A variety of trimming techniques for pseudo-Boolean proof logs

Arthur GONTIER et al

September 13, 2025

Table of Contents

Proof logs

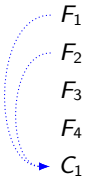
F_1

F_2

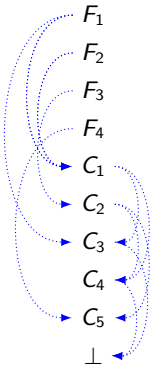
F_3

F_4

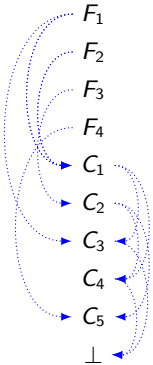
Proof logs

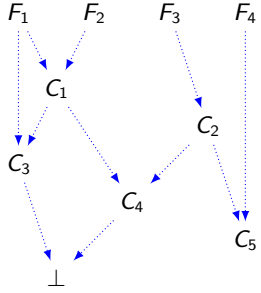


Proof logs

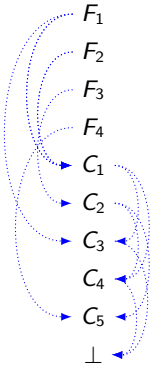


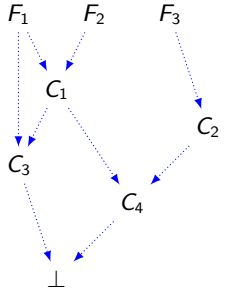
Proof logs





Proof logs





Looking for antecedents

pol F_1 3 * C_1 + s

Looking for antecedents

pol F_1 3 * C_1 + s

Antecedents: F_1, C_1

Looking for antecedents

pol F_1 3 * C_1 + s rup $ax + by + cz \geq r$

Antecedents: F_1, C_1

Looking for antecedents

pol F_1 3 * C_1 + s

Antecedents: F_1, C_1

rup $ax + by + cz \geq r$

Antecedents: All ctrs used in rup check

Looking for antecedents

pol F_1 3 * C_1 + s

Antecedents: F_1, C_1

rup $ax + by + cz \geq r$

Antecedents: All ctrs used in rup check

red $ax + by \geq r$: x 0 : subproof
  proofgoal C_1
    pol C_2 F_3 + s
    rup $0 \geq 1$
  qed : -1
  proofgoal C_2
    pol C_1 F_2 + s
    rup $0 \geq 1$
  qed : -1
qed red

Looking for antecedents

pol F_1 3 * C_1 + s

Antecedents: F_1, C_1

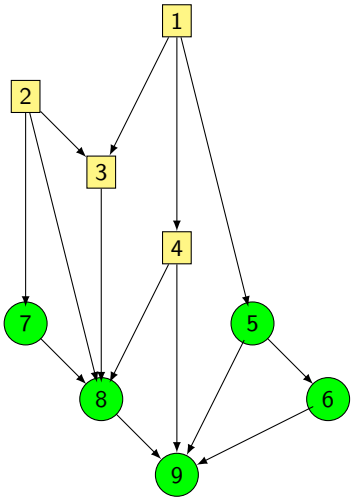
rup $ax + by + cz \geq r$

Antecedents: All ctrs used in rup check

red $ax + by \geq r$: x 0 : subproof
  proofgoal C_1
    pol C_2 F_3 + s
    rup $0 \geq 1$
  qed : -1
  proofgoal C_2
    pol C_1 F_2 + s
    rup $0 \geq 1$
  qed : -1
qed red

Antecedents: If proofgoal is marked then subproof antecedants are.

Precise deletion



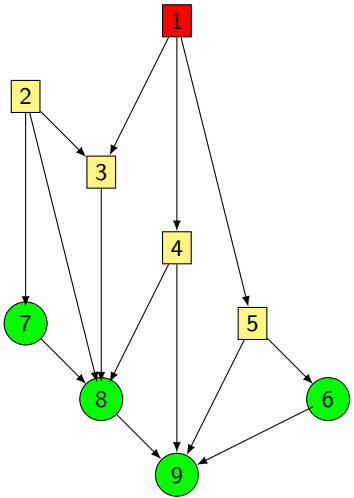
1

2

3

4

Precise deletion



1

2

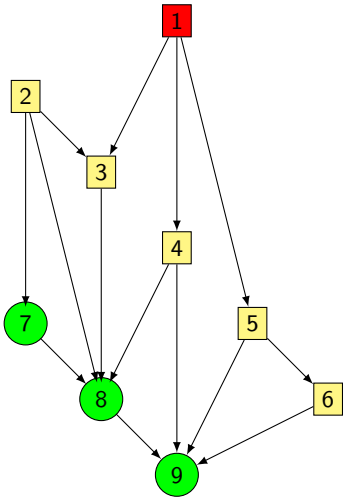
3

4

5

del id 1

Precise deletion



1

2

3

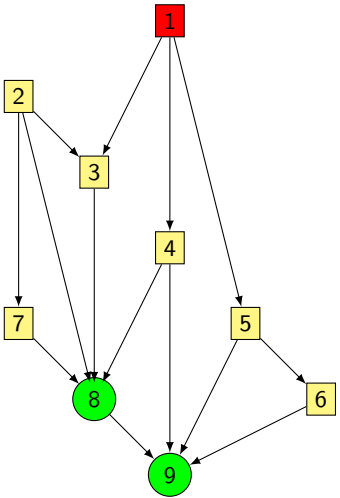
4

5

del id 1

6

Precise deletion



1

2

3

4

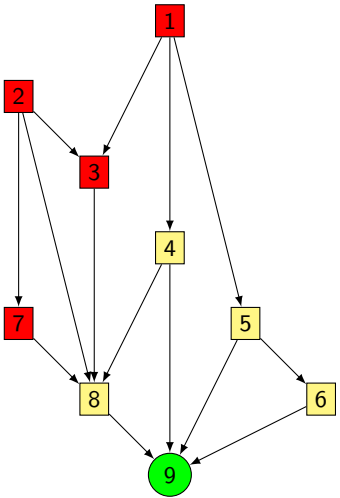
5

del id 1

6

7

Precise deletion



1

2

3

4

5

del id 1

6

7

8

del id 2 3 7

Cone-first Rup

Algorithm 1: rup

```
1 while no contradiction do
2     propagate()
```

Cone-first Rup

Algorithm 2: rup

```
1 while no contradiction do
2     propagate()
```

Algorithm 3: Cone-first rup

```
1 only_marked ← true
2 while no contradiction do
3     if only_marked then
4         only_marked ← try_propagate_marked()
5     else
6         propagate_one()
7         only_marked ← true
```

Table of Contents

Trimmer for faster proof checks

1) Decoration :
   - Produce decorated proofs
   - Forward pass on the proof
   - Compute pol lines
   - Add ctrs on deletions

Trimmer for faster proof checks

1) Decoration :

- Produce decorated proofs
- Forward pass on the proof
- Compute pol lines
- Add ctrs on deletions

Example :
pol F_1 3 * C_1 + s
e $ax + by + cz \geq r$
e $cz \geq r$
del F_2

Trimmer for faster proof checks

1) Decoration :
   - Produce decorated proofs
   - Forward pass on the proof
   - Compute pol lines
   - Add ctrs on deletions

2) Backward trimming :
   - Backward pass from the contradiction
   - Run cone-first rup
   - Mark the needed ctrs
   - Conditionally mark proofgoals

Example :
pol F_1 3 * C_1 + s
e $ax + by + cz \geq r$
e $cz \geq r$
del F_2

Trimmer for faster proof checks

1) Decoration :
   - Produce decorated proofs
   - Forward pass on the proof
   - Compute pol lines
   - Add ctrs on deletions

2) Backward trimming :
   - Backward pass from the contradiction
   - Run cone-first rup
   - Mark the needed ctrs
   - Conditionally mark proofgoals

3) Compaction
   - Forward pass
   - Writing elaborated proof
   - Mark needed proofgoals
   - Add deletions

Example :
pol F_1 3 * C_1 + s
e $ax + by + cz \geq r$
e $cz \geq r$
del F_2

Trimmer for faster proof checks

1) Decoration :
   - Produce decorated proofs
   - Forward pass on the proof
   - Compute pol lines
   - Add ctrs on deletions

2) Backward trimming :
   - Backward pass from the contradiction
   - Run cone-first rup
   - Mark the needed ctrs
   - Conditionally mark proofgoals

3) Compaction
   - Forward pass
   - Writing elaborated proof
   - Mark needed proofgoals
   - Add deletions

Example :
pol F_1 3 * C_1 + s
e $ax + by + cz \geq r$
e $cz \geq r$
del F_2

PS: If your solver can output the decorated proof or the proof only has pol lines, you can skip phase 1

Trimmer for faster proof checks

1) Decoration :
   - Produce decorated proofs
   - Forward pass on the proof
   - Compute pol lines
   - Add ctrs on deletions

2) Backward trimming :
   - Backward pass from the contradiction
   - Run cone-first rup
   - Mark the needed ctrs
   - Conditionally mark proofgoals

3) Compaction
   - Forward pass
   - Writing elaborated proof
   - Mark needed proofgoals
   - Add deletions

Example :
pol F_1 3 * C_1 + s
e $ax + by + cz \geq r$
e $cz \geq r$
del F_2

PS: If your solver can output the decorated proof or the proof only has pol lines, you can skip phase 1
PPS: We can write to file each intermediate proof to avoid mem out on big proofs.

State of development (Berhan)

Implemented :

- Cone-first rup
- All phases for pol and rup
- Full-ram
- Skip decoration option

State of development (Berhan)

Implemented :

- Cone-first rup
- All phases for pol and rup
- Full-ram
- Skip decoration option

Working on (by priority) :

- Redundance
- Optimisation (soli)
- Strengthening to core
- File option

State of development (Berhan)

Implemented :

- Cone-first rup
- All phases for pol and rup
- Full-ram
- Skip decoration option

Working on (by priority) :

- Redundance
- Optimisation (soli)
- Strengthening to core
- File option

End goal :

- Supporting full veriPB grammar
- Code optimisation
- PB trimming paper

Table of Contents

Trimming for smaller proofs

Use cases:

- Using proof as certificate
- Try understanding your problem

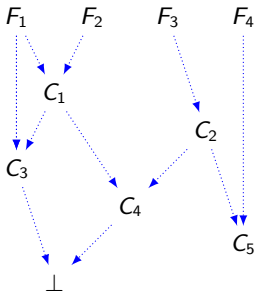
Trimming for smaller proofs

Use cases:

- Using proof as certificate
- Try understanding your problem

Philosophy:

- Take more time and resources to trim more
- Never trust Matthew



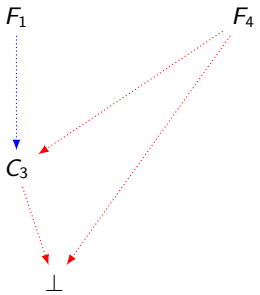
Trimming for smaller proofs

Use cases:

- Using proof as certificate
- Try understanding your problem

Philosophy:

- Take more time and resources to trim more
- Never trust Matthew



Do not trust Matthew's deletions

- A derived constraint may be useful later.

$C_{1.1}$
$C_{1.2}$
C_1
del id $C_{1.1}$ $C_{1.2}$
$C_{2.1}$
$C_{2.2}$
C_2
del id $C_{2.1}$ $C_{2.2}$
$C_{3.1}$
$C_{3.2}$
C_3
del id $C_{3.1}$ $C_{3.2}$
C_4
$\bot$

Do not trust Matthew's deletions

- A derived constraint may be useful later.

$C_{1.1}$
$C_{1.2}$
C_1
del id $C_{1.1}$ $C_{1.2}$
$C_{2.1}$
$C_{2.2}$
C_2
del id $C_{2.1}$ $C_{2.2}$
$C_{3.1}$
$C_{3.2}$
C_3
del id $C_{3.1}$ $C_{3.2}$
C_4
$\bot$

$C_{1.1}$
$C_{1.2}$
C_1
C_2
C_3
del id $C_{1.1}$ $C_{1.2}$
C_4
$\bot$

Do not trust Matthew's deletions

- A derived constraint may be useful later.
- Be careful with redundance; they may need prior deletions to work.

```
red ¬y ≥ 1 :  y 0
del id -1
red y ≥ 1 :  y 1
```

$C_{1.1}$
$C_{1.2}$
C_1
del id $C_{1.1}$ $C_{1.2}$
$C_{2.1}$
$C_{2.2}$
C_2
del id $C_{2.1}$ $C_{2.2}$
$C_{3.1}$
$C_{3.2}$
C_3
del id $C_{3.1}$ $C_{3.2}$
C_4
$\bot$

$C_{1.1}$
$C_{1.2}$
C_1
C_2
C_3
del id $C_{1.1}$ $C_{1.2}$
C_4
$\bot$

Do not trust Matthew's deletions

- A derived constraint may be useful later.
- Be careful with redundance; they may need prior deletions to work.

```
red ¬y ≥ 1 :  y 0
del id -1
red y ≥ 1 :  y 1
```

- Fine if you don't reuse the names

$C_{1.1}$
$C_{1.2}$
C_1
del id $C_{1.1}$ $C_{1.2}$
$C_{2.1}$
$C_{2.2}$
C_2
del id $C_{2.1}$ $C_{2.2}$
$C_{3.1}$
$C_{3.2}$
C_3
del id $C_{3.1}$ $C_{3.2}$
C_4
$\bot$

$C_{1.1}$
$C_{1.2}$
C_1
C_2
C_3
del id $C_{1.1}$ $C_{1.2}$
C_4
$\bot$

Better rup for trimming ?

- Cone-first

Better rup for trimming ?

- Cone-first
- Heuristic prioritizing the highest ctrs

Better rup for trimming ?

- Cone-first
- Heuristic prioritizing the highest ctrs
- Conflict analysis to get smaller antecedent set

Better rup for trimming ?

- Cone-first
- Heuristic prioritizing the highest ctrs
- Conflict analysis to get smaller antecedent set

Algorithm 4: rup

```
1 only_m ← true
2 while no contradiction do
3     if only_m then
4         only_m ← try_prop_m_ordered()
5     else
6         prop_one_ordered()
7         only_m ← true
8     update_implG(lit,lvl,ctrID)
9 MarkAntecedantsFromConflictAnalysis()
```

Better rup for trimming ?

- Cone-first
- Heuristic prioritizing the highest ctrs
- Conflict analysis to get smaller antecedent set

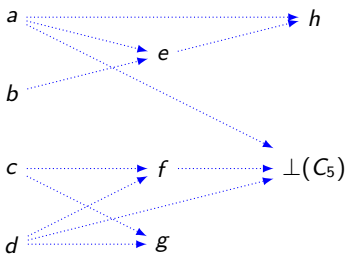
Algorithm 5: rup

```
1 only_m ← true
2 while no contradiction do
3     if only_m then
4         only_m ← try_prop_m_ordered()
5     else
6         prop_one_ordered()
7         only_m ← true
8     update_implG(lit,lvl,ctrID)
9 MarkAntecedantsFromConflictAnalysis()
```

Algorithm 6: MAFCA

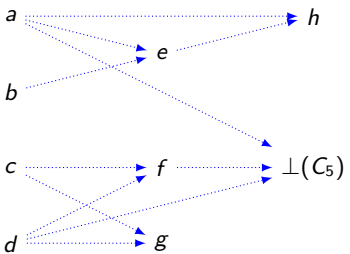
```
1  enqueue(PQ,⊥,1)
2  while not_empty(PQ) do
3      lit ← dequeue(PQ)
4      id ← implG[lit]
5      mark(id)
6      lits ← contributing_lits(id)
7      sort(lits,by:order)
8      for l ∈ lits do
9          if propagation then
10             break
11         else
12             set_lit(l,false)
13             enqueue(PQ,l,lvl)
```

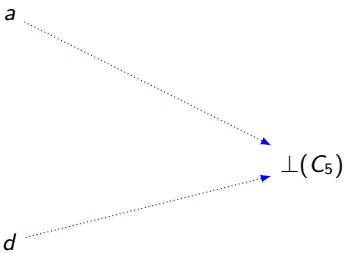
Conflict analysis example



$$\neg a + \neg d + \neg f \geq 2 \qquad (C_5)$$

Conflict analysis example





$$\neg a + \neg d + \neg f \geq 2 \qquad (C_5)$$

Width trimming

- Idea: weaken unused literals in equations

Width trimming

- Idea: weaken unused literals in equations

$$a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+10x+y+z \geq 21$$

Width trimming

- Idea: weaken unused literals in equations

$$a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+10x+y+z \geq 21$$

Width trimming

- Idea: weaken unused literals in equations

$$a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+10x+y+z \geq 21$$

$$a+b+c+d+e+10x \geq 1$$

Width trimming

- Idea: weaken unused literals in equations

$$a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+10x+y+z \geq 21$$

$$a + b + c + d + e + 10x \geq 1$$

- How to propagate this conelits through pol lines ?

Width trimming

- Idea: weaken unused literals in equations

$$a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+10x+y+z \geq 21$$

$$a + b + c + d + e + 10x \geq 1$$

- How to propagate this conelits through pol lines ?

$$a + \neg b + \neg c + d + e + 10x \geq 5 \quad (1)$$
$$2b + 2x \geq 1 \quad (2)$$
$$c + x \geq 1 \quad (3)$$
`pol 1 2 3 + +` (4)
`e` $a + b + d + e + 13x \geq 5$ (5)

Width trimming

- Idea: weaken unused literals in equations

$$a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+10x+y+z \geq 21$$

$$a+b+c+d+e+10x \geq 1$$

- How to propagate this conelits through pol lines ?

$$a+\neg b+\neg c+d+e+10x \geq 5 \quad (1)$$

$$2b+2x \geq 1 \quad (2)$$

$$c+x \geq 1 \quad (3)$$

`pol 1 2 3 + +` (4)

`e` $a+b+d+e+13x \geq 5$ (5)

$$conelits \cup (poslits \cap neglits)$$

Width trimming

- Idea: weaken unused literals in equations

$$a+b+c+d+e+f+g+h+i+j+k+l+m+n+o+p+q+r+s+t+u+v+w+10x+y+z \geq 21$$

$$a + b + c + d + e + 10x \geq 1$$

- How to propagate this conelits through pol lines ?

$$a + \neg b + \neg c + d + e + 10x \geq 5 \quad (1)$$
$$2b + 2x \geq 1 \quad (2)$$
$$c + x \geq 1 \quad (3)$$
`pol 1 2 3 + +` (4)
`e` $a + b + d + e + 13x \geq 5$ (5)

$$\mathit{conelits} \cup (\mathit{poslits} \cap \mathit{neglits})$$

- What about saturation and division ?

Visualisations

The graphs do not fit in the margin

Justification

1 Log proof skeleton

F_1

F_2

assert C_1

assert C_2

assert C_3

assert C_4

C_5

C_6

$\bot$

Justification

1. Log proof skeleton
2. Trim proof skeleton

F_1
F_2
assert C_1
assert C_2
assert C_3
assert C_4
C_5
C_6
$\bot$

F_1

assert C_2

assert C_4
C_5

$\bot$

Justification

1. Log proof skeleton
2. Trim proof skeleton
3. Justify to get a full proof
4. Normal trimming and/or checking

F_1
F_2
assert C_1
assert C_2
assert C_3
assert C_4
C_5
C_6
$\bot$

F_1

assert C_2

assert C_4
C_5

$\bot$

F_1
$C_{2.1}$
$C_{2.2}$
$C_{2.3}$
$C_{2.4}$
$C_{2.5}$
$C_{2.6}$
C_2
$C_{4.1}$
$C_{4.2}$
$C_{4.3}$
C_4
C_5
$\bot$

State of development (Arthur)

Implemented :

- Cone-first rup with conflict analysis, and heuristics.
- Support pol, rup, red and optimisation (soli)
- Full-ram (max 3T)

State of development (Arthur)

Implemented :

- Cone-first rup with conflict analysis, and heuristics.
- Support pol, rup, red and optimisation (soli)
- Full-ram (max 3T)

Working on (by priority) :

- Width trimming (weakening unused literals)
- Visualisations
- Justifications

State of development (Arthur)

Implemented :

- Cone-first rup with conflict analysis, and heuristics.
- Support pol, rup, red and optimisation (soli)
- Full-ram (max 3T)

Working on (by priority) :

- Width trimming (weakening unused literals)
- Visualisations
- Justifications

End goal :

- Get proofs to be as small as possible
- Get satisfactory problem insight from small proofs
- PB trimming paper

Table of Contents

Some funny instances (These comparison are unfair)

	sizes			times (s)		
Instance	veriPB	fastPB	smallPB	veriPB	fastPB	smallPB (parse trim write verif)
LVg2g80	5.219 MB	6.455 MB	414.1 KB	44.2	12.5	100 (2.7 96.5 0.16 0.86)

Some funny instances (These comparison are unfair)

	sizes			times (s)		
Instance	veriPB	fastPB	smallPB	veriPB	fastPB	smallPB (parse trim write verif)
LVg3g17	1.665 MB	499.4 KB	643.8 KB	0.46	0.79	1.19 (0.21 0.66 0.02 0.3)
bio027085	2.922 MB	333.2 KB	75.15 KB	4.28	32.4	3.7 (2.35 0.98 0.02 0.35)
LVg2g80	5.219 MB	6.455 MB	414.1 KB	44.2	12.5	100 (2.7 96.5 0.16 0.86)

Some funny instances (These comparison are unfair)

	sizes			times (s)		
Instance	veriPB	fastPB	smallPB	veriPB	fastPB	smallPB (parse trim write verif)
LVg3g17	1.665 MB	499.4 KB	643.8 KB	0.46	0.79	1.19 (0.21 0.66 0.02 0.3)
bio027085	2.922 MB	333.2 KB	75.15 KB	4.28	32.4	3.7 (2.35 0.98 0.02 0.35)
LVg2g80	5.219 MB	6.455 MB	414.1 KB	44.2	12.5	100 (2.7 96.5 0.16 0.86)
LVg3g41	8.664 MB	6.284 MB	421.1 KB	45.3	46.4	261 (4.72 245 1.02 9.94)

Some funny instances (These comparison are unfair)

	sizes			times (s)		
Instance	veriPB	fastPB	smallPB	veriPB	fastPB	smallPB (parse trim write verif)
LVg3g17	1.665 MB	499.4 KB	643.8 KB	0.46	0.79	1.19 (0.21 0.66 0.02 0.3)
bio027085	2.922 MB	333.2 KB	75.15 KB	4.28	32.4	3.7 (2.35 0.98 0.02 0.35)
LVg2g80	5.219 MB	6.455 MB	414.1 KB	44.2	12.5	100 (2.7 96.5 0.16 0.86)
LVg3g41	8.664 MB	6.284 MB	421.1 KB	45.3	46.4	261 (4.72 245 1.02 9.94)
bio170075	18.54 MB	14.15 MB	585.5 KB	192.0	173	571 (15.3 523 2.26 30.8)

Some funny instances (These comparison are unfair)

	sizes			times (s)		
Instance	veriPB	fastPB	smallPB	veriPB	fastPB	smallPB (parse trim write verif)
LVg3g17	1.665 MB	499.4 KB	643.8 KB	0.46	0.79	1.19 (0.21 0.66 0.02 0.3)
bio027085	2.922 MB	333.2 KB	75.15 KB	4.28	32.4	3.7 (2.35 0.98 0.02 0.35)
LVg2g80	5.219 MB	6.455 MB	414.1 KB	44.2	12.5	100 (2.7 96.5 0.16 0.86)
LVg3g41	8.664 MB	6.284 MB	421.1 KB	45.3	46.4	261 (4.72 245 1.02 9.94)
bio170075	18.54 MB	14.15 MB	585.5 KB	192.0	173	571 (15.3 523 2.26 30.8)
LVg53g56	46.81 MB	148.0 MB	12.89 MB	7.93	22.5	48.8 (10.2 32.9 1.95 3.84)

Conclusion

Merci pour votre attention !

Questions ?