Unsatisfiability Proofs

Henrich Lauko

IA072 – Seminar on Concurrency

November 30, 2016

Is SAT solver credible?

Possible results:

- SAT
- UNSAT
- TIMEOUT

Criteria on unsatisfiability proofs

- easily checkable

Criteria on unsatisfiability proofs

- easily checkable
- small/reasonable size

Criteria on unsatisfiability proofs

- easily checkable
- small/reasonable size
- proof checking quicker than solver

Criteria on unsatisfiability proofs

- easily checkable
- small/reasonable size
- proof checking quicker than solver
- proof production not slowing down the solver

Criteria on unsatisfiability proofs

- easily checkable
- small/reasonable size
- proof checking quicker than solver
- proof production not slowing down the solver
- minimize modifications of solver

Resolution

Resolution rule:

$$\frac{x \vee C \qquad \neg x \vee D}{C \vee D}\, x$$

We write $(x \vee C) \diamond (\neg x \vee D) = (C \vee D)$.

Resolution

Resolution rule:

$$\frac{x \lor C \qquad \neg x \lor D}{C \lor D}\, x$$

We write $(x \lor C) \diamond (\neg x \lor D) = (C \lor D)$.

Resolution chain

$$((a \lor c) \diamond (\neg a \lor b)) \diamond (\neg a \lor \neg b) = (\neg a \lor c)$$

Resolution chains are computed from left.

Resolution proofs – TraceCheck proof format

$$\varphi = (\neg b \vee c) \wedge (a \vee c) \wedge (\neg a \vee b) \wedge (\neg a \vee \neg b) \wedge (a \vee \neg b) \wedge (b \vee \neg c)$$

Resolution proofs – TraceCheck proof format

$$\varphi = (\neg b \vee c) \wedge (a \vee c) \wedge (\neg a \vee b) \wedge (\neg a \vee \neg b) \wedge (a \vee \neg b) \wedge (b \vee \neg c)$$

$$(\neg a \vee \neg b) \diamond (a \vee \neg b) = \neg b$$

$$(a \vee c) \diamond (\neg a \vee b) \diamond (\neg b \vee c) = c$$

$$(b \vee \neg c) \diamond (\neg b) \diamond (c) = \emptyset$$

Resolution proofs – TraceCheck proof format

$$\varphi = (\neg b \vee c) \wedge (a \vee c) \wedge (\neg a \vee b) \wedge (\neg a \vee \neg b) \wedge (a \vee \neg b) \wedge (b \vee \neg c)$$

$$(\neg a \vee \neg b) \diamond (a \vee \neg b) = \neg b$$

$$(a \vee c) \diamond (\neg a \vee b) \diamond (\neg b \vee c) = c$$

$$(b \vee \neg c) \diamond (\neg b) \diamond (c) = \emptyset$$

1	-2	3	0	**0**		
2	1	3	0	**0**		
3	-1	2	0	**0**		
4	-1	-2	0	**0**		
5	1	-2	0	**0**		
6	2	-3	0	**0**		
7	-2	0	**4**	**5**	**0**	
8	3	0	**1**	**2**	**3**	**0**
9	0	**6**	**7**	**8**	**0**	

Imperfections of resolution proofs

1	-2	3	0	**0**		
2	1	3	0	**0**		
3	-1	2	0	**0**		
4	-1	-2	0	**0**		
5	1	-2	0	**0**		
6	2	-3	0	**0**		

7	-2	0	**4**	**5**	**0**	
8	3	0	**1**	**2**	**3**	**0**
9	0	**6**	**7**	**8**	**0**	

Imperfections of resolution proofs

1	-2	3	0	**0**		
2	1	3	0	**0**		
3	-1	2	0	**0**		
4	-1	-2	0	**0**		
5	1	-2	0	**0**		
6	2	-3	0	**0**		
7	-2	0	**4**	**5**	**0**	
8	3	0	**1**	**2**	**3**	**0**
9	0	**6**	**7**	**8**	**0**	

- extremely large (dozens of gigabytes)

Imperfections of resolution proofs

1	-2	3	0	**0**		
2	1	3	0	**0**		
3	-1	2	0	**0**		
4	-1	-2	0	**0**		
5	1	-2	0	**0**		
6	2	-3	0	**0**		
7	-2	0	**4**	**5**	**0**	
8	3	0	**1**	**2**	**3**	**0**
9	0	**6**	**7**	**8**	**0**	

- extremely large (dozens of gigabytes)
- hard to modify solver

Imperfections of resolution proofs

1	-2	3	0	**0**		
2	1	3	0	**0**		
3	-1	2	0	**0**		
4	-1	-2	0	**0**		
5	1	-2	0	**0**		
6	2	-3	0	**0**		
7	-2	0	**4**	**5**	**0**	
8	3	0	**1**	**2**	**3**	**0**
9	0	**6**	**7**	**8**	**0**	

- extremely large (dozens of gigabytes)
- hard to modify solver
- computationaly hard for solver to find correct order of resolutions and determine the clauses on which to apply resolution

Imperfections of resolution proofs

```
1  -2   3  0  0
2   1   3  0  0
3  -1   2  0  0
4  -1  -2  0  0
5   1  -2  0  0
6   2  -3  0  0

7  -2   0  4  5  0
8   3   0  1  2  3  0
9   0   6  7  8  0
```

- extremely large (dozens of gigabytes)
- hard to modify solver
- computationaly hard for solver to find correct order of resolutions and determine the clauses on which to apply resolution
- but easy to check – in deterministic log space

Looking for better proofs

2003 – Goldberg and Novikov introduced **Clausal proofs**

Clausal proof

Clausal proof P is represented as queue of lemmas $l_1, \ldots, l_n$, where $l_n = \emptyset$.

Looking for better proofs

2003 – Goldberg and Novikov introduced **Clausal proofs**

Clausal proof

Clausal proof P is represented as queue of lemmas $l_1, \ldots, l_n$, where $l_n = \emptyset$.

- We want lemmas to be implied by φ, because then $BCP(\varphi \wedge \neg l)$ produces empty clause $\emptyset$.

Reverse unit propagation clause (RUP)

Unit clause and **Unit propagation**

- Given a formula φ, if unit propagation on φ produces $\emptyset$, then φ is unsatisfiable.

Reverse unit propagation clause (RUP)

Unit clause and **Unit propagation**

- Given a formula φ, if unit propagation on φ produces $\emptyset$, then φ is unsatisfiable.

Reverse unit propagation clause

Let $C = (l_1 \vee l_2 \cdots \vee l_n)$ and $\neg C = (\neg l_1) \wedge (\neg l_2) \wedge \cdots \wedge (\neg l_n)$.
Then C is RUP clause with respect to φ, if $\varphi \wedge \neg C \vdash_1 \emptyset$.

- *Reverse* because unit clauses $\neg C$ are propagated back into earlier clauses.
- Typical RUP clauses are the learned clauses in CDCL.

RUP format

RUP proof

Given a fomula φ, a clausal proof $P = \{l_1, \ldots, l_n\}$ is a valid RUP proof for φ if $l_n = \emptyset$ and for all l_i holds that:

$$\varphi \wedge l_1 \wedge \cdots \wedge l_{i-1} \wedge \neg l_i \;\vdash_1\; \emptyset$$

RUP format

RUP proof

Given a fomula φ, a clausal proof $P = \{l_1, \ldots, l_n\}$ is a valid RUP proof for φ if $l_n = \emptyset$ and for all l_i holds that:

$$\varphi \wedge l_1 \wedge \cdots \wedge l_{i-1} \wedge \neg l_i \;\vdash_1\; \emptyset$$

$$\varphi = (\neg b \vee c) \wedge (a \vee c) \wedge (\neg a \vee b) \wedge (\neg a \vee \neg b) \wedge (a \vee \neg b) \wedge (b \vee \neg c)$$

Clausal proof (RUP)

```
-2  0
 3  0
 0
```

$$P_\varphi := \{(\neg b), (c), \emptyset\}$$

$$\varphi \wedge (b) \qquad \vdash_1 \emptyset$$

$$\varphi \wedge (\neg b) \wedge (\neg c) \qquad \vdash_1 \emptyset$$

$$\varphi \wedge (\neg b) \wedge (c) \wedge (\neg\emptyset) \qquad \vdash_1 \emptyset$$

RUP checking

```
RUPchecker(CNF formula φ, queue Q of lemmas)
  while Q is not empty
    l ← Q.dequeue()
    φ′ ← BCP(φ ∧ ¬l)
    if (∅ ∉ φ′) then
      return "checking failed"
    φ ← BCP(φ ∧ l)
    if (∅ ∈ φ) then
      return UNSAT
  return "all lemmas validated"
```

```
BCP(CNF formula φ)
  while ∃(x) ∈ φ
    for c ∈ φ with ¬x ∈ c
      c ← c \ {¬x}
    for c ∈ φ with x ∈ c
      φ ← φ \ {c}
  return F
```

RUP summary

Pros

- significantly smaller
- minor modifications to solver

Cons

- expensive checking
- complex algorithms for checking

RUP summary

Pros

- significantly smaller
- minor modifications to solver

Cons

- expensive checking
- complex algorithms for checking

DRUP – Delete Reverse Unit Propagtion

Format extends RUP by integrating clause deletion information into proofs.

Generalization of clausal proofs

- Clauses in clausal proof are **redundant** clauses.

Generalization of clausal proofs

- Clauses in clausal proof are **redundant** clauses.

Redundancy properties

1. tautology (T)

Generalization of clausal proofs

- Clauses in clausal proof are **redundant** clauses.

Redundancy properties

1. tautology (T)
2. asymmetric tautology (AT) – if ALA(φ, C) has property T

Asymmetric literal addition (ALA)

ALA(φ, C) computes C until fixpoint as follows, if $l_1, \ldots, l_k \in C$ and there is a clause $(l_1 \vee \cdots \vee l_k \vee l) \in \varphi \setminus \{C\}$ for some literal l, let $C := C \vee \neg l$.

Generalization of clausal proofs

- Clauses in clausal proof are **redundant** clauses.

Redundancy properties

1. tautology (T)
2. asymmetric tautology (AT) – if $\mathrm{ALA}(\varphi, C)$ has property T
3. resolution tautology (RT) $=$ blocked clauses

Asymmetric literal addition (ALA)

$\mathrm{ALA}(\varphi, C)$ computes C until fixpoint as follows, if $l_1, \ldots, l_k \in C$ and there is a clause $(l_1 \vee \cdots \vee l_k \vee l) \in \varphi \setminus \{C\}$ for some literal l, let $C := C \vee \neg l$.

Resolution property RP

(i) C has property P or (ii) There is a literal $l \in \varphi$ such that for each clause $C' \in \varphi$ with $\neg l \in C'$, each resolvent $C \diamond C'$ has P.

Generalization of clausal proofs

- Clauses in clausal proof are **redundant** clauses.

Redundancy properties

1. tautology (T)
2. asymmetric tautology (AT) – if $\text{ALA}(\varphi, C)$ has property T
3. resolution tautology (RT) = blocked clauses
4. resolution asymmetric tautology (RAT)

Asymmetric literal addition (ALA)

$\text{ALA}(\varphi, C)$ computes C until fixpoint as follows, if $l_1, \ldots, l_k \in C$ and there is a clause $(l_1 \vee \cdots \vee l_k \vee l) \in \varphi \setminus \{C\}$ for some literal l, let $C := C \vee \neg l$.

Resolution property RP

(i) C has property P or (ii) There is a literal $l \in \varphi$ such that for each clause $C' \in \varphi$ with $\neg l \in C'$, each resolvent $C \diamond C'$ has P.

Example

$$\varphi = (a \lor b) \;\land\; (b \lor c) \;\land\; (\neg b \lor \neg c)$$

- $(a \lor \neg a)$

Example

$$\varphi = (a \lor b) \ \land \ (b \lor c) \ \land \ (\neg b \lor \neg c)$$

- $(a \lor \neg a)$
  - has T

Example

$$\varphi = (a \vee b) \ \wedge \ (b \vee c) \ \wedge \ (\neg b \vee \neg c)$$

- $(a \vee \neg a)$
  - has T
- $(a \vee \neg c)$

Example

$$\varphi = (a \vee b) \;\wedge\; (b \vee c) \;\wedge\; (\neg b \vee \neg c)$$

- $(a \vee \neg a)$
  - has T
- $(a \vee \neg c)$
  - does not have T

Example

$$\varphi = (a \vee b) \;\wedge\; (b \vee c) \;\wedge\; (\neg b \vee \neg c)$$

- $(a \vee \neg a)$
  - has T
- $(a \vee \neg c)$
  - does not have T
  - has AT because unit propagation under $(\neg a \wedge c)$ results in conflict

Example

$$\varphi = (a \vee b) \;\wedge\; (b \vee c) \;\wedge\; (\neg b \vee \neg c)$$

- $(a \vee \neg a)$
  - has T
- $(a \vee \neg c)$
  - does not have T
  - has AT because unit propagation under $(\neg a \wedge c)$ results in conflict
  - has RT on literal a, because φ contains no clauses with $\neg a$

Example

$$\varphi = (a \vee b) \;\wedge\; (b \vee c) \;\wedge\; (\neg b \vee \neg c)$$

- $(a \vee \neg a)$
  - has T
- $(a \vee \neg c)$
  - does not have T
  - has AT because unit propagation under $(\neg a \wedge c)$ results in conflict
  - has RT on literal a, because φ contains no clauses with $\neg a$

Example

$$\varphi = (a \vee b) \ \wedge \ (b \vee c) \ \wedge \ (\neg b \vee \neg c)$$

- $(a \vee \neg a)$
  - has T
- $(a \vee \neg c)$
  - does not have T
  - has AT because unit propagation under $(\neg a \wedge c)$ results in conflict
  - has RT on literal a, because φ contains no clauses with $\neg a$
- $(\neg a \vee c)$

Example

$$\varphi = (a \vee b) \;\wedge\; (b \vee c) \;\wedge\; (\neg b \vee \neg c)$$

- $(a \vee \neg a)$
  - has T
- $(a \vee \neg c)$
  - does not have T
  - has AT because unit propagation under $(\neg a \wedge c)$ results in conflict
  - has RT on literal a, because φ contains no clauses with $\neg a$
- $(\neg a \vee c)$
  - does not have T

Example

$$\varphi = (a \lor b) \ \land \ (b \lor c) \ \land \ (\neg b \lor \neg c)$$

- $(a \lor \neg a)$
  - has T
- $(a \lor \neg c)$
  - does not have T
  - has AT because unit propagation under $(\neg a \land c)$ results in conflict
  - has RT on literal a, because φ contains no clauses with $\neg a$
- $(\neg a \lor c)$
  - does not have T
  - does not have AT

Example

$$\varphi = (a \vee b) \;\wedge\; (b \vee c) \;\wedge\; (\neg b \vee \neg c)$$

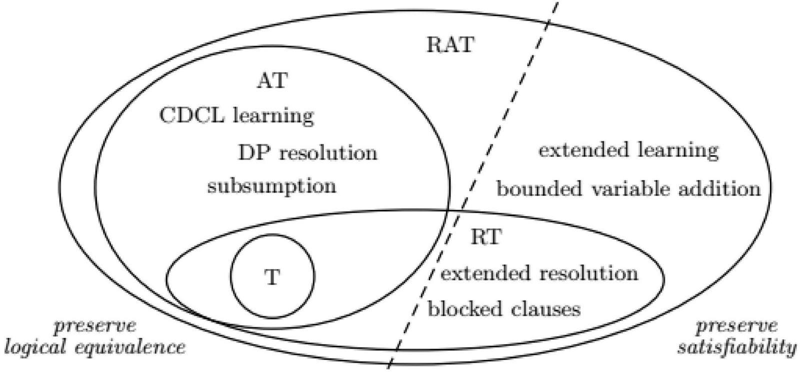
- $(a \vee \neg a)$
  - has T
- $(a \vee \neg c)$
  - does not have T
  - has AT because unit propagation under $(\neg a \wedge c)$ results in conflict
  - has RT on literal a, because φ contains no clauses with $\neg a$
- $(\neg a \vee c)$
  - does not have T
  - does not have AT
  - does not have RT, there is no tautological resolvent on $\neg a$ on $(a \vee b)$ and for c on $(\neg b \vee \neg c)$

Example

$$\varphi = (a \vee b) \;\wedge\; (b \vee c) \;\wedge\; (\neg b \vee \neg c)$$

- $(a \vee \neg a)$
  - has T
- $(a \vee \neg c)$
  - does not have T
  - has AT because unit propagation under $(\neg a \wedge c)$ results in conflict
  - has RT on literal a, because φ contains no clauses with $\neg a$
- $(\neg a \vee c)$
  - does not have T
  - does not have AT
  - does not have RT, there is no tautological resolvent on $\neg a$ on $(a \vee b)$ and for c on $(\neg b \vee \neg c)$
  - has RAT, because $(\neg a \vee c) \diamond (a \vee b) = (b \vee c)$ and unit propagation on $(\neg b \wedge \neg c)$ results in conflict

More redundancy properties



Extended resolution

Extension rule

Allows to iteratively add definitions of form $x := a \wedge b$ by adding caluses $(x \vee \neg a \vee \neg b) \wedge (\neg x \vee a) \wedge (\neg x \vee v)$.

Blocked clauses

Blocked clause

Given formula φ, a clause C, and literal $l \in C$, the literal l blocks C with respect to φ if:

1. for each clause $D \in \varphi$ with $\neg l \in D$, $C \diamond_l D$ is tautology, or
2. $\neg l \in C$, i.e., C itself is tautology.

Blocked clauses

Blocked clause

Given formula φ, a clause C, and literal $l \in C$, the literal l blocks C with respect to φ if:

1. for each clause $D \in \varphi$ with $\neg l \in D$, $C \diamond_l D$ is tautology, or
2. $\neg l \in C$, i.e., C itself is tautology.

Example

$\varphi = (\neg b \vee c) \wedge (a \vee c) \wedge (\neg a \vee b) \wedge (\neg a \vee \neg b) \wedge (a \vee \neg b) \wedge (b \vee \neg c)$

$(\neg b \vee c)$ is blocked on c, because $(\neg b \vee c) \diamond_c (b \vee \neg c) = (\neg b \vee b)$.

Since φ is unsatisfiable, $\varphi \setminus \{(\neg b \vee c)\}$ must be unsatisfiable.

Blocked clauses

Blocked clause

Given formula φ, a clause C, and literal $l \in C$, the literal l blocks C with respect to φ if:

1. for each clause $D \in \varphi$ with $\neg l \in D$, $C \diamond_l D$ is tautology, or
2. $\neg l \in C$, i.e., C itself is tautology.

Example

$\varphi = (\neg b \vee c) \wedge (a \vee c) \wedge (\neg a \vee b) \wedge (\neg a \vee \neg b) \wedge (a \vee \neg b) \wedge (b \vee \neg c)$

$(\neg b \vee c)$ is blocked on c, because $(\neg b \vee c) \diamond_c (b \vee \neg c) = (\neg b \vee b)$.

Since φ is unsatisfiable, $\varphi \setminus \{(\neg b \vee c)\}$ must be unsatisfiable.

Blocked clause addition is generalization of extended resolution. May add clauses not logically implied by the formula.

Resolution Asymmetric Tautologies (RAT)

Resolution asymmetric tautology

Clause C has RAT on l with respect to φ if for all $D \in \varphi$ with $\neg l \in D$ holds that

$$\varphi \wedge \neg C \wedge (\neg D \setminus (l)) \vdash_1 \emptyset$$

Resolution Asymmetric Tautologies (RAT)

Resolution asymmetric tautology

Clause C has RAT on l with respect to φ if for all $D \in \varphi$ with $\neg l \in D$ holds that

$$\varphi \wedge \neg C \wedge (\neg D \setminus (l)) \vdash_1 \emptyset$$

RAT is generalization of RUP clauses:

$$\varphi \wedge \neg C \vdash_1 \emptyset \implies \varphi \wedge \neg C \wedge (\neg D \setminus (l)) \vdash_1 \emptyset$$

Resolution Asymmetric Tautologies (RAT)

Resolution asymmetric tautology

Clause C has RAT on l with respect to φ if for all $D \in \varphi$ with $\neg l \in D$ holds that

$$\varphi \wedge \neg C \wedge (\neg D \setminus (l)) \vdash_1 \emptyset$$

RAT is generalization of RUP clauses:

$$\varphi \wedge \neg C \vdash_1 \emptyset \implies \varphi \wedge \neg C \wedge (\neg D \setminus (l)) \vdash_1 \emptyset$$

and, if clause C is blocked on l, then for all $D \in \varphi$ with $\neg l \in D$ holds that C contains a literal $k \neq l$ such that $\neg k \in D$, so:

$$\varphi \wedge (k) \wedge (\neg k) \vdash_1 \emptyset \implies \varphi \wedge \neg C \wedge (\neg D \setminus (l)) \vdash_1 \emptyset$$

Excursion to bounded variable addition

Bounded variable addition

Adds new variable to express dependenties of variables in clauses and potentially shrinks the formula.

Example

$\varphi = (\neg a \vee \neg b \vee \neg c) \wedge (a \vee d) \wedge (a \vee e) \wedge (b \vee d) \wedge (b \vee e) \wedge (c \vee d)$

BVA introduces a new variable f.

$\varphi = (f \vee a) \wedge (f \vee b) \wedge (f \vee c) \wedge (\neg f \vee d) \wedge (\neg f \vee e)$

- Cannot be expressed only with resolution steps.

Proofs with extended resolution – RAT and DRAT

Extends RUP format with RAT lemmas.

$$\varphi = (\neg a \lor \neg b \lor \neg c) \land (a \lor d) \land (a \lor e) \land (b \lor d) \land (b \lor e) \land (c \lor d) \land (c \lor e) \land (\neg d \lor \neg e)$$

- Uses BVA to replace first six clauses by five new cluses using a fresh new variable
- New clauses are RAT clauses.
- Final proof is only $\{(f), \emptyset\}$.

```
     6  1  0
     6  2  0
     6  3  0
    -6  4  0
    -6  5  0
d    1  4  0
d    2  4  0
d    3  4  0
d    1  5  0
d    2  5  0
d    3  5  0
        6  0
           0
```

RAT checking

Resolution asymmetric tautology

Clause C has RAT on l with respect to φ if for all $D \in \varphi$ with $\neg l \in D$ holds that

$$\varphi \wedge \neg C \wedge (\neg D \setminus (l)) \vdash_1 \emptyset$$

```
RATchecker(CNF formula φ, queue Q of lemmas)
  while Q is not empty
    L ← Q.dequeue()
    if ∅ ∉ BCP(φ ∧ ¬L) then // check if L has AT
      let l be the first literal in L // L has RAT on l
      forall C ∈ φ_¬l do
        R ← C ◇ L
        if ∅ ∉ BCP(φ ∧ ¬R)
          return "checking failed"
    φ ← BCP(φ ∧ L)
    if ∅ ∈ φ
      return UNSAT
  return "all lemmas validated"
```

Some comparison

CNF formula

```
p cnf 4 16
 1  2  3  4 0
 1  2  3 -4 0
 1  2 -3  4 0
 1  2 -3 -4 0
 1 -2  3  4 0
 1 -2  3 -4 0
 1 -2 -3  4 0
 1 -2 -3 -4 0
-1  2  3  4 0
-1  2  3 -4 0
-1  2 -3  4 0
-1  2 -3 -4 0
-1 -2  3  4 0
-1 -2  3 -4 0
-1 -2 -3  4 0
-1 -2 -3 -4 0
```

smallest RUP proof

```
1 2 3 0
1 2 0
1 3 0
1 0
2 3 0
2 0
3 0
0
```

smallest RAT proof

```
1 0
2 0
3 0
0
```

RAT proof with ER

```
 5  1  2  0
 5  1 -2  0
 5 -1  2  0
 5 -1 -2  0
-5  3  4  0
-5  3 -4  0
-5 -3  4  0
-5 -3 -4  0
 5  1  0
 5  0
 3  0
 0
```

Benchmarks

	# variables		# clauses				time	
benchmark	input	proof	input	AT	RT	total	solving	checking
PH_{10}	90	379	415	99,682	867	100,973	5.28	24.72
PH_{11}	110	814	561	260,677	2,112	263,350	13.51	72.08
PH_{12}	132	1,450	738	1,512,453	3,954	1,517,145	145.29	3,521.23
Urq_3_5	45	2,126	446	281,761	6,243	288,450	8.33	17.38
Urq_3_6	54	3,842	688	1,156,477	11,364	1,168,529	52.69	152.36
Urq_3_7	42	1,147	342	102,950	3,315	106,607	2.20	3.95
Urq_3_8	44	1,518	416	149,286	4,422	154,124	3.70	5.86

	original			BVA preprocessed			RAT proof checking		
benchmark	# vars	# cls	time	# vars	# cls	time	#AT	#RAT	time
PH_{10}	90	330	7.71	117	226	1.25	42,853	198	4.19
PH_{11}	110	440	84.42	151	281	12.34	225,959	295	152.82
PH_{12}	132	572	494.29	187	342	8.45	181,603	402	69.01
rbcl_07	1,128	57,446	52.92	1,784	7,598	2.88	72,073	19,681	6.76
rbcl_08	1,278	67,720	1,763.36	1,980	9,004	10.72	151,894	22,830	37.58
rbcl_09	1,430	79,118	—	2,190	10,492	129.20	882,213	26,639	2,631.28

Proofs limitations

- no method for easy extraction of unsatisfiability proofs from lookahead solvers
- how to handle preprocessing of formulas (Gaussian Elimination, Cardinality Resolution, Symmetry Breaking)
- no common API for proof manipulations
- extraction of resolution proof from clausal proof
- using clausal proofs for interpolants generation