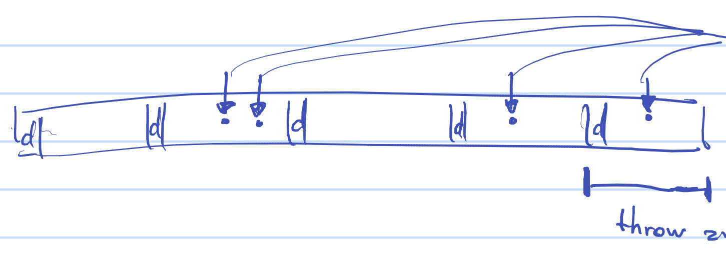
Learning, conflict graph, 1UIP:
↳ learn clause with exactly 1 literal at maximum decision level.
this makes it easy to backtrack.

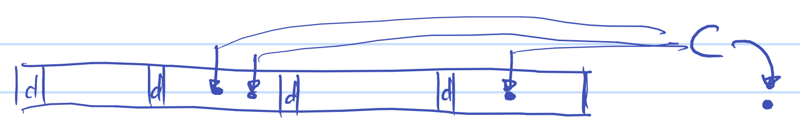


C 1UIP property.

trail α

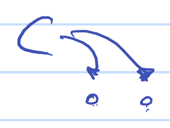
$\alpha' \subseteq \alpha$ with all vars since last decision removed.

α'

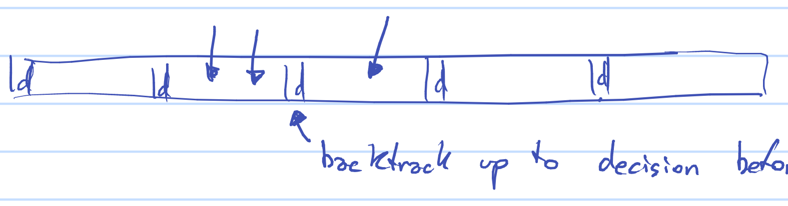


only one literal not set by α'!
$C|_{\alpha'}$ is a unit clause.

$C|_{\alpha'}$ is unit → next step after learning is unit propagation.



1UIP:
↳ learn clause with exactly 1 literal at maximum decision level. → wrt clause, not trail.



backtrack up to decision before max level in learned clause.

a learned clause st. after learning & backtracking causes unit propagation is called "asserting".

Clause database: data structure where keep clauses. initially $D = F$.
every learned clause we have $D = D \cup \{C_t\}$.

→ add clauses.
→ find which clauses unit / conflict.
↳ for each decision / unit prop.
for each clause $C \in D$.
if C contains variable d/u : restrict C.
→ unrestrict clauses when backtrack.

] can do better with lazy propagation "2WL"
↑
two watched literals.

while not solved:
if conflict: learn(). // $D \cup D \cup \{C_t\}$; backtrack().
if unit clause: propagate // $\alpha \cup \{\ell\}$; restrict().
o/w ~~decision~~ // $\alpha \cup \{\ell\}$ → forced. ↳ decision.
↳ decision
→ restart trail. ✓
→ forget clauses. ←

Restarts: $\alpha = \emptyset$.

makes no sense in DPLL!
but could be useful in randomized algorithm.
→ algorithm f, randomized, zero-error.
$\Pr[f \text{ runs in poly time.}] = 1/2$.
→ $\Pr[f \text{ runs in exp time.}] = 1/2$

original motivation: view CDCL as random (pseudo-random). $\mathbb{E}\, T(f) =$ exponential.

nowadays: 1. get out of bad part of search space (similar to optimization)
run f for a few steps. → 1, 2, 3, ...
if f terminates: answer.
o/w interrupt f; run again.
$\mathbb{E}\, T(f') =$ polynomial.

2. have a long trail, just optimizing running time.
↓
"quantitative" (poly speed-up)
→ "qualitative" (exp. speed-up).

open problem: is CDCL + restarts exp. better than CDCL \ restarts on some inputs?
general upper bounds use restarts: yes needed?
no particular examples need restarts: not needed?
discuss, make more precise later.

CDCL with restarts is complete: learning C → never reach same conflict again.

Forgetting. can $D \to D' \subset D$.

CDCL with forgetting not necessarily complete. (e.g. set $D' = \emptyset$ every time).

but maybe not so bad.

th: $\forall F$ $\forall t$ time step in running of CDCL(F). $\exists D_t$, $|D_t| = n + |F|$
↓
n variables
such that remembering D_t is enough for algorithm to complete.

no forgetting: $D_0 = F, D_1, \ldots D_T$
↳ $|D_T| = |F| + T$.

$D_0 = F, D_1', \ldots D_T'$ if state (α_t, D_t') instead of (α_t, D_t) run is also valid.
↑
$|D_t| \le n + |F|$

random decisions, remember subset of size $f(n)$, chosen at random (e.g. throw a cls at random after learning).

what is expected running time of CDCL?
not known? (open).
for other randomized algorithms it is known.

To specify CDCL need to specify
→ decisions → learning → restarts → forgetting
→ in addition: might not do greedy conflicts / UP.

while ( )
if conflict: —
~~o/w~~ if unit: —
~~o/w~~ decision or
restart or
forget.

How good is CDCL?
→ worst case: bad.

lemma: if CDCL runs in time T on a formula F then there is a resolution refutation of F of length T.]

th: $\exists$ family $\{F_n\}_{n \in \mathbb{N}}$, $|vars(F_n)| = n$, $|F_n| = O(n)$, st F_n requires resolution proofs of length $2^{\Omega(n)}$.

corollary: $\exists$ ~~~~~~ st CDCL(F_n) runs for $2^{\Omega(n)}$ steps.

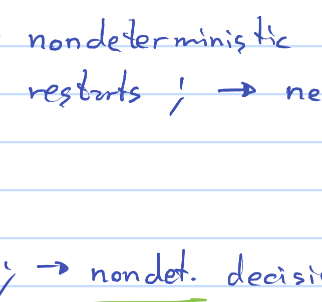
Maybe we can say sth in a fixed parameter tractability sense?
e.g. if treewidth $= k$ ⇒ run in time $O(n^k)$?
some other parameter?

yes!

converse of lemma: if there is resolution refutation of length L, can CDCL run quickly?

$\pi = (C_1, C_2 \ldots C_L)$ $C_i \in F$ or $C_i = Res(C_j, C_k)$ $j, k < i$

length of π is L.



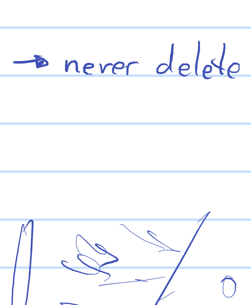
F is exp. large.
π is poly.

def: algorithm A poly-simulates resolution if
$\forall F, \pi_F$ $A(F)$ runs in time $poly(|F| + |\pi_F|)$.

th: [BKS'04] CDCL → non-greedy; → nondeterministic decisions; → adversarial / any learning; → always restarts; → never delete.
poly-simulates resolution.

th: [PD'11] CDCL → greedy (normal); → nondet. decisions; → asserting learning (1UIP, decision, ...); → frequent restarts; → never delete.
(any reasonable)
↳ first unique implication point.
poly-simulates resolution.

th: [AFT'11] ~~nondet decisions~~ random decisions.
poly-simulates bounded-width resolution
↳ $C_1 \ldots C_L$: $|C_i| < k$.



first cut such that exactly one variable at max. decision level.