Size-Space Bounds and Trade-offs for CDCL Proofs

Marc Vinyals

KTH Royal Institute of Technology
Stockholm, Sweden

Joint work with Jan Johannsen,
Massimo Lauria and Jakob Nordström

FLoC Workshop on Proof Complexity

Algorithms for SAT [1960s]

- State-of-the-art 1960s: DPLL

```
while not solved :
  unit_propagate()
  if conflict :


    backtrack()
  else :
    decide_variable_assignment()
```

Algorithms for SAT

- State-of-the-art 1960s: DPLL
- State-of-the-art now: Conflict Driven Clause Learning

```
while not solved :
  unit_propagate()
  if conflict :
    learn()

    backjump()
  else :
    decide_variable_assignment()
```

Algorithms for SAT

- State-of-the-art 1960s: DPLL
- State-of-the-art now: Conflict Driven Clause Learning

```
while not solved :
  unit_propagate()
  if conflict :
    learn()
    maybe_restart()
    backjump()
  else :
    decide_variable_assignment()
```

What is the power of CDCL?

- Execution trace of CDCL $\rightarrow$ proof in resolution

What is the power of CDCL?

- Execution trace of CDCL $\rightarrow$ proof in resolution
- Even true for most forms of preprocessing

What is the power of CDCL?

- Execution trace of CDCL $\rightarrow$ proof in subsystem of resolution
- Even true for most forms of preprocessing
- How strong a subsystem?
  - DPLL only as strong as tree-like
  - CDCL DAG-like, how much?

Known results

Line of research:

CDCL polynomially simulates resolution, but artificial models

[Beame, Kautz, Sabharwal '04], [Van Gelder '05],

[Hertel, Bacchus, Pitassi, Van Gelder '08], [Buss, Hoffmann, Johannsen '08]

Known results

Line of research:
CDCL polynomially simulates resolution, but artificial models
[Beame, Kautz, Sabharwal '04], [Van Gelder '05],
[Hertel, Bacchus, Pitassi, Van Gelder '08], [Buss, Hoffmann, Johannsen '08]

Under natural model (still some technical assumptions)

- CDCL p-simulates resolution [Pipatsrisawat, Darwiche '09]
- Randomized CDCL efficiently finds narrow proofs [Atserias, Fichte, Thurley '09]

Both results can be obtained from both papers

An even more faithful model?

Technical assumptions

- Unlimited memory
- Very frequent restarts
- Random decisions

An even more faithful model?

Technical assumptions

- Unlimited memory
- Very frequent restarts
- Random decisions

Relevant questions

- Can we model memory/space?
- How important are restarts?
- Performance of actual heuristic vs random decisions?

An even more refined model

- Based on [Pipatsrisawat, Darwiche '09], [Atserias, Fichte, Thurley '09]; also ideas from [Buss, Hoffmann, Johannsen '08]

An even more refined model

- Based on [Pipatsrisawat, Darwiche '09], [Atserias, Fichte, Thurley '09]; also ideas from [Buss, Hoffmann, Johannsen '08]

- Proof as explicit resolution DAG + verification of CDCL-like

An even more refined model

- Based on [Pipatsrisawat, Darwiche '09], [Atserias, Fichte, Thurley '09]; also ideas from [Buss, Hoffmann, Johannsen '08]
- Proof as explicit resolution DAG + verification of CDCL-like
- Natural measure of space captures erasure of learnt clauses

A closer look at CDCL

```
while not solved :
  unit_propagate()     // Unit resolution
  if conflict :
    learn()
    maybe_restart()
    backjump()
  else :
    assign_variable() // New decision
```

Data structures

- Branching sequence (assignments by decision or propagation)
- Clause database (list of learned clauses)

A closer look at CDCL

(Partial) input: $z \wedge (x \vee y) \wedge (\overline{u} \vee \overline{w}) \wedge (u \vee \overline{w} \vee \overline{x} \vee y \vee \overline{z})$

```
while not solved :
  unit_propagate()
  if conflict :
    learn()
    maybe_restart()
    backjump()
  else :
    assign_variable()
```

Data structures

- Branching sequence
- Clause database: $w \vee \overline{x} \vee y \vee \overline{z}$

A closer look at CDCL

(Partial) input: $z \wedge (x \vee y) \wedge (\overline{u} \vee \overline{w}) \wedge (u \vee \overline{w} \vee \overline{x} \vee y \vee \overline{z})$

```
while not solved :
  unit_propagate()
  if conflict :
    learn()
    maybe_restart()
    backjump()
  else :
    assign_variable()
```

$z = 1 \quad z$

Data structures

- Branching sequence
- Clause database: $w \vee \overline{x} \vee y \vee \overline{z}$

A closer look at CDCL

(Partial) input: $z \wedge (x \vee y) \wedge (\overline{u} \vee \overline{w}) \wedge (u \vee \overline{w} \vee \overline{x} \vee y \vee \overline{z})$

```
while not solved :
  unit_propagate()
  if conflict :
    learn()
    maybe_restart()
    backjump()
  else :
    assign_variable()
```

$z = 1$ z

$y \stackrel{d}{=} 0$ (Decision)

Data structures

- Branching sequence
- Clause database: $w \vee \overline{x} \vee y \vee \overline{z}$

A closer look at CDCL

(Partial) input: $z \wedge (x \vee y) \wedge (\overline{u} \vee \overline{w}) \wedge (u \vee \overline{w} \vee \overline{x} \vee y \vee \overline{z})$

```
while not solved :
  unit_propagate()
  if conflict :
    learn()
    maybe_restart()
    backjump()
  else :
    assign_variable()
```

$z = 1$	z
$y \stackrel{d}{=} 0$	(Decision)
$x = 1$	$x \vee y$

Data structures

- Branching sequence
- Clause database: $w \vee \overline{x} \vee y \vee \overline{z}$

A closer look at CDCL

(Partial) input: $z \wedge (x \vee y) \wedge (\overline{u} \vee \overline{w}) \wedge (u \vee \overline{w} \vee \overline{x} \vee y \vee \overline{z})$

```
while not solved :
  unit_propagate()
  if conflict :
    learn()
    maybe_restart()
    backjump()
  else :
    assign_variable()
```

$z = 1$	z
$y \stackrel{d}{=} 0$	(Decision)
$x = 1$	$x \vee y$
$w = 1$	$w \vee \overline{x} \vee y \vee \overline{z}$

Data structures

- Branching sequence
- Clause database: $w \vee \overline{x} \vee y \vee \overline{z}$

A closer look at CDCL

(Partial) input: $z \wedge (x \vee y) \wedge (\overline{u} \vee \overline{w}) \wedge (u \vee \overline{w} \vee \overline{x} \vee y \vee \overline{z})$

```
while not solved :
  unit_propagate()
  if conflict :
    learn()
    maybe_restart()
    backjump()
  else :
    assign_variable()
```

$z = 1$	z
$y \stackrel{d}{=} 0$	(Decision)
$x = 1$	$x \vee y$
$w = 1$	$w \vee \overline{x} \vee y \vee \overline{z}$
$u = 0$	$\overline{u} \vee \overline{w}$

Data structures

- Branching sequence
- Clause database: $w \vee \overline{x} \vee y \vee \overline{z}$

A closer look at CDCL

(Partial) input: $z \wedge (x \vee y) \wedge (\overline{u} \vee \overline{w}) \wedge (u \vee \overline{w} \vee \overline{x} \vee y \vee \overline{z})$

```
while not solved :
  unit_propagate()
  if conflict :
    learn()
    maybe_restart()
    backjump()
  else :
    assign_variable()
```

$z = 1$	z
$y \stackrel{d}{=} 0$	(Decision)
$x = 1$	$x \vee y$
$w = 1$	$w \vee \overline{x} \vee y \vee \overline{z}$
$u = 0$	$\overline{u} \vee \overline{w}$

$u \vee \overline{w} \vee \overline{x} \vee y \vee \overline{z}$

Data structures

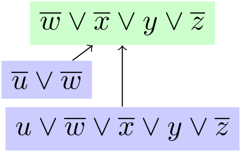
- Branching sequence
- Clause database: $w \vee \overline{x} \vee y \vee \overline{z}$

A closer look at CDCL

(Partial) input: $z \wedge (x \vee y) \wedge (\overline{u} \vee \overline{w}) \wedge (u \vee \overline{w} \vee \overline{x} \vee y \vee \overline{z})$

```
while not solved :
  unit_propagate()
  if conflict :
    learn()
    maybe_restart()
    backjump()
  else :
    assign_variable()
```

$z = 1$	z
$y \overset{d}{=} 0$	(Decision)
$x = 1$	$x \vee y$
$w = 1$	$w \vee \overline{x} \vee y \vee \overline{z}$
$u = 0$	$\overline{u} \vee \overline{w}$



Data structures

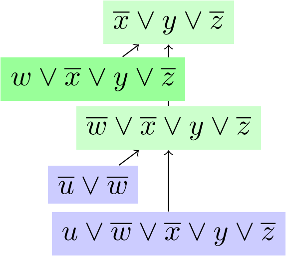
- Branching sequence
- Clause database: $w \vee \overline{x} \vee y \vee \overline{z}$

A closer look at CDCL

(Partial) input: $z \wedge (x \vee y) \wedge (\overline{u} \vee \overline{w}) \wedge (u \vee \overline{w} \vee \overline{x} \vee y \vee \overline{z})$

```
while not solved :
  unit_propagate()
  if conflict :
    learn()
    maybe_restart()
    backjump()
  else :
    assign_variable()
```

$z = 1$	z
$y \overset{d}{=} 0$	(Decision)
$x = 1$	$x \vee y$
$w = 1$	$w \vee \overline{x} \vee y \vee \overline{z}$



Data structures

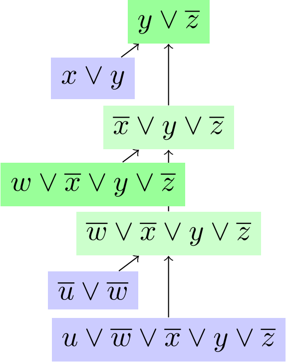
- Branching sequence
- Clause database: $w \vee \overline{x} \vee y \vee \overline{z}$

A closer look at CDCL

(Partial) input: $z \wedge (x \vee y) \wedge (\overline{u} \vee \overline{w}) \wedge (u \vee \overline{w} \vee \overline{x} \vee y \vee \overline{z})$

```
while not solved :
  unit_propagate()
  if conflict :
    learn()
    maybe_restart()
    backjump()
  else :
    assign_variable()
```

$z = 1$	z
$y \overset{d}{=} 0$	(Decision)
$x = 1$	$x \vee y$



Data structures

- Branching sequence
- Clause database: $w \vee \overline{x} \vee y \vee \overline{z}$

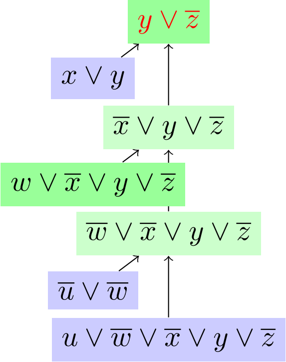
A closer look at CDCL

(Partial) input: $z \wedge (x \vee y) \wedge (\overline{u} \vee \overline{w}) \wedge (u \vee \overline{w} \vee \overline{x} \vee y \vee \overline{z})$

```
while not solved :
  unit_propagate()
  if conflict :
    learn()
    maybe_restart()
    backjump()
  else :
    assign_variable()
```

$z = 1$ z

$y \stackrel{d}{=} 0$ (Decision)



Data structures

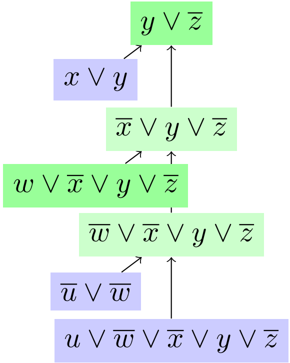
- Branching sequence
- Clause database: $w \vee \overline{x} \vee y \vee \overline{z}$ $\quad y \vee \overline{z}$

A closer look at CDCL

(Partial) input: $z \wedge (x \vee y) \wedge (\overline{u} \vee \overline{w}) \wedge (u \vee \overline{w} \vee \overline{x} \vee y \vee \overline{z})$

```
while not solved :
  unit_propagate()
  if conflict :
    learn()
    maybe_restart()
    backjump()
  else :
    assign_variable()
```

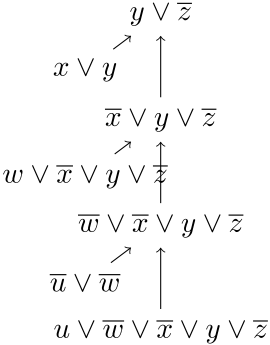
$z = 1 \quad z$



Next: $y = 1$ because of learned clause $y \vee \overline{z}$
Guaranteed by standard learning heuristic 1UIP

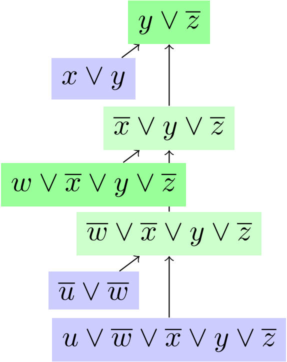
Proof system as annotated resolution

- Clauses as resolution DAG



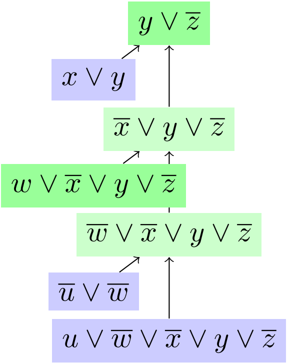
Proof system as annotated resolution

- Clauses as resolution DAG
- Grouped by sequences of input resolution



Proof system as annotated resolution

- Clauses as resolution DAG
- Grouped by sequences of input resolution
- Learned clauses allowed in later steps

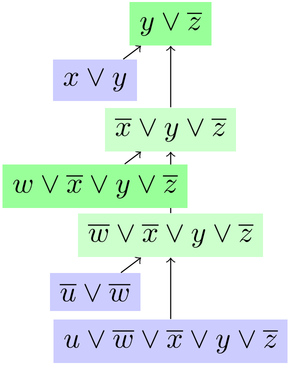


Proof system as annotated resolution

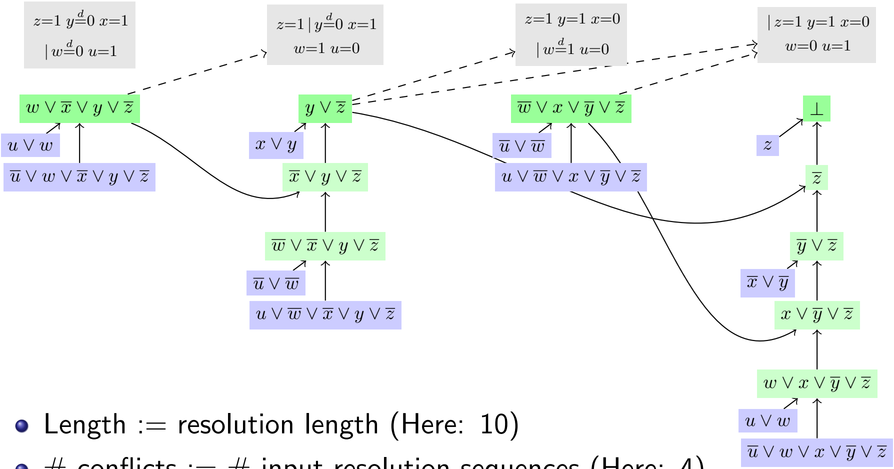
$z{=}1 \mid y \overset{d}{=} 0 \; x{=}1$

$w{=}1 \; u{=}0$

- Clauses as resolution DAG
- Grouped by sequences of input resolution
- Learned clauses allowed in later steps
- Branching sequence allows local checks



Standard measures



- Length := resolution length (Here: 10)
- # conflicts := # input resolution sequences (Here: 4)
- Space := database size (Here: 2)

Some facts

Theorem

CDCL proof system polynomially simulates resolution length

By [Pipatsrisawat, Darwiche '09], [Atserias, Fichte, Thurley '09]

Some facts

Theorem

CDCL proof system polynomially simulates resolution length

By [Pipatsrisawat, Darwiche '09], [Atserias, Fichte, Thurley '09]

Observation

CDCL proofs are valid resolution proofs

Hence all lower bounds on length, space and trade-offs apply

Length and space upper bounds

Worst case for CDCL proof system same as resolution

Proposition

Every formula has proofs in length $O(2^n)$ and space $O(n)$ simultaneously

Not surprising, but also not immediate

Length and space upper bounds

Worst case for CDCL proof system same as resolution

Proposition

Every formula has proofs in length $O(2^n)$ and space $O(n)$ simultaneously

Not surprising, but also not immediate

But can we simulate general resolution with respect to both length and space?

Length and space upper bounds

Worst case for CDCL proof system same as resolution

Proposition

Every formula has proofs in length $O(2^n)$ and space $O(n)$ simultaneously

Not surprising, but also not immediate

But can we simulate general resolution with respect to both length and space? Regular resolution?

Length and space upper bounds

Worst case for CDCL proof system same as resolution

Proposition

Every formula has proofs in length $O(2^n)$ and space $O(n)$ simultaneously

Not surprising, but also not immediate

But can we simulate general resolution with respect to both length and space? Regular resolution? Even tree-like resolution?

Length and space upper bounds

Worst case for CDCL proof system same as resolution

Proposition

Every formula has proofs in length $O(2^n)$ and space $O(n)$ simultaneously

Not surprising, but also not immediate

But can we simulate general resolution with respect to both length and space? Regular resolution? Even tree-like resolution?

Work in progress

Length-space trade-offs with restarts

Trade-offs from [Ben-Sasson, Nordström '11] also hold with

Theorem

There exists a family of formulas such that:

1. *There are short CDCL proofs*
2. *There are small CDCL proofs*
3. *Optimizing one measure blows up the other*

Length-space trade-offs with restarts

Trade-offs from [Ben-Sasson, Nordström '11] also hold with

Theorem

There exists a family of formulas such that:

1. *There are short CDCL proofs*
2. *There are small CDCL proofs*
3. *Optimizing one measure blows up the other* ✓

Proof sketch.

(3) immediate from resolution [BN'11]
Are there matching proofs in CDCL?

Length-space trade-offs with restarts

Trade-offs from [Ben-Sasson, Nordström '11] also hold with

Theorem

There exists a family of formulas such that:

1. *There are short CDCL proofs* ✓
2. *There are small CDCL proofs* ✓
3. *Optimizing one measure blows up the other* ✓

Proof sketch.

(3) immediate from resolution [BN'11]

Are there matching proofs in CDCL? Yes

Simulate resolution clause by clause, restart at every clause

Space preserved

Standard 1UIP learning heuristic □

Trade-offs without restarts

Theorem

There exists a specific family of formulas such that:

1. *There are CDCL proofs in space* $s = O(1)$
2. *There are CDCL proofs in length* $L = O(n^2/s)$
3. *Every proof requires length* $L = \Omega(n^2/s^2)$

Line of research investigating power of restarts

Upper bounds rely on restarts. Necessary?

Trade-offs without restarts

Theorem

There exists a specific family of formulas such that:

1. *There are non-restarting proofs in space* $s = O(1)$
2. *There are non-restarting proofs in length* $L = O(n^2/s)$
3. *Every proof requires length* $L = \Omega(n^2/s^2)$

Line of research investigating power of restarts

Upper bounds rely on restarts. Necessary? No
We craft explicit proofs without restarts for some families

Trade-offs without restarts

Theorem

There exists a specific family of formulas such that:

1. *There are non-restarting proofs in space* $s = O(1)$
2. *There are non-restarting proofs in length* $L = O(n^2/s)$
3. *Every proof requires length* $L = \Omega(n^2/s^2)$

Line of research investigating power of restarts

Upper bounds rely on restarts. Necessary? No
We craft explicit proofs without restarts for some families

Plausible for most results from [Ben-Sasson, Nordström '11] to follow
Technically involved, work in progress

More trade-offs?

Open: analogous results for the trade-offs in [Beame, Beck, Impagliazzo '12] and [Beck, Nordström, Tang '13]

- Conceivable for CDCL
- Less clear without restarts

Summary

- New CDCL proof system faithfully models:
  - Forgetting clauses
  - Restarts
  - Learning heuristics

Summary

- New CDCL proof system faithfully models:
  - Forgetting clauses
  - Restarts
  - Learning heuristics
- Some upper bounds & trade-offs
  - All resolution lower bounds

Summary

- New CDCL proof system faithfully models:
  - Forgetting clauses
  - Restarts
  - Learning heuristics
- Some upper bounds & trade-offs
  - All resolution lower bounds
- Open Problems:
  - Compare to resolution (general, regular, tree-like)
  - Separate general resolution / CDCL with no restarts and 1UIP

Summary

- New CDCL proof system faithfully models:
  - Forgetting clauses
  - Restarts
  - Learning heuristics
- Some upper bounds & trade-offs
  - All resolution lower bounds
- Open Problems:
  - Compare to resolution (general, regular, tree-like)
  - Separate general resolution / CDCL with no restarts and 1UIP

Thanks!