar{b} \vee d \vee e$. Why?
Look at the inconclusive round $e \overset{\mathrm{d}}{=} 0, d \overset{\mathrm{d}}{=} 0$.

Example and non-example

Let

$$F = (a \vee \bar{b}) \wedge (b \vee c) \wedge (\bar{a} \vee \bar{b} \vee d \vee e).$$

Note: F absorbs $a \vee c$. Why?
$a = 0$ implies $b = 0$ and $b = 0$ implies $c = 1$, by UNIT;
$c = 0$ implies $b = 1$ and $b = 1$ implies $a = 1$, by UNIT.

Note: F does not absorb $\bar{b} \vee d \vee e$. Why?
Look at the inconclusive round $e \stackrel{\text{d}}{=} 0, d \stackrel{\text{d}}{=} 0$.

Note: Both $F \models a \vee c$ and $F \models \bar{b} \vee d \vee e$. Why?

Example and non-example

Let

$$F = (a \vee \bar{b}) \wedge (b \vee c) \wedge (\bar{a} \vee \bar{b} \vee d \vee e).$$

Note: F absorbs $a \vee c$. Why?
$a = 0$ implies $b = 0$ and $b = 0$ implies $c = 1$, by UNIT;
$c = 0$ implies $b = 1$ and $b = 1$ implies $a = 1$, by UNIT.

Note: F does not absorb $\bar{b} \vee d \vee e$. Why?
Look at the inconclusive round $e \overset{\text{d}}{=} 0, d \overset{\text{d}}{=} 0$.

Note: Both $F \models a \vee c$ and $F \models \bar{b} \vee d \vee e$. Why?
Resolve 1st and 2nd, and 1st and 3rd, respectively.

Key properties of absorption

Logical consequence:
If F absorbs C, then $F \models C$.

Key properties of absorption

Logical consequence:
If F absorbs C, then $F \models C$.

Contradiction:
If F absorbs x and $\neg x$,
then any round started with F yields a conflict without decisions.

Key properties of absorption

Logical consequence:
If F absorbs C, then $F \models C$.

Contradiction:
If F absorbs x and $\neg x$,
then any round started with F yields a conflict without decisions.

Monotonicity:

- if $C \in F$, then F absorbs C,
- if $F \subseteq G$ and F absorbs C, then G absorbs C,
- if $C \subseteq D$ and F absorbs C, then F absorbs D.

Non-absorbed resolvents

Let F be a CNF-formula with n variables.
Let $\frac{A \quad B}{C}$ be a valid resolution inference; C non-empty.

Non-absorbed resolvents

Let F be a CNF-formula with n variables.
Let $\frac{A \quad B}{C}$ be a valid resolution inference; C non-empty.

Lemma (for DECISION learning scheme)

If F absorbs A and B, but not C,
then there exists a round R started with F such that:

1. *R is conclusive and learns a clause C' with $C' \subseteq C$,*
2. *R makes at most $|C|$ decisions.*

Non-absorbed resolvents

Let F be a CNF-formula with n variables.
Let $\frac{A \quad B}{C}$ be a valid resolution inference; C non-empty.

Lemma (for DECISION learning scheme)

If F absorbs A and B, but not C,
then there exists a round R started with F such that:

1. *R is conclusive and learns a clause C' with $C' \subseteq C$,*
2. *R makes at most $|C|$ decisions.*

Interpretation of 1:

When R happens, C becomes absorbed.

Intrepretation of 2:

R has probability $\Omega\left(\frac{1}{(2n)^{|C|}}\right)$ of happening.

Bottom-line (for DECISION scheme only)

Theorem (A.-Fichte-Thurley)

If F has a resolution refutation of width k,
then the algorithm learns the empty clause after $O(n^{2k})$ restarts,
with probability at least 0.99.

Theorem (AFT, Pipatsriwasat-Darwiche)

If F has a resolution refutation of length m,
then there exist choices to learn the empty clause after $O(m)$ restarts.

Part IV

BOUNDED-DEGREE SEMI-ALGEBRAIC PROOFS

Linear Programming

Formulation:

$$
\begin{aligned}
\min \quad & c_1 x_1 + \cdots + c_n x_n \\
\text{s.t.} \quad & a_{11} x_1 + \cdots + a_{1n} x_n \geq b_1 \\
& \vdots \\
& a_{m1} x_1 + \cdots + a_{mn} x_n \geq b_m \\
& x_1, \ldots, x_n \in \mathbb{R}
\end{aligned}
$$

Linear Programming

Formulation:

$$
\begin{array}{ll}
\min & c_1 x_1 + \cdots + c_n x_n \\
\text{s.t.} & a_{11} x_1 + \cdots + a_{1n} x_n \geq b_1 \\
 & \quad \vdots \\
 & a_{m1} x_1 + \cdots + a_{mn} x_n \geq b_m \\
 & x_1, \ldots, x_n \in \mathbb{R}
\end{array}
$$

Shorter form:

$$
\begin{array}{ll}
\min & c^{\mathrm{T}} x \\
\text{s.t.} & Ax \geq b \\
 & x \in \mathbb{R}^n
\end{array}
$$

Proof of Optimality for LP

Duality theorem:

$$\begin{array}{llcll} \min & c^{\mathrm{T}}x & = & \max & y^{\mathrm{T}}b \\ \text{s.t.} & Ax \geq b & & \text{s.t.} & y^{\mathrm{T}}A = c^{\mathrm{T}} \\ & x \in \mathbb{R}^n & & & y \geq 0 \\ & & & & y \in \mathbb{R}^m \end{array}$$

Proof of Optimality for LP

Duality theorem:

$$\begin{array}{llcll} \min & c^{\mathrm{T}}x & = & \max & y^{\mathrm{T}}b \\ \text{s.t.} & Ax \geq b & & \text{s.t.} & y^{\mathrm{T}}A = c^{\mathrm{T}} \\ & x \in \mathbb{R}^n & & & y \geq 0 \\ & & & & y \in \mathbb{R}^m \end{array}$$

Proof system version:

Use

$$\frac{a_i^{\mathrm{T}}x \geq b_i \qquad a_j^{\mathrm{T}}x \geq b_j}{y_i a_i^{\mathrm{T}}x + y_j a_j^{\mathrm{T}}x \geq y_i b_i + y_j b_j} \quad [y_i \geq 0, y_j \geq 0]$$

to derive

$$y^{\mathrm{T}}Ax \geq y^{\mathrm{T}}b$$

Adding Integrality 0-1 Constraints

Chvátal-Gomory cuts (cutting planes):

$$\frac{a_1 x + \cdots + a_n x \geq b}{a_1 x + \cdots + a_n x \geq \lceil b \rceil} \quad [a_1, \ldots, a_n \in \mathbb{Z}]$$

Adding Integrality 0-1 Constraints

Chvátal-Gomory cuts (cutting planes):

$$\frac{a_1 x + \cdots + a_n x \geq b}{a_1 x + \cdots + a_n x \geq \lceil b \rceil} \quad [a_1, \ldots, a_n \in \mathbb{Z}]$$

Semi-algebraic proofs:

$$\frac{}{x_i \geq 0} \qquad \frac{}{1 - x_i \geq 0} \qquad \frac{}{x_i^2 - x_i \geq 0} \qquad \frac{}{x_i - x_i^2 \geq 0}$$

Adding Integrality 0-1 Constraints

Chvátal-Gomory cuts (cutting planes):

$$\frac{a_1 x + \cdots + a_n x \geq b}{a_1 x + \cdots + a_n x \geq \lceil b \rceil} \quad [a_1, \ldots, a_n \in \mathbb{Z}]$$

Semi-algebraic proofs:

$$\frac{}{x_i \geq 0} \qquad \frac{}{1 - x_i \geq 0} \qquad \frac{}{x_i^2 - x_i \geq 0} \qquad \frac{}{x_i - x_i^2 \geq 0}$$

$$\frac{P \geq 0 \quad Q \geq 0}{\lambda P + \mu Q \geq 0} \qquad \frac{P \geq 0 \quad Q \geq 0}{PQ \geq 0} \qquad \frac{}{P^2 \geq 0}$$

Lift and Project Methods

Lovász-Schrijver/Sherali-Adams lift-and-project methods:

$$\frac{}{x_i \geq 0} \qquad \frac{}{1 - x_i \geq 0} \qquad \frac{}{x_i^2 - x_i \geq 0} \qquad \frac{}{x_i - x_i^2 \geq 0}$$

$$\frac{P \geq 0 \quad Q \geq 0}{\lambda P + \mu Q \geq 0} \qquad \frac{P \geq 0}{P \cdot x_i \geq 0} \qquad \frac{P \geq 0}{P \cdot (1 - x_i) \geq 0} \qquad \left(\frac{}{P^2 \geq 0} \right)$$

Bounded-rank/Bounded-degree Proofs

Definition:

1. Rank of an SA-proof is the maximum number of liftings in a path from the hypotheses to the conclusion.
2. Degree of an SA-proof is the maximum algebraic degree of any of its polynomials.

Bounded-rank/Bounded-degree Proofs

Definition:

1. Rank of an SA-proof is the maximum number of liftings in a path from the hypotheses to the conclusion.
2. Degree of an SA-proof is the maximum algebraic degree of any of its polynomials.

Facts:

1. Existence of rank-k SA-refutations in time $n^{O(k)}$.
2. Degree-k SA simulates width-k resolution.
3. Degree-k SA simulates Gaussian elimination for k-XOR-SAT.

Gaussian Elimination for k-XOR-SAT

Main tool [Grigoriev-Hirsch-Pasechnik]:

If c is an integer and $L(x) = \sum_i a_i x_i$ with integer a_i, then

$$(L(x) - c)(L(x) - c + 1) \geq 0$$

has short SA proofs of constant degree.

Gaussian Elimination for k-XOR-SAT

Main tool [Grigoriev-Hirsch-Pasechnik]:

If c is an integer and $L(x) = \sum_i a_i x_i$ with integer a_i, then

$$(L(x) - c)(L(x) - c + 1) \geq 0$$

has short SA proofs of constant degree.

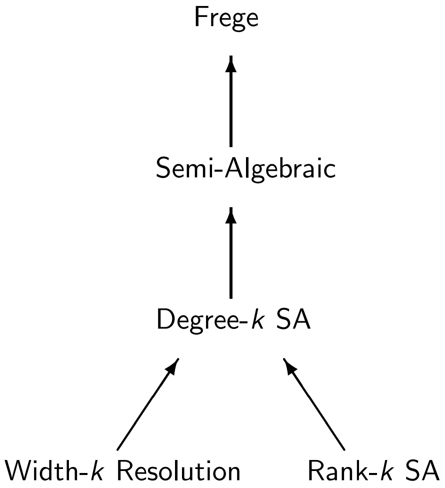
Expressing "evenness":

If $L(x) = \sum_i a_i x_i$ with integer a_i, then $L(x)$ is even iff

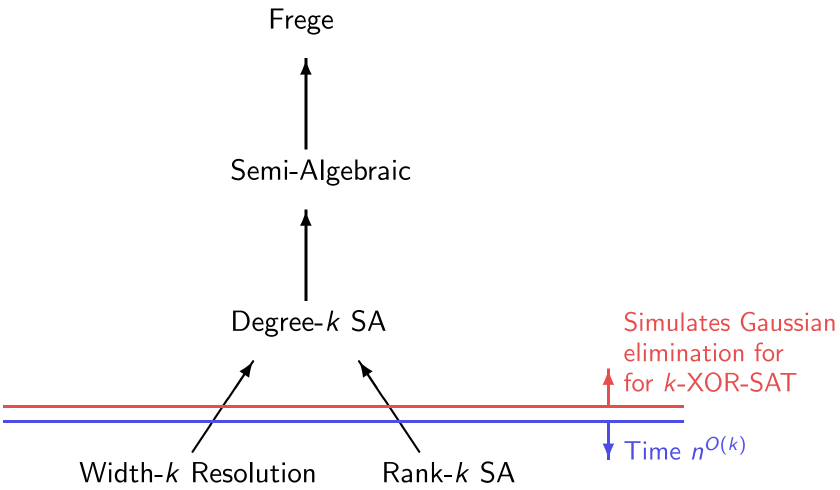
$$\begin{aligned}
\left(\tfrac{1}{2}L(x) - M\right)\left(\tfrac{1}{2}L(x) - M + 1\right) &\geq 0 \\
\left(\tfrac{1}{2}L(x) - M + 1\right)\left(\tfrac{1}{2}L(x) - M + 2\right) &\geq 0 \\
&\vdots \\
\left(\tfrac{1}{2}L(x) + M - 1\right)\left(\tfrac{1}{2}L(x) + M\right) &\geq 0
\end{aligned}$$

for $M = \sum_i a_i$, an upper bound on $|\frac{1}{2}L(x)|$.

Hierarchy “width-restricted” proof systems



Hierarchy "width-restricted" proof systems



Part V

CONCLUDING REMARKS

Two-sentence summary

Proof search problem for resolution and above:

At least as hard as parity games
(a notorious $>$ 20-year-old unsolved problem).

Bounded-width vs SAT-solvers:

Under mild conditions, CDCL algorithms behave (in principle)
at least as good as bounded-width resolution.

Semi-Algebraic proof systems:

Interesting "new" algorithms for proof-search (LP-based).
Surprising power of bounded-degree version (Gaussian elimination).