On CDCL-based Proof Systems with the Ordered Decision Strategy

Nathan Mull[1], Shuo Pang[2], Alexander Razborov[3]

July 6-8, 2020

[1] University of Chicago, Department of Computer Science
[2] University of Chicago, Department of Mathematics
[3] University of Chicago, USA and Steklov Mathematical Institute, Moscow, Russia

Background

Theorem (Pipatsrisawat and Darwiche, 2011)

(Informal) CDCL SAT solvers with the nondeterministic decision strategy can simulate general resolution.

[Beame et al., 2004] were the first to study the relationship between CDCL and the resolution proof system. The [Pipatsrisawat and Darwiche, 2011] result is arguably the first to consider a model that fairly closely resembles actual solver implementations.

Question. How much does this result depend on the nondeterminism in the decision strategy? For what other decision strategies does this kind of result hold?

Other Decision Strategies

Theorem (Asterias, Fichte, and Thurley, 2011)

*(Informal) CDCL SAT solvers with the **random** decision strategy can simulate **bounded-width** resolution.*

The [Pipatsrisawat and Darwiche, 2011] and
[Atserias et al., 2011] results were obtained concurrently.

Theorem (Vinyals, 2019)

*(Informal) CDCL SAT solvers with the **VSIDS** decision strategy **cannot** simulate general resolution.*

The Ordered Decision Strategy

The *ordered decision strategy*: choose the smallest unassigned variable according to a fixed order.

Unit propagations *do not* need to adhere to the fixed order. We want to understand when this can be leveraged.

This strategy has been studied in the context of DLL without clause learning [Beame et al., 2002].

Motivating Question

Is there a family of unsatisfiable CNFs $\{\phi_i\}_{i=1}^{\infty}$ that have polynomial sized resolution refutations but require superpolynomial time for CDCL with the ordered decision strategy, for any order?

Our Contributions

1. CDCL SAT Solvers with the ordered decision strategy and the DECISION learning scheme **cannot** simulate general resolution. They are no more powerful than ordered resolution.

2. CDCL SAT Solvers with the ordered decision strategy and a learning scheme we call FIRST-L can simulate general resolution, **given it can also ignore and unit propagations and conflicts**.

We also introduce a model and language for CDCL to aid the presentation of our results.

A Note on Interpretation

The first result applies to actual solver implementations, albeit with a rarely used combination of heuristics. It could, in principle, be demonstrated by experiment.

The second result does not apply to actual solvers. We prove that a reasonable proof system for simulating CDCL SAT solvers with the ordered decision strategy is surprisingly strong.

Other Refinements

- **Restarts.** These kinds of results often require very frequent restarts. It is unclear if this is necessary. This is an active area, with one paper on the subject in this conference.
  [Li et al., 2020]

- **Clause Deletion.** They also require that all learned clauses must stay in memory. Some solvers have rather aggressive clause deletion policies. Size-space trade-offs in can be extended to results about such policies.
  [Elffers et al., 2016].

Notation

We assume a finite set of variables $x_1, x_2, \ldots, x_n$.

We sometimes use x_i^0 in place of the literal $\neg x_i$ and x_i^1 in place of the literal x_i.

We use 0 for the empty clause.

Permutations $\pi \in S_n$ are extended to variables and literals as

$$\pi(x_i^a) \stackrel{\text{def}}{=} \pi(x_i) \stackrel{\text{def}}{=} \pi(i).$$

The Model

Motivations

Our model is not intended to be novel or controversial, it is influenced by previous models [Nieuwenhuis et al., 2006, Atserias et al., 2011, Elffers et al., 2016]. The underlying structure of the model is essentially a labeled transition system.

However, our motivations the opposite of existing models. Rather than define a model that is close to actual implementations, we define a basic model and then study its restrictions.

This makes it flexible enough to handle nonstandard sources of nondeterminism (ignoring unit propagations, ignoring conflicts) and make assumptions about nondeterminism more explicit.

Perhaps more useful is the associated language that makes statements about CDCL more precise and concise.

States and Transitions

A **trail** is an ordered partial assignment annotated with 'u's or 'd's. Λ is the empty trail and $t[\leq i]$ is the prefix of t of length i.

A **state** is a pair consisting of a CNF formula and a trail.

A state (τ, t) is **terminal** if $\tau|_t = 1$ or $0 \in \tau$.

$\mathbb{S}_n$ is the set of all states on n variables and $\mathbb{S}_n^{\circ}$ is the set of all non-terminal states.

For each state $S \in \mathbb{S}_n^{\circ}$, there is a set $\text{Actions}(S)$ and function $\text{Transition}_S : \text{Actions}(S) \to \mathbb{S}_n$. We write $S \xRightarrow{A} S'$ if $\text{Transition}_S(A) = S'$.

Actions

Actions$(S) \stackrel{\text{def}}{=} D(S) \dot{\cup} U(S) \dot{\cup} L(S)$ where

- $D(S)$ consists of $x_i \stackrel{d}{=} a$ such that x_i does not appear in t and $a \in \{0, 1\}$, with

$$(\tau, t) \stackrel{x_i \stackrel{d}{=} a}{\Longrightarrow} (\tau, [t, x_i \stackrel{d}{=} a]).$$

- $U(S)$ consists of $x_i \stackrel{u}{=} a$ for which $\tau|_t$ contains the unit clause x_i^a, with

$$(\tau, t) \stackrel{x_i \stackrel{u}{=} a}{\Longrightarrow} (\tau, [t, x_i \stackrel{u}{=} a]).$$

- $L(S)$ will consist of clause-trail pairs (C, t') defined in the next slide, with

$$(\tau, t) \stackrel{(C,t')}{\Longrightarrow} (\tau \cup \{C\}, t').$$

Clause Learning and $L(S)$

For state $S = (\tau, [x_{i_1} \overset{*_{i_1}}{=} a_{i_1}, \ldots, x_{i_r} \overset{*_{i_r}}{=} a_{i_r}])$, we define

- $\mathbb{C}_{r+1}(S) \overset{\text{def}}{=} \{D \in \tau : D|_t = 0\}$.
- If $*_{i_k} = u$, for $D \in \mathbb{C}_{k+1}(S)$,
  - If $x_{i_k}^{1-a_{i_k}} \in D$, then resolve D with all C such that $C|_{t[\leq k-1]} = x_{i_k}^{a_{i_k}}$ and add to $\mathbb{C}_k(S)$.
  - If $x_{i_k}^{1-a_{i_k}} \notin D$, then add D to $\mathbb{C}_k(S)$.
- If $*_{i_k} = d$, then $\mathbb{C}_k(S) = \mathbb{C}_{k+1}(S)$.

$$\mathbb{C}(S) \overset{\text{def}}{=} \bigcup_{i=1}^{r} \mathbb{C}_i(S)$$

$$L(S) \overset{\text{def}}{=} \begin{cases} \{(0, \Lambda)\} & 0 \in C \\ \{(C, t') : C \in \mathbb{C}(S)/\tau \text{ and } C|_{t'} \neq 0\} & \text{otherwise.} \end{cases}$$

Clause Learning and $L(S)$

For state $S = (\tau, [x_{i_1} \overset{*_{i_1}}{=} a_{i_1}, \ldots, x_{i_r} \overset{*_{i_r}}{=} a_{i_r}])$, we define

- $\mathbb{C}_{r+1}(S) \overset{\text{def}}{=} \{D \in \tau : D|_t = 0\}$.
- If $*_{i_k} = u$, for $D \in \mathbb{C}_{k+1}(S)$,
  - If $x_{i_k}^{1-a_{i_k}} \in D$, then resolve D with all C such that
- If $*_{i_k} = d$, then $\mathbb{C}_k(S) = \mathbb{C}_{k+1}(S)$.

$\mathbb{C}(S)$ is the set of all learnable clauses.

$$\mathbb{C}(S) \overset{\text{def}}{=} \bigcup_{i=1}^{r} \mathbb{C}_i(S)$$

$$L(S) \overset{\text{def}}{=} \begin{cases} \{(0, \Lambda)\} & 0 \in C \\ \{(C, t') : C \in \mathbb{C}(S)/\tau \text{ and } C|_{t'} \neq 0\} & \text{otherwise.} \end{cases}$$

Solvers

A **solver** is a partial function on $\mathbb{S}_n^{\mathrm{o}}$ such that $\mu(S) \in \mathrm{Actions}(S)$ when $\mu(S)$ is defined.

A **local class of solvers** is a collection of subsets $\mathrm{AllowedActions}(S) \subseteq \mathrm{Actions}(S)$ for $S \in \mathbb{S}_n^{\mathrm{o}}$ and consists of all solvers μ such that $\mu(S) \in \mathrm{AllowedActions}(S)$.

Amendments

Amendments *remove* actions from Actions(S) where $S = (\tau, t)$.

ALWAYS-C: $D(S)$ and $U(S)$ are removed from Actions(S) with $0 \in \mathbb{C}|_t$.

ALWAYS-U: $D(S)$ is removed from Actions(S) when there is a unit clause in $\mathbb{C}|_t$.

π-D: For permutation π, keep in $D(S)$ only $x_i \stackrel{d}{=} 0$ and $x_i \stackrel{d}{=} 1$ where x_i is the smallest variable not in t.

DECISION-L: In $L(S)$, keep only (C, t) where $C \in \mathbb{C}_1(S)$.

FIRST-L: In $L(S)$, keep only (C, t) where C is the result of resulting a conflict clause ($D \in \mathbb{C}_{r+1}(S)$) and some other clause.

CDCL Proof System

A **successful run** on τ is a sequence

$$S_0 \overset{A_0}{\Longrightarrow} S_1 \overset{A_1}{\Longrightarrow} \ldots S_{L-1} \overset{A_{L-1}}{\Longrightarrow} S_L$$

where $S_0 = (\tau, \Lambda)$ and $A_k \in \text{Actions}(S_k)$ and S_L is a terminal state.

For amendments $\mathcal{A}_1, \ldots, \mathcal{A}_r$, we let $\mathsf{CDCL}(\mathcal{A}_1, \ldots, \mathcal{A}_r)$ be the (possibly incomplete) proof system whose proofs are successful runs in which no action is affected by $\mathcal{A}_1, \ldots, \mathcal{A}_r$.

Example. $\mathsf{CDCL}(\mathsf{ALWAYS\text{-}C}, \mathsf{ALWAYS\text{-}U}, \mathsf{DECISION\text{-}L})$ polynomially simulates general resolution. This is a corollary of [Pipatsrisawat and Darwiche, 2011] which captures most of its content.

Contributions Rewritten

1. CDCL(DECISION-L, π-D) is polynomially equivalent to π-ordered resolution.

2. CDCL(FIRST-L, π-D) is polynomially equivalent to general resolution.

CDCL(DECISION-L, π-D) $=_p$ π-Ordered Resolution

π-Ordered Resolution

π-ordered resolution is the subsystem of resolution with the rule

$$\frac{C \vee x_i^a \qquad D \vee x_i^{1-a}}{C \vee D} \qquad \forall l \in (C \wedge D)(\pi(l) < \pi(x_i)).$$

π-Half-Ordered Resolution

π-half-ordered resolution is the subsystem of resolution with the rule

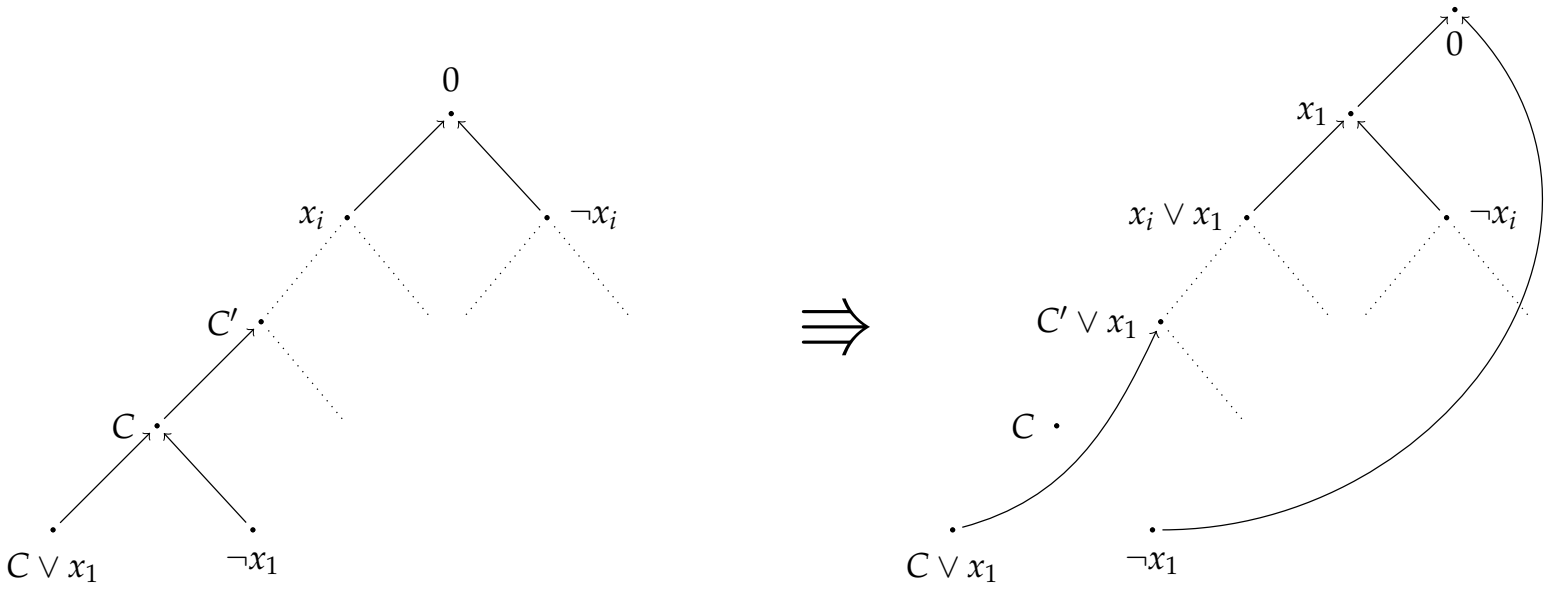
$$\frac{C \vee x_i^a \qquad D \vee x_i^{1-a}}{C \vee D} \qquad \forall l \in C(\pi(l) < \pi(x_i)).$$

We use π-half-ordered resolution as an intermediate system:

$$\begin{aligned} \mathsf{CDCL}(\mathsf{DECISION\text{-}L}, \pi\text{-}\mathsf{D}) &=_p \pi\text{-half-ordered resolution} \\ &=_p \pi\text{-ordered resolution} \end{aligned}$$

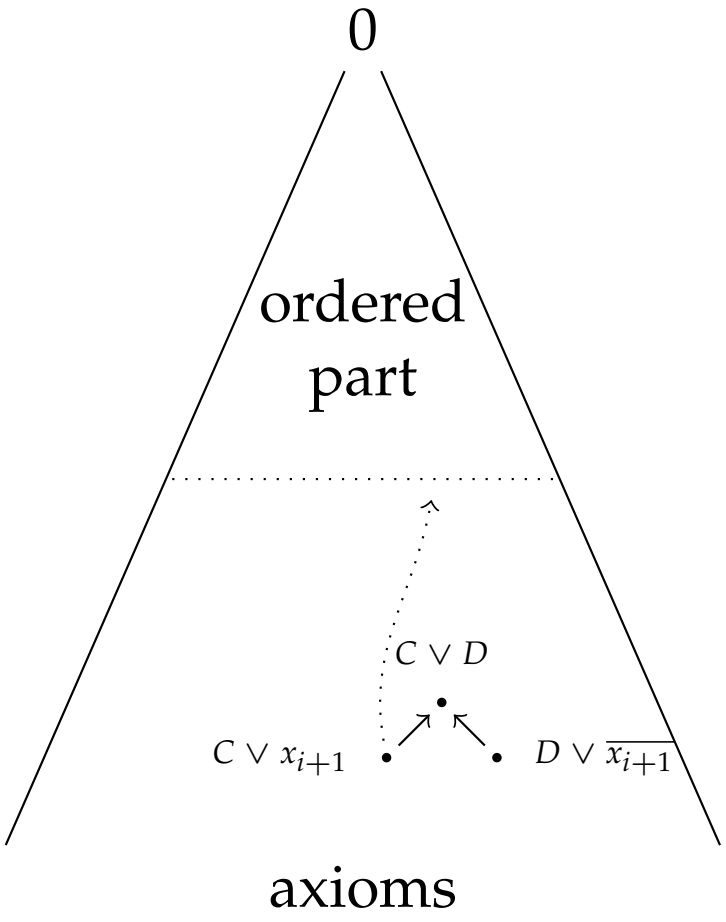
π-Half-Ord. Res. $=_p$ π-Ord. Res.

Idea. Fix unordered resolution steps from the top down by delaying them. For example:



Observation. If x_1 is the smallest variable appearing in a half-ordered proof, then it must appear as a unit clause. We can delay any resolutions with this unit clause until the end of the proof.

Ordered up to i



A proof is **ordered up to** i if for all clauses derived by resolving two clauses on some variable x s.t. $\pi(x) \leq i$, all following resolution steps are on variables x' where $\pi(x') < \pi(x) \leq i$.

π-ordered proofs are exactly those ordered up to $n - 1$.

A proof ordered up to i roughly splits into ordered part and remaining part containing only resolutions on variables greater than x_i.

Main Lemma

Lemma

If ϕ has a half-ordered refutation Π, then it has a refutation Π_i that is ordered up to i of size $\leq (i+1)\Pi$.

We can fixed the unordered steps in roughly the same way as in the first picture, except that they essentially must all be done simultaneously so that the size of the proof does not blow up.

CDCL(DECISION-L, π-D) $=_p$ π-Half Ord. Res.

The fact that CDCL(DECISION-L, π-D) polynomially simulates π-half-ordered resolution essentially follows from definitions.

For the other direction, it suffices to show that a learned clause $C \in \mathbb{C}_1((\tau, t))$ can be derived efficiently from clauses in τ.

Idea. The derivation of C can be viewed as a sequence of resolutions on the variables whose values are unit propagated in t. They can be reordered and duplicated to derive C while maintaining half-orderedness.

The New Derivation

For clause D learnable from τ, there are clauses $C_1, \ldots, C_{k+1}$ in τ with

$$C'_{k+1} \stackrel{\text{def}}{=} C_{k+1}$$
$$C'_j \stackrel{\text{def}}{=} \text{Res}(C'_{j+1}, C_j)$$

such that $D = C'_1$. We show that the derivation given by

$$C_{\gamma,\gamma} \stackrel{\text{def}}{=} C_\gamma$$
$$C_{\gamma,j} \stackrel{\text{def}}{=} \begin{cases} \text{Res}(C_{j,1}, C_{\gamma,j+1}) & \text{they are resolvable} \\ C_{\gamma,j+1} & \text{otherwise.} \end{cases}$$

has the property that $C'_1 = C_{k+1,1} = D$. Furthermore, the resolved variable is maximal in $C_{j,1}$ for all $j \in [k+1]$, so the derivation is π-half-ordered.

$\mathsf{CDCL}(\mathsf{FIRST\text{-}L}, \pi\text{-}\mathsf{D}) =_p$ General Resolution

π-Trail Resolution

We again use an intermediate proof system. It has the following rules:

$$\frac{t}{[t, x_i \overset{d}{=} a]}, \quad \text{(Decision rule)}$$

where x_i is the π-smallest index such that x_i does not appear in t and $a \in \{0, 1\}$ is arbitrary;

$$\frac{t \qquad C}{[t, x_i \overset{u}{=} a]}, \quad \text{(Unit propagation rule)}$$

where $C|_t = x_i^a$;

$$\frac{C \vee x_i^a \qquad D \vee x_i^{1-a} \qquad t}{C \vee D}, \quad \text{(Learning rule)}$$

where $(C \vee D)|_t = 0$, $(x_i \overset{*}{=} a) \in t$ and all other variables of C appear before x_i in t.

CDCL(FIRST-L, π-D) $=_p$ π-Trail Res. $=_p$ Gen. Res.

The first equivalence follows directly from definitions, and even holds for CDCL(π-D).

The second equivalent is by far our most technical result.

Idea. Because of the unit propagation rule, π-trail resolution is significantly more power in the presence of unit clauses. Our simulation algorithm generates a proof of *all literals* appearing in the simulated proof Π, and we apply it recursively to *pieces* of Π.

To build pieces of Π that are small enough to recurse on without blowing up the output size but large enough to make progress in the simulation, we need a new operator on proofs that we call *Variable Deletion*.

Variable Deletion

Variable deletion is analogous to restriction but for sets of variables as opposed to sets of variable assignments.

For a set of variables S,

$$\mathsf{Del}_S(\tau) \stackrel{\text{def}}{=} \{C \setminus \bigcup_{x \in S} \{x^0, x^1\} : C \in \tau\} \setminus \{0\}.$$

A resolution proof is **connected** if it has a unique sink.

Lemma

For a connected refutation Π with axioms τ and proper subset of variables S, $\mathsf{Del}_S(\Pi)$ is a refutation of $\mathsf{Del}_S(\tau)$.

This allows for a surgery-like process. We can simulate local parts of Π, and then stitch them back together.

Conclusions

We show that, modulo some nonstandard assumptions about the model, CDCL solvers with the ordered decision strategy vary in strength from ordered to general resolution depending on the learning scheme.

We have presented a flexible model of CDCL with more degrees of freedom and, hence, more possible systems to consider for study in the future.

Question. What is the exact strength of CDCL(π-D, ALWAYS-C, ALWAYS-U) or CDCL(π-D, 1UIP-L)?

Thank You

Bibliography I


Atserias, A., Fichte, J. K., and Thurley, M. (2011).
Clause-learning algorithms with many restarts and bounded-width resolution.
Journal of artificial intelligence research, 40:353–373.


Beame, P., Karp, R., Pitassi, T., and Saks, M. (2002).
The efficiency of resolution and Davis–Putnam procedures.
SIAM Journal on Computing, 31(4):1048–1075.


Beame, P., Kautz, H., and Sabharwal, A. (2004).
Towards understanding and harnessing the potential of clause learning.
Journal of Artificial Intelligence Research, 22:319–351.

Bibliography II

Elffers, J., Johannsen, J., Lauria, M., Magnard, T., Nordström, J., and Vinyals, M. (2016).
Trade-offs between time and memory in a tighter model of CDCL SAT solvers.
In *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 160–176. Springer.

Li, C., Fleming, N., Vinyals, M., Pitassi, T., and Ganesh, V. (2020).
Towards a Complexity-theoretic Understanding of Restarts in SAT solvers.
In *Theory and Applications of Satisfiability Testing – SAT 2020*. Springer.

Bibliography III


Nieuwenhuis, R., Oliveras, A., and Tinelli, C. (2006).
Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T).
Journal of the ACM (JACM), 53(6):937–977.


Pipatsrisawat, K. and Darwiche, A. (2011).
On the power of clause-learning SAT solvers as resolution engines.
Artificial Intelligence, 175(2):512–525.