CDCL vs Resolution

Marc Vinyals

DPLL

$y \vee z \quad y \vee \bar{z} \quad x \vee \bar{y} \vee z \quad x \vee \bar{y} \vee \bar{z} \quad \bar{x} \vee \bar{y}$

Algorithm 1: DPLL

while *not solved* **do**
  if *conflict* **then** backtrack()
  else if *unit* **then** propagate()
  else branch()

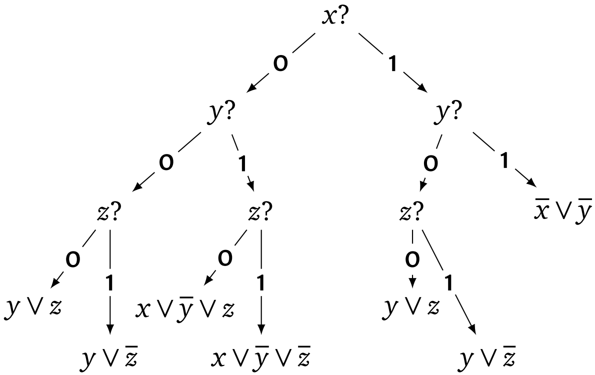
State: partial assignment

DPLL

$y \lor z \quad y \lor \overline{z} \quad x \lor \overline{y} \lor z \quad x \lor \overline{y} \lor \overline{z} \quad \overline{x} \lor \overline{y}$

Algorithm 1: DPLL

while *not solved* **do**
- **if** *conflict* **then** backtrack()
- **else if** *unit* **then** propagate()
- **else** branch()

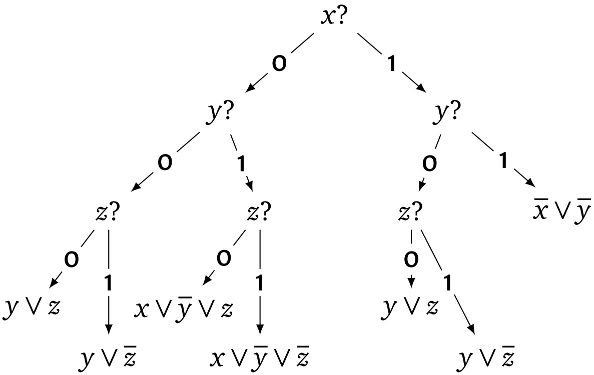


State: partial assignment

Resolution

- Interpret DPLL run as resolution proof

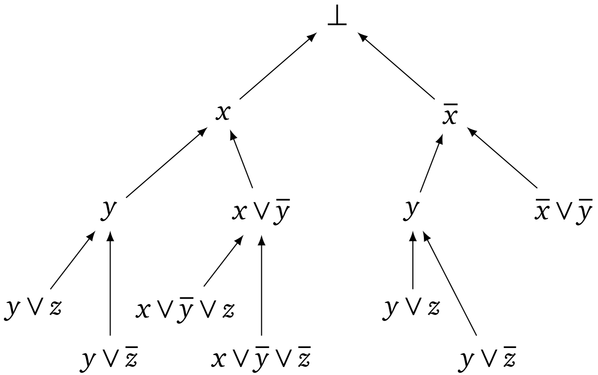
$$\frac{C \lor v \qquad D \lor \overline{v}}{C \lor D}$$



Resolution

- Interpret DPLL run as resolution proof

$$\frac{C \vee v \qquad D \vee \overline{v}}{C \vee D}$$

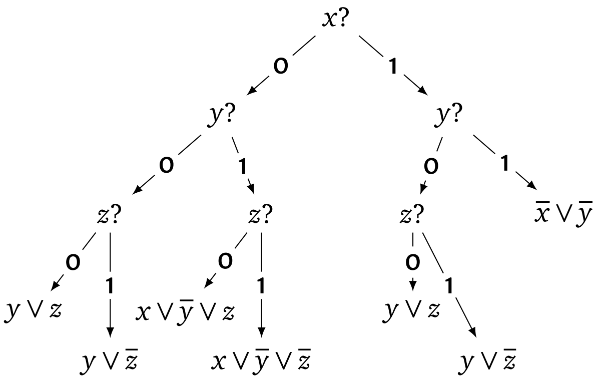


Resolution

- Interpret DPLL run as resolution proof

$$\frac{C \vee v \qquad D \vee \overline{v}}{C \vee D}$$

- And Resolution → DPLL?



Resolution to DPLL

Algorithm 1: DPLL

```
while not solved do
    if conflict then backtrack()
    else if unit then
        propagate()
    else branch on topmost
        available variable
```

DPLL can reproduce tree-like resolution proofs with at most $O(n)$ overhead

- # branches in search tree $\leq$ # branches in proof
- branch length $\leq n$
- $\Rightarrow$ proof size $L \rightarrow$ search tree size $\leq nL$

Resolution to DPLL

Sometimes $\Omega(n)$ overhead is needed.

- Take complete tautology over $x_1, \ldots, x_{\log n}$.
- Replace two variables in every clause with $y_{i,1}$.
- Add implications $y_{i,j} \to y_{i,j+1}$.
- Add another complete tautology over $x_1, \ldots, x_{\log n}$.

Resolution to DPLL

Sometimes $\Omega(n)$ overhead is needed.

- Take complete tautology over $x_1, \ldots, x_{\log n}$.
- Replace two variables in every clause with $y_{i,1}$.
- Add implications $y_{i,j} \to y_{i,j+1}$.
- Add another complete tautology over $x_1, \ldots, x_{\log n}$.

Notation

$\mathcal{C}(S) = \{\bigvee_{i \in S} x_i^{b_i} \mid b \in \{0,1\}^S\}$ = all $2^{|S|}$ full-width clauses over variables in $\{x_i \mid i \in S\}$

$\ell = \log n$

Formula

C	for $C \in \mathcal{C}([\ell])$
$C \vee y_{i,1}$	for $C \in \mathcal{C}(S)$, $S \in \binom{[\ell]}{\ell-2}$, $i \in [\ell]$
$y_{i,j} \to y_{i,j+1}$	for $i \in [\ell]$, $j \in [n]$

Resolution to DPLL

Sometimes $\Omega(n)$ overhead is needed.

- Take complete tautology over $x_1, \ldots, x_{\log n}$.
- Replace two variables in every clause with $y_{i,1}$.
- Add implications $y_{i,j} \to y_{i,j+1}$.
- Add another complete tautology over $x_1, \ldots, x_{\log n}$.

- Tree-like proof: branch on variables $x_1, \ldots, x_{\log n}$.
  Size $2^{\log n} = n$.

Notation

$\mathcal{C}(S) = \{\bigvee_{i \in S} x_i^{b_i} \mid b \in \{0,1\}^S\}$ = all $2^{|S|}$ full-width clauses over variables in $\{x_i \mid i \in S\}$

$\ell = \log n$

Formula

C	for $C \in \mathcal{C}([\ell])$
$C \vee y_{i,1}$	for $C \in \mathcal{C}(S)$, $S \in \binom{[\ell]}{\ell-2}$, $i \in [\ell]$
$y_{i,j} \to y_{i,j+1}$	for $i \in [\ell]$, $j \in [n]$

Resolution to DPLL

Sometimes $\Omega(n)$ overhead is needed.

- Take complete tautology over $x_1, \ldots, x_{\log n}$.
- Replace two variables in every clause with $y_{i,1}$.
- Add implications $y_{i,j} \to y_{i,j+1}$.
- Add another complete tautology over $x_1, \ldots, x_{\log n}$.

Notation

$\mathcal{C}(S) = \{\bigvee_{i \in S} x_i^{b_i} \mid b \in \{0,1\}^S\}$ = all $2^{|S|}$ full-width clauses over variables in $\{x_i \mid i \in S\}$

$\ell = \log n$

Formula

C	for $C \in \mathcal{C}([\ell])$
$C \vee y_{i,1}$	for $C \in \mathcal{C}(S)$, $S \in \binom{[\ell]}{\ell-2}$, $i \in [\ell]$
$y_{i,j} \to y_{i,j+1}$	for $i \in [\ell]$, $j \in [n]$

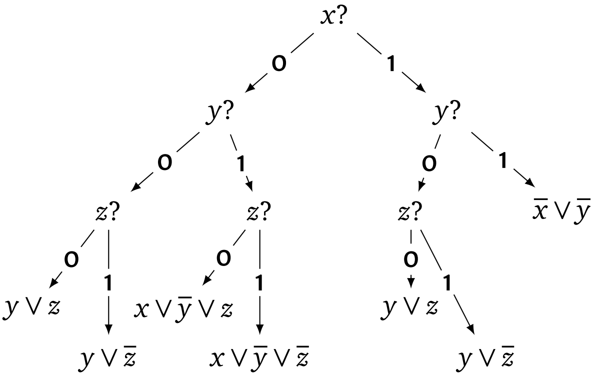
- Tree-like proof: branch on variables $x_1, \ldots, x_{\log n}$.
  Size $2^{\log n} = n$.
- DPLL run: branch on variables $x_1, \ldots, x_{\log n-2}$, propagate all $y_{i,j}$, branch on $x_{\log n-1}, x_{\log n}$.
  Size $2^{\log n} \cdot n \log n \simeq n^2$.

DPLL

$y \lor z \quad y \lor \overline{z} \quad x \lor \overline{y} \lor z \quad x \lor \overline{y} \lor \overline{z} \quad \overline{x} \lor \overline{y}$

Algorithm 1: DPLL

while *not solved* **do**
- **if** *conflict* **then** backtrack()
- **else if** *unit* **then** propagate()
- **else** branch()



State: partial assignment

CDCL

$y \vee z \quad y \vee \bar{z} \quad x \vee \bar{y} \vee z \quad x \vee \bar{y} \vee \bar{z} \quad \bar{x} \vee \bar{y}$

Algorithm 2: CDCL

while *not solved* **do**

  if *conflict* **then** learn()

  else if *unit* **then** propagate()

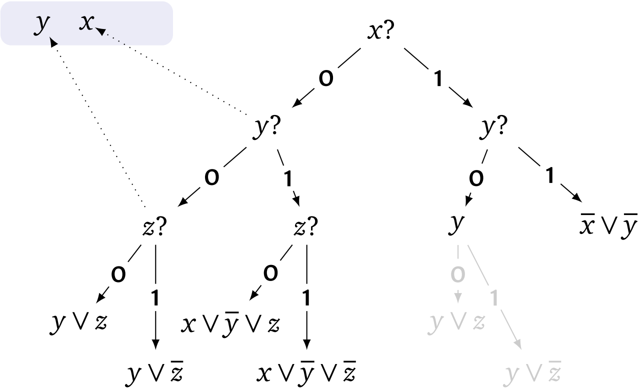
  else

    maybe forget()

    maybe restart()

    branch()

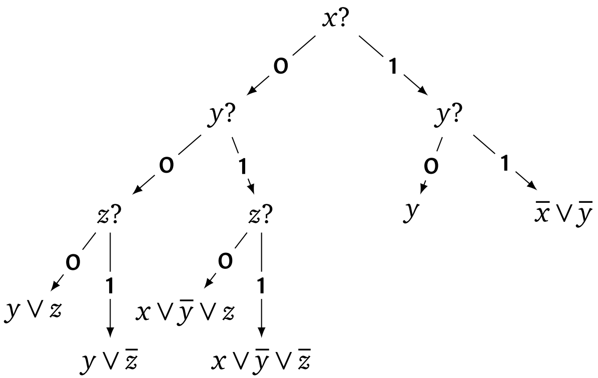
State: partial assignment
& learned clauses



Resolution

- Interpret CDCL run as resolution proof

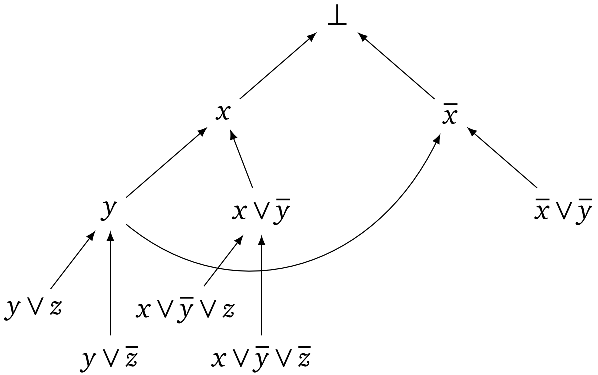
$$\frac{C \vee v \qquad D \vee \overline{v}}{C \vee D}$$



Resolution

- Interpret CDCL run as resolution proof

$$\frac{C \vee v \qquad D \vee \overline{v}}{C \vee D}$$



CDCL vs Resolution

- CDCL implicit proofs are in resolution form
- DPLL proofs only in weaker “tree-like” resolution form
  - There are formulas with polynomial resolution proofs but all tree-like proofs are exponential
- Is CDCL as powerful as general resolution?

CDCL vs Resolution

- CDCL implicit proofs are in resolution form
- DPLL proofs only in weaker “tree-like” resolution form
  - There are formulas with polynomial resolution proofs but all tree-like proofs are exponential
- Is CDCL as powerful as general resolution?

- Partial results in 2000s

[Beame, Kautz, Sabharwal ’04]
[Van Gelder ’05]
[Hertel, Bacchus, Pitassi, Van Gelder ’08]
[Buss, Hoffmann, Johannsen ’08]

CDCL vs Resolution

- CDCL implicit proofs are in resolution form
- DPLL proofs only in weaker "tree-like" resolution form
  - There are formulas with polynomial resolution proofs but all tree-like proofs are exponential
- Is CDCL as powerful as general resolution?

- Partial results in 2000s

[Beame, Kautz, Sabharwal '04]
[Van Gelder '05]
[Hertel, Bacchus, Pitassi, Van Gelder '08]
[Buss, Hoffmann, Johannsen '08]

- Yes (under natural model)

[Pipatsrisawat, Darwiche '09]
[Atserias, Fichte, Thurley '09]
[Beyersdorff, Böhm '21]

CDCL equivalent to Resolution: Results

Theorem [Pipatsrisawat, Darwiche '09]

With **non-deterministic** variable decisions,
CDCL can efficiently find resolution proofs

Theorem [Atserias, Fichte, Thurley '09]

With **random** variable decisions,
CDCL can efficiently find **bounded-width** resolution proofs

CDCL equivalent to Resolution: Results

Theorem [Pipatsrisawat, Darwiche '09]

With **non-deterministic** variable decisions,
CDCL can efficiently ~~find~~ reproduce resolution proofs

Theorem [Atserias, Fichte, Thurley '09]

With **random** variable decisions,
CDCL can efficiently find **bounded-width** resolution proofs

CDCL equivalent to Resolution: Simulation

- Derivation $\pi = C_1, \ldots, C_t$.
- Goal: learn every clause $C_i \in \pi$.

CDCL equivalent to Resolution: Simulation

- Derivation $\pi = C_1, \ldots, C_t$.
- Goal: ~~learn~~ absorb every clause $C_i \in \pi$.
- C **absorbed** if learning C does not enable more unit propagations.

CDCL equivalent to Resolution: Simulation

- Derivation $\pi = C_1, \ldots, C_t$.
- Goal: ~~learn~~ absorb every clause $C_i \in \pi$.
- C **absorbed** if learning C does not enable more unit propagations.

Example

$x \vee y \vee z \quad x \vee y \vee \overline{z}$

$x \vee y$ not absorbed:

- if $x = 0$ then would propagate y, but DB does not.

CDCL equivalent to Resolution: Simulation

- Derivation $\pi = C_1, \ldots, C_t$.
- Goal: ~~learn~~ absorb every clause $C_i \in \pi$.
- C **absorbed** if learning C does not enable more unit propagations.

Example

$x \vee y \vee z \quad x \vee y \vee \overline{z}$

$x \vee y$ not absorbed:

- if $x = 0$ then would propagate y, but DB does not.

$x \vee z \quad y \vee z \quad x \vee y \vee \overline{z}$

$x \vee y$ is absorbed:

- if $x = 0$ then propagate $z = 1$ and $y = 1$;
- if $y = 0$ then propagate $z = 1$ and $x = 1$.

CDCL equivalent to Resolution: Simulation

- Derivation $\pi = C_1, \ldots, C_t$.
- Goal: ~~learn~~ absorb every clause $C_i \in \pi$.
- C **absorbed** if learning C does not enable more unit propagations.

Algorithm 3: Simulation

for $C_i \in \pi$ **do**
  while C_i *not absorbed* **do**
    if *conflict* **then**
      learn()
      restart()
    else if *unit* **then** propagate()
    else assign a literal in C_i to false

CDCL equivalent to Resolution: Assumptions

```
for C_i ∈ π do
    while C_i not absorbed do
        if conflict then
            learn()
            restart()
        else if unit then propagate()
        else assign a literal in C_i to false
    restart()
```

- Optimal variable choices
- Clauses not thrown away
- Frequent restarts
- Standard learning

CDCL equivalent to Resolution: Assumptions

```
for C_i ∈ π do
    while C_i not absorbed do
        if conflict then
            learn()
            restart()
        else if unit then propagate()
        else assign a literal in C_i to false
    restart()
```

- Optimal variable choices
- Clauses not thrown away
- Frequent restarts
- Standard learning

CDCL equivalent to Resolution: Assumptions

```
for C_i ∈ π do
    while C_i not absorbed do
        if conflict then
            learn()
            restart()
        else if unit then propagate()
        else assign a literal in C_i to false
    restart()
```

- Optimal variable choices
- Clauses not thrown away
- Frequent restarts
- Standard learning

CDCL equivalent to Resolution: Assumptions

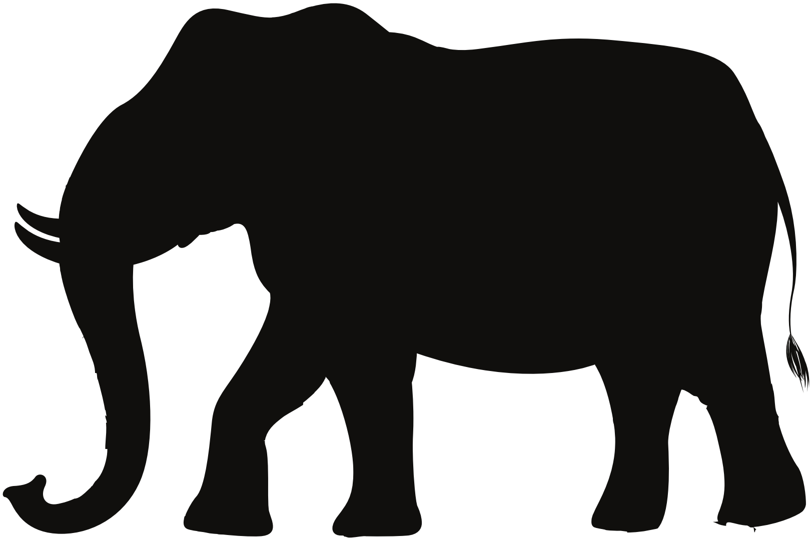
for $C_i \in \pi$ **do**
  while C_i *not absorbed* **do**
    if *conflict* **then**
      learn()
      restart()
    else if *unit* **then** propagate()
    else assign a literal in C_i to false
  restart()

- Optimal variable choices
- Clauses not thrown away
- Frequent restarts
- Standard learning

CDCL equivalent to Resolution: Assumptions

```
for C_i ∈ π do
    while C_i not absorbed do
        if conflict then
            learn()
            restart()
        else if unit then propagate()
        else assign a literal in C_i to false
    restart()
```

- Optimal variable choices
- Clauses not thrown away
- Frequent restarts
- Standard learning



CDCL equivalent to Resolution: Assumptions

```
for C_i ∈ π do
    while C_i not absorbed do
        if conflict then
            learn()
            restart()
        else if unit then propagate()
        else assign a literal in C_i to false
    restart()
```

- Optimal variable choices
- Clauses not thrown away
- Frequent restarts
- Standard learning

Branching

Optimal variable choices are needed

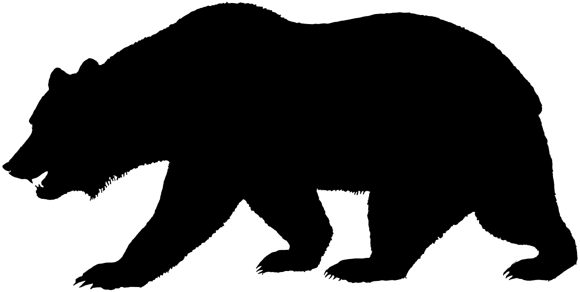
- No deterministic algorithm simulates resolution unless FPT hierarchy collapses.
  [Alekhnovich, Razborov '01]
- No deterministic algorithm simulates resolution unless $\mathrm{P} = \mathrm{NP}$.
  [Atserias, Müller '19]

Branching

Optimal variable choices are needed

- No deterministic algorithm simulates resolution unless FPT hierarchy collapses.
  [Alekhnovich, Razborov '01]
- No deterministic algorithm simulates resolution unless $P = NP$.
  [Atserias, Müller '19]

- CDCL with any static order exponentially worse than resolution.
  [Mull, Pang, Razborov '19]
- CDCL with VSIDS and similar heuristics exponentially worse than resolution.
  [V '20]



CDCL equivalent to Resolution: Assumptions

---

for $C_i \in \pi$ **do**
    while C_i *not absorbed* **do**
        if *conflict* **then**
            learn()
            restart()
        else if *unit* **then** propagate()
        else assign a literal in C_i to false
    restart()

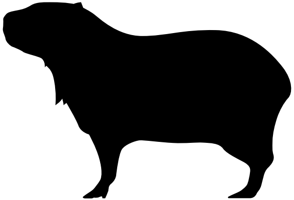
- Optimal variable choices
- Clauses not thrown away
- Frequent restarts
- Standard learning

Throwing Clauses Away

- With nondeterministic erasures enough to keep only $n \ll L$ clauses in memory.
  [Esteban, Torán '01]
- But more are needed to simulate resolution:
- Keeping $\ll n$ clauses can exponentially blow-up runtime.
  [Ben Sasson, Nordström '11]
- Keeping $\ll n^k$ clauses can superpolynomially blow-up runtime.
  [Beame, Beck, Impagliazzo '12; Beck, Nordström, Tang '13]

Throwing Clauses Away

- With nondeterministic erasures enough to keep only $n \ll L$ clauses in memory.
  [Esteban, Torán '01]
- But more are needed to simulate resolution:
- Keeping $\ll n$ clauses can exponentially blow-up runtime.
  [Ben Sasson, Nordström '11]
- Keeping $\ll n^k$ clauses can superpolynomially blow-up runtime.
  [Beame, Beck, Impagliazzo '12; Beck, Nordström, Tang '13]
- Keeping only narrow clauses can exponentially blow-up runtime.
  [Thapen '16]
- What about clauses with low LBD?



CDCL equivalent to Resolution: Assumptions

```
for C_i ∈ π do
    while C_i not absorbed do
        if conflict then
            learn()
            restart()
        else if unit then propagate()
        else assign a literal in C_i to false
    restart()
```

- Optimal variable choices
- Clauses not thrown away
- Frequent restarts
- Standard learning

Frequent Restarts

- Does useful work happen between restarts?

Frequent Restarts

- Does useful work happen between restarts?

- CDCL without restarts and non-greedy UP/conflicts simulates resolution.
  [Beame, Kautz, Sabharwal ’04]
- CDCL without restarts and preprocessing simulates resolution.
  [Hertel, Bacchus, Pitassi, Van Gelder ’08]

Frequent Restarts

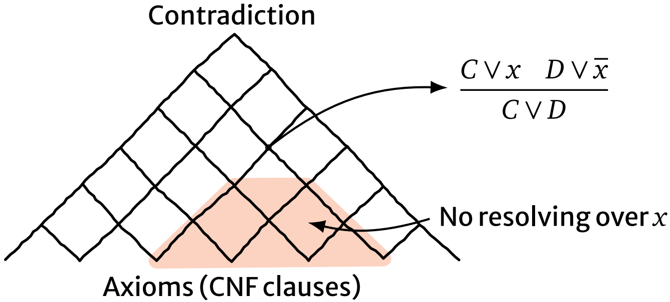
- Does useful work happen between restarts?

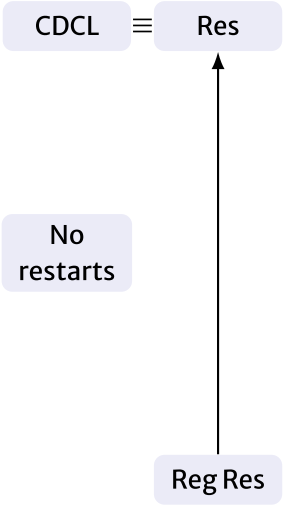
- CDCL without restarts and non-greedy UP/conflicts simulates resolution.
  [Beame, Kautz, Sabharwal '04]
- CDCL without restarts and preprocessing simulates resolution.
  [Hertel, Bacchus, Pitassi, Van Gelder '08]

- CDCL without restarts between *regular* and *standard* resolution.

CDCL and Regular Resolution

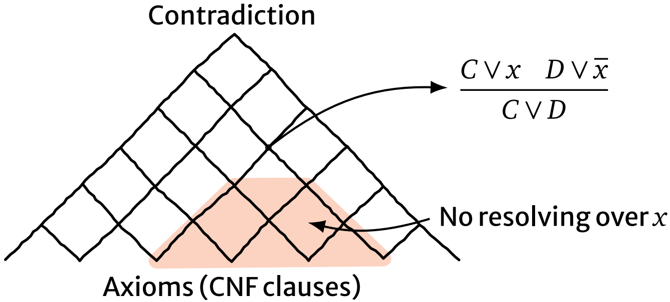
- Regular resolution: do not resolve a variable twice on same path.

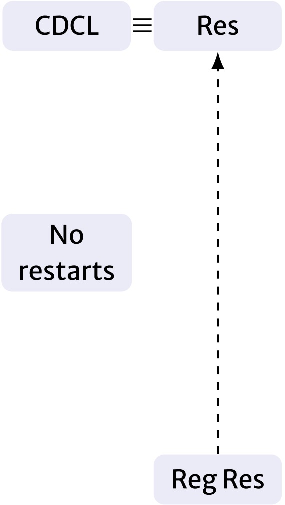




CDCL and Regular Resolution

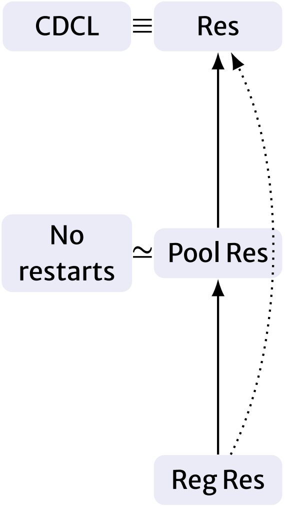
- Regular resolution: do not resolve a variable twice on same path.
- Regular resolution exponentially weaker than general.
  (Exist formulas with short proofs but exponentially long regular proofs)





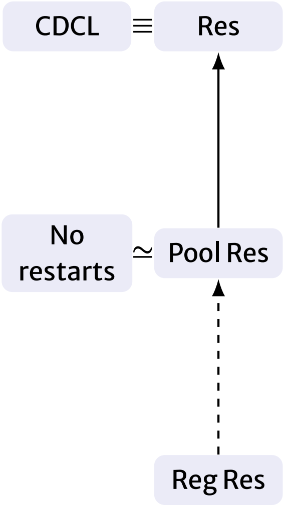
CDCL and Regular Resolution

- Regular resolution: do not resolve a variable twice on same path.
- Regular resolution exponentially weaker than general.
  (Exist formulas with short proofs but exponentially long regular proofs)
- Pool resolution $\simeq$ CDCL w/o restarts. [Van Gelder '05]
  - Pool res $\geq$ Regular res $\Rightarrow$ Formulas that separate general and regular are good candidates to separate general and pool.



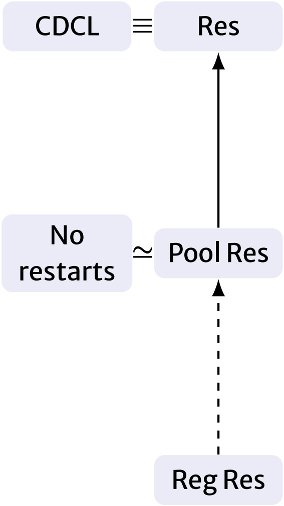
CDCL and Regular Resolution

- Regular resolution: do not resolve a variable twice on same path.
- Regular resolution exponentially weaker than general.
  (Exist formulas with short proofs but exponentially long regular proofs)
- Pool resolution $\simeq$ CDCL w/o restarts. [Van Gelder '05]
  - Pool res $\geq$ Regular res $\Rightarrow$ Formulas that separate general and regular are good candidates to separate general and pool.
- All such formulas easy for pool resolution.
  [Bonet, Buss, Johannsen '12]
  [Buss, Kołodziejczyk '14]



CDCL and Regular Resolution

- Regular resolution: do not resolve a variable twice on same path.
- Regular resolution exponentially weaker than general.
  (Exist formulas with short proofs but exponentially long regular proofs)
- Pool resolution $\simeq$ CDCL w/o restarts. [Van Gelder '05]
  - Pool res $\geq$ Regular res $\Rightarrow$ Formulas that separate general and regular are good candidates to separate general and pool.
- All such formulas easy for pool resolution.
  [Bonet, Buss, Johannsen '12]
  [Buss, Kołodziejczyk '14]
- Formula with CDCL proof of length L but requires $L+1$ w/o restarts?



CDCL equivalent to Resolution: Assumptions

```
for Ci ∈ π do
    while Ci not absorbed do
        if conflict then
            learn()
            restart()
        else if unit then propagate()
        else assign a literal in Ci to false
    restart()
```

- Optimal variable choices
- Clauses not thrown away
- Frequent restarts
- Standard learning

Learning

- Any **asserting** learning scheme works.
- C asserting if unit after backtracking.
- 1UIP is asserting.

Learning

- Any **asserting** learning scheme works.
- C asserting if unit after backtracking.
- 1UIP is asserting.

- Less overhead with decision learning scheme.
- Is decision faster than 1UIP?
- How much overhead is needed?

Merge Resolution

- A resolution step is a merge if C and D share a literal.

Merge

$$\frac{x \lor y \lor z \qquad x \lor y \lor \overline{z}}{x \lor y}$$

Not a merge

$$\frac{x \lor z \qquad y \lor \overline{z}}{x \lor y}$$

- Merge resolution: at least one premise either axiom or merge. [Andrews '68]

Merge Resolution

- A resolution step is a merge if C and D share a literal.

Merge

$$\frac{x \vee y \vee z \qquad x \vee y \vee \overline{z}}{x \vee y}$$

Not a merge

$$\frac{x \vee z \qquad y \vee \overline{z}}{x \vee y}$$

- Merge resolution: at least one premise either axiom or merge. [Andrews '68]
- Merge resolution 2.0: only reuse merges.
- 1UIP produces merge resolution proofs.
- Merge resolution can simulate standard resolution with $O(n)$ overhead.
- And $\Omega(n)$ overhead sometimes needed. [Fleming, Ganesh, Kolokolova, Li, V]

Take Home

- CDCL equivalent to Resolution
- But only under assumptions, not all reasonable

Take Home

- CDCL equivalent to Resolution
- But only under assumptions, not all reasonable

Open Problems

- CDCL-specific results about space?
- Are restarts important?
- How much overhead do we need?

Take Home

Images: Vecteezy.com

- CDCL equivalent to Resolution
- But only under assumptions, not all reasonable

Open Problems

- CDCL-specific results about space?
- Are restarts important?
- How much overhead do we need?

Thanks!