Recall:

def D absorbs C if for every literal $a \in C$ it holds that setting $\alpha = \overline{C \setminus a}$ propagates a.

properties of absorption:

(i) → sound: D absorbs C then $D \models C$.

(ii) → closed: $C \in D$ then D absorbs C.

(iii) → clause-monotone: D absorbs C, $C \subseteq C'$ then D abs C'.

(iv) → DB-monotone: " $D \subseteq D'$ then $D' - C$.

th: CDCL (greedy) p-simulates resolution.

proof: assume $\pi = C_1 \ldots C_L$ proof of F. $n = |Vars(F)|$.

invariant: at time $i' = i \cdot poly(n)$ $D_{i'}$ absorbs all $C_j : j \leq i$.

obs enough to prove $D_{i'}$ absorbed. we'll do that.

if $C_i \in F$ then $C_i \in F \subseteq D_i$, hence already absorbed.

o/w $C_i = Res(A, B, x)$ and $A = C_j$, $B = C_{j'}$, j and $j' < i$, hence A and B absorbed.

obs if we set $\alpha = \overline{C_i} = \overline{A \vee B}$

we have $A|_\alpha = x$, $B|_\alpha = \bar{x}$.

since A and B absorbed, assignment α on DB D_i' propagates x and not x, which is a conflict.

this suggests a plan:

for each $a \in C_i$:
  while $D_{i'}|_{\overline{C_i \setminus a}}$ does not UP a:
    if conflict:
      learn E $(D_{i'+1} = D_{i'} \cup \{E\})$
      restart $(\alpha = \emptyset)$
    if unit: propagate
    o/w: decide $b = 0$ for some $b \in C_i$.

by construction after this procedure C_i is absorbed.
we have to show will finish in time $poly(n)$.

claim: we never set $b = 1$ for $b \in C_i$.

proof: this can only happen because of unit prop.
but then we would have for each $a \in C_i$
either $D_{i'}|_{\overline{C_i \setminus a}} \vdash_1 b \wedge \bar{b}$!! $(a \neq b)$
or $D_{i'}|_{\overline{C_i \setminus b}} \vdash_1 b$ $(a = b)$
hence C_i would already be absorbed.

obs if learning scheme was "decision", then procedure would finish after first conflict.
indeed, all decisions are $b = 0$ for some $b \in C_i$, and "decision" learning is $E = \overline{\text{decisions}} \subseteq C_i$, and learning $E \subseteq C_i$ enough to absorb C_i.

if learning scheme is only asserting, we need more work.
recall under asserting learning we only know that if $D|_\alpha \vdash_1$!!, then $E|_{\alpha'}$ is unit for some $\alpha' \subsetneq \alpha$. let $\ell = E|_{\alpha'}$.
at least one decision less.

claim: after at most n^2 steps we have $\ell = a$.

proof: let α be a conflict assignment and β its set of decisions. let ℓ be the literal becoming asserting.

If we ever assign β again then ℓ is propagated, hence we cannot learn ℓ again, and all literals that we learn after setting β are different.
hence after at most n learned clauses we have a as the asserting literal.
Note this is true if we assign β, but not if we only assign a prefix. Worst case we need to learn n clauses for each of the n prefixes, which makes n^2 steps.

Since we run this procedure $\leq |C|$ times for each clause and we have L clauses, the total number of learned clauses is $\leq L \cdot n^3 = L \cdot poly(n)$.

Also, learning a clause takes $poly(n)$ time, hence the runtime is $L \, poly(n)$ as we wanted to show. ⨳.

We proved CDCL can produce resolution proofs, but we needed to know the proof we were looking for to choose which variables to branch on.
If we do not insist on any resolution proof then random restrictions are enough.

def $\pi = \{C_1 \ldots C_L\}$ has width w if $|C_i| \leq w \; \forall i$.

prop: Exists algo st if F has proof of width w, will find it in time $n^{O(w)}$

proof: algo is as follows:

  $D = F$.
  while !! $\notin D$:
    pick $A, B \in D$ st $|Res(A,B)| \leq w$
    set $D = D \cup \{Res(A,B)\}$.

runtime is $n^{O(w)}$ because $|D| \leq$ # clauses with at most w literals $\leq \binom{n}{w} \cdot 3^w = n^{O(w)}$.

[AFT11]

th: If F has proof of width w, then CDCL with random decisions will find it in time $L \cdot n^{O(w)}$.

proof: Replace nondeterministic choices with random choices in the proof of the previous theorem.
with probability $\geq \frac{1}{(2n)^w}$ we will learn a clause as intended by the algorithm, and otherwise we will learn another clause. But learning other clauses does not hurt us.
Therefore expected runtime is

$n \cdot poly(n) \cdot (2n)^w$. ⨳

obs using CDCL seems overkill in view of the simple algorithm we just saw. However, we can run CDCL without knowing w!

obs we do not need to restart after every conflict. it is enough to restart after $poly(n)$ many conflicts, because learning extra clauses does not hurt.