Two Flavors of DRAT

Adrián Rebola-Pardo, Armin Biere

TU Wien, JKU Linz

Pragmatics of SAT 2018
Oxford, UK
July 7th, 2018

Supported by FWF W1255-N23 and Microsoft Research through its PhD Programme

Correct proofs

When is a refutation π of F correct?

Correct proofs

When is a refutation π of F correct? “Whenever F is unsatisfiable” is not a reasonable answer!

this property is independent of π

Correct proofs

When is a refutation π of F correct? “Whenever F is unsatisfiable” is not a reasonable answer!

this property is independent of π

$$F = xyz \wedge xy\overline{z} \wedge x\overline{y}z \wedge x\overline{yz} \wedge \overline{x}yz \wedge \overline{x}y\overline{z} \wedge \overline{xy}z \wedge \overline{xyz}$$

$\pi =$ `Despite the constant negative press covfefe`

Correct proofs

When is a refutation π of F correct? "Whenever F is unsatisfiable" is not a reasonable answer!

this property is independent of π

π is a correct refutation of F whenever the inferences in π conform to some standard that *makes sense*.

i.e. a sound proof system

Correct proofs

When is a refutation π of F correct? "Whenever F is unsatisfiable" is not a reasonable answer!

this property is independent of π

π is a correct refutation of F whenever the inferences in π conform to some standard that *makes sense*.

i.e. a sound proof system

$$F = xyz \wedge xy\overline{z} \wedge x\overline{y}z \wedge x\overline{y}\overline{z} \wedge \overline{x}yz \wedge \overline{x}y\overline{z} \wedge \overline{x}\overline{y}z \wedge \overline{x}\overline{y}\overline{z}$$

$$\pi = yz,\ \overline{y}z,\ y\overline{z},\ \overline{y}\overline{z},\ z,\ \overline{z},\ \bot$$

π is a correct resolution refutation of F.

Correct proofs

When is a refutation π of F correct? “Whenever F is unsatisfiable” is not a reasonable answer!

this property is independent of π

π is a correct refutation of F whenever the inferences in π conform to some standard that *makes sense*.

i.e. a sound proof system

$$F = xyz \wedge xy\overline{z} \wedge x\overline{y}z \wedge x\overline{y}\,\overline{z} \wedge \overline{x}yz \wedge \overline{x}y\overline{z} \wedge \overline{x}\,\overline{y}z \wedge \overline{x}\,\overline{y}\,\overline{z}$$

$$\pi = yz,\ \overline{y}z,\ y\overline{z},\ \overline{y}\,\overline{z},\ z,\ \overline{z},\ \bot$$

π is a correct resolution refutation of F.

Proof correctness depends on the proof system, not on implication or consistency!

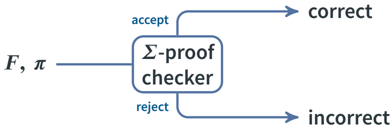
otherwise “$\bot$” is always a correct proof

Proof checking

Assume we have a proof checking procedure for a sound proof system Σ.

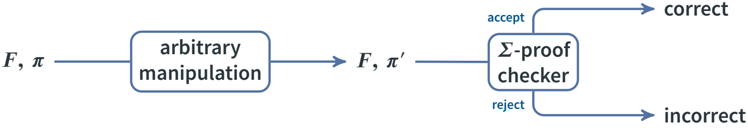
Proof checking

Assume we have a proof checking procedure for a sound proof system Σ.



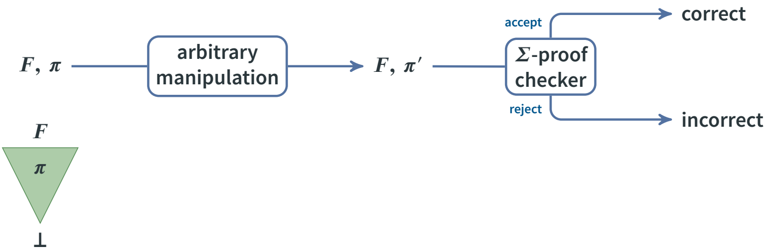
Proof checking

Assume we have a proof checking procedure for a sound proof system Σ.



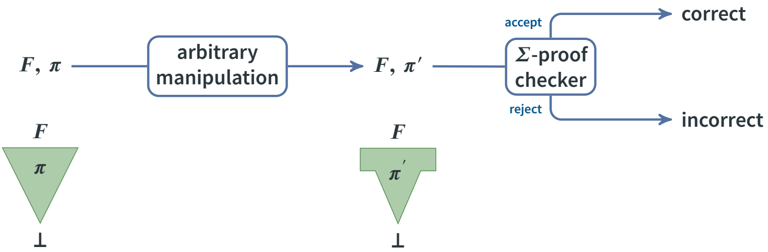
Proof checking

Assume we have a proof checking procedure for a sound proof system Σ.



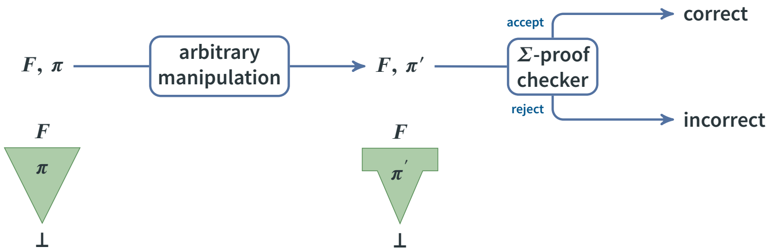
Proof checking

Assume we have a proof checking procedure for a sound proof system Σ.



Proof checking

Assume we have a proof checking procedure for a sound proof system Σ.

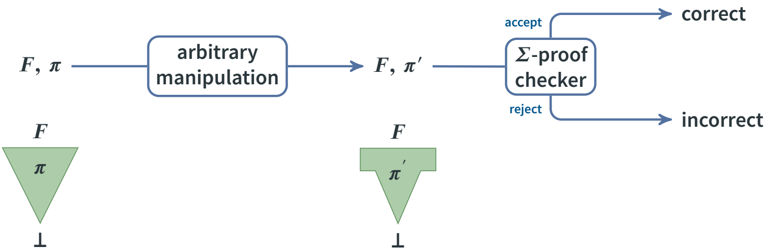


Theorem

this checking procedure is sound: whenever it succeeds, F is unsatisfiable.

Proof checking

Assume we have a proof checking procedure for a sound proof system Σ.



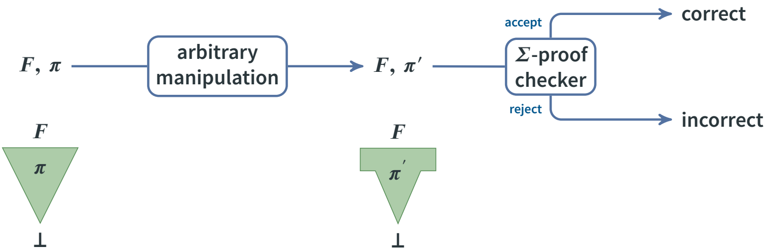
Theorem

this checking procedure is sound: whenever it succeeds, F is unsatisfiable.

But in general we cannot claim that π is correct or incorrect!

Proof checking

Assume we have a proof checking procedure for a sound proof system Σ.



Theorem

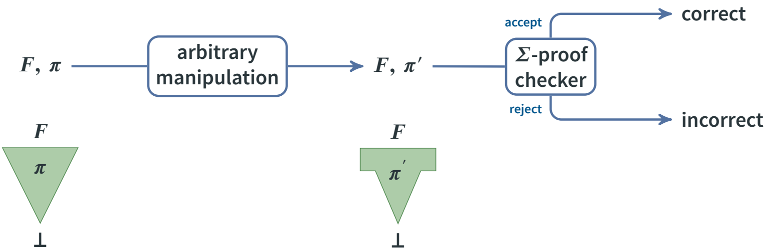
this checking procedure is sound: whenever it succeeds, F is unsatisfiable.

But in general we cannot claim that π is correct or incorrect!

This implicitly defines a sound proof system

Proof checking

Assume we have a proof checking procedure for a sound proof system Σ.



Theorem

this checking procedure is sound: whenever it succeeds, F is unsatisfiable.

But in general we cannot claim that π is correct or incorrect!

This implicitly defines a sound proof system

Problem what should we do when a proof is rejected?

In this talk

DRAT proofs [Wetzler, Heule, Hunt ’14] are the *de facto* standard for certifying correctness of SAT solvers.

In this talk

DRAT proofs [Wetzler, Heule, Hunt ’14] are the *de facto* standard for certifying correctness of SAT solvers.

DRAT checkers are checking proofs with respect to a different correctness criterion than the original DRAT definition.

In this talk

DRAT proofs [Wetzler, Heule, Hunt ’14] are the *de facto* standard for certifying correctness of SAT solvers.

DRAT checkers are checking proofs with respect to a different correctness criterion than the original DRAT definition.

- Why do DRAT checkers do this?

In this talk

DRAT proofs [Wetzler, Heule, Hunt ’14] are the *de facto* standard for certifying correctness of SAT solvers.

DRAT checkers are checking proofs with respect to a different correctness criterion than the original DRAT definition.

- Why do DRAT checkers do this?
- Can we formalize this correctness criterion as a proof system?

In this talk

DRAT proofs [Wetzler, Heule, Hunt ’14] are the *de facto* standard for certifying correctness of SAT solvers.

DRAT checkers are checking proofs with respect to a different correctness criterion than the original DRAT definition.

- Why do DRAT checkers do this?
- Can we formalize this correctness criterion as a proof system?
- Is this criterion sound (i.e. can it only refute unsatisfiable formulas)?

In this talk

DRAT proofs [Wetzler, Heule, Hunt ’14] are the *de facto* standard for certifying correctness of SAT solvers.

DRAT checkers are checking proofs with respect to a different correctness criterion than the original DRAT definition.

- Why do DRAT checkers do this?
- Can we formalize this correctness criterion as a proof system?
- Is this criterion sound (i.e. can it only refute unsatisfiable formulas)?
- How do the two criteria relate to each other?

In this talk

DRAT proofs [Wetzler, Heule, Hunt '14] are the *de facto* standard for certifying correctness of SAT solvers.

DRAT checkers are checking proofs with respect to a different correctness criterion than the original DRAT definition.

- Why do DRAT checkers do this?
- Can we formalize this correctness criterion as a proof system?
- Is this criterion sound (i.e. can it only refute unsatisfiable formulas)?
- How do the two criteria relate to each other?

Discussion which of the two criteria is more convenient?

DRAT proofs, in theory

DRAT proofs as instruction strings

DRAT proofs strings of introduction and deletion instructions

DRAT proofs as instruction strings

DRAT proofs strings of introduction and deletion instructions

i: xy, d: $xy\bar{z}$, i: x, d: y, i: $\bot$

xyz	$xy\bar{z}$	$x\bar{y}z$	$x\overline{yz}$
$\bar{x}yz$	$\bar{x}y\bar{z}$	$\overline{xy}z$	$\overline{xyz}$

DRAT proofs as instruction strings

DRAT proofs   strings of introduction and deletion instructions

i: xy,   d: $xy\bar{z}$,   i: x,   d: y,   i: $\bot$

xyz	$xy\bar{z}$	$x\bar{y}z$	$x\overline{yz}$
$\bar{x}yz$	$\bar{x}y\bar{z}$	$\overline{xy}z$	$\overline{xyz}$

DRAT proofs as instruction strings

DRAT proofs strings of introduction and **deletion** instructions

i: xy, d: $xy\bar{z}$, i: x, d: y, i: $\bot$

xyz	$xy\bar{z}$	$x\bar{y}z$	$x\bar{y}\bar{z}$
$\bar{x}yz$	$\bar{x}y\bar{z}$	$\bar{x}\bar{y}z$	$\bar{x}\bar{y}\bar{z}$

DRAT proofs as instruction strings

DRAT proofs strings of introduction and deletion instructions

A DRAT proof modifies an initial formula F into the accumulated formulas F_j

$$\text{i: } xy,\quad \text{d: } xy\bar{z},\quad \text{i: } x,\quad \text{d: } y,\quad \text{i: } \bot$$

xyz	$xy\bar{z}$	$x\bar{y}z$	$x\overline{yz}$
$\bar{x}yz$	$\bar{x}y\bar{z}$	$\overline{xy}z$	$\overline{xyz}$

DRAT proofs as instruction strings

DRAT proofs strings of introduction and deletion instructions

A DRAT proof modifies an initial formula F into the accumulated formulas F_j

i: xy, d: $xy\bar{z}$, i: x, d: y, i: $\bot$

$F_0 = F$

xyz $xy\bar{z}$ $x\bar{y}z$ $x\bar{y}\bar{z}$

$\bar{x}yz$ $\bar{x}y\bar{z}$ $\bar{x}\bar{y}z$ $\bar{x}\bar{y}\bar{z}$

DRAT proofs strings of introduction and deletion instructions

A DRAT proof modifies an initial formula F into the **accumulated formulas** F_j

i: xy, d: $xy\overline{z}$, i: x, d: y, i: $\bot$

F_1

xyz	$xy\overline{z}$	$x\overline{y}z$	$x\overline{y}\overline{z}$
$\overline{x}yz$	$\overline{x}y\overline{z}$	$\overline{x}\overline{y}z$	$\overline{x}\overline{y}\overline{z}$
xy			

DRAT proofs as instruction strings

DRAT proofs strings of introduction and deletion instructions

A DRAT proof modifies an initial formula F into the accumulated formulas F_j

i: xy, d: $xy\overline{z}$, i: x, d: y, i: $\perp$

F_2

xyz $\quad x\overline{y}z$ $\quad x\overline{yz}$

$\overline{x}yz$ $\quad \overline{x}y\overline{z}$ $\quad \overline{xy}z$ $\quad \overline{xyz}$

xy

DRAT proofs strings of introduction and deletion instructions

A DRAT proof modifies an initial formula F into the accumulated formulas F_j

i: xy, d: $xy\overline{z}$, i: x, d: y, i: $\bot$

F_3

xyz		$x\overline{y}z$	$x\overline{yz}$
$\overline{x}yz$	$\overline{x}y\overline{z}$	$\overline{xy}z$	$\overline{xyz}$
xy	x		

DRAT proofs as instruction strings

DRAT proofs strings of introduction and deletion instructions

A DRAT proof modifies an initial formula F into the accumulated formulas F_j

i: xy, d: $xy\bar{z}$, i: x, d: y, i: $\bot$

F_4

xyz		$x\bar{y}z$	$x\overline{yz}$
$\bar{x}yz$	$\bar{x}y\bar{z}$	$\overline{xy}z$	$\overline{xyz}$
xy	x		

DRAT proofs as instruction strings

DRAT proofs strings of introduction and deletion instructions

A DRAT proof modifies an initial formula F into the accumulated formulas F_j

i: xy, d: $xy\bar{z}$, i: x, d: y, i: $\bot$

F_5

xyz		$x\bar{y}z$	$x\bar{y}\bar{z}$
$\bar{x}yz$	$\bar{x}y\bar{z}$	$\bar{x}\bar{y}z$	$\bar{x}\bar{y}\bar{z}$
xy	x	$\bot$	

Correct DRAT proofs

Deletion instructions are always correct. [Heule, Hunt, Wetzler ’14]

Correct DRAT proofs

Deletion instructions are always correct. [Heule, Hunt, Wetzler '14]

In this case, $F \wedge C \models F$

An introduction instruction i: C is correct over the accumulated formula F if:

Correct DRAT proofs

Deletion instructions are always correct. [Heule, Hunt, Wetzler '14]

In this case, $F \wedge C \vDash F$

An introduction instruction i: C is correct over the accumulated formula F if:

- C is a reverse unit propagation (RUP) in F [Novikov, Goldberg '03]: unit propagation over $F \wedge \neg C$ leads to conflict.

  In this case, $F \vDash C$

Correct DRAT proofs

Deletion instructions are always correct. [Heule, Hunt, Wetzler '14]

In this case, $F \wedge C \models F$

An introduction instruction i: C is correct over the accumulated formula F if:

- C is a reverse unit propagation (RUP) in F [Novikov, Goldberg '03]: unit propagation over $F \wedge \neg C$ leads to conflict.

  In this case, $F \models C$

- C is a resolution asymmetric tautology (RAT) in F [Heule, Hunt, Wetzler '13]: there is a *pivot* literal $x \in C$ such that $C \vee D \setminus \{x\}$ is a RUP in F for every clause $D \in F$.

  In this case, F *is satisfiability-equivalent to* $F \wedge C$

Correct DRAT proofs

Deletion instructions are always correct. [Heule, Hunt, Wetzler '14]

In this case, $F \wedge C \vDash F$

An introduction instruction i: C is correct over the accumulated formula F if:

- C is a reverse unit propagation (RUP) in F [Novikov, Goldberg '03]: unit propagation over $F \wedge \neg C$ leads to conflict.

  In this case, $F \vDash C$

- C is a resolution asymmetric tautology (RAT) in F [Heule, Hunt, Wetzler '13]: there is a *pivot* literal $x \in C$ such that $C \vee D \setminus \{x\}$ is a RUP in F for every clause $D \in F$.

  In this case, F *is satisfiability-equivalent to* $F \wedge C$

Unit propagation is needed to check inferences

DRAT checkers use the good old two-watched literal schema

Correct DRAT proofs

Deletion instructions are always correct. [Heule, Hunt, Wetzler '14]

In this case, $F \wedge C \vDash F$

An introduction instruction i: C is correct over the accumulated formula F if:

- C is a reverse unit propagation (RUP) in F [Novikov, Goldberg '03]: unit propagation over $F \wedge \neg C$ leads to conflict.

  In this case, $F \vDash C$

- C is a resolution asymmetric tautology (RAT) in F [Heule, Hunt, Wetzler '13]: there is a *pivot* literal $x \in C$ such that $C \vee D \setminus \{x\}$ is a RUP in F for every clause $D \in F$.

  In this case, F *is satisfiability-equivalent to* $F \wedge C$

Unit propagation is needed to check inferences

DRAT checkers use the good old two-watched literal schema

Observation

RAT introduction is non-monotonic: C is a RAT in $F \nRightarrow C$ is a RAT in $F \wedge G$

deletion may disable but also enable RAT inferences! *[Rebola-Pardo, Philipp '17]*

DRAT proofs, in practice

DRAT proofs, in practice

The class of accepted proofs by DRAT checkers differs from the class of proofs accepted by the DRAT definition.

DRAT proofs, in practice

The class of accepted proofs by DRAT checkers differs from the class of proofs accepted by the DRAT definition.

Multiset semantics required for efficient proof generation

Probably should have been included in the definition

DRAT proofs, in practice

The class of accepted proofs by DRAT checkers differs from the class of proofs accepted by the DRAT definition.

Multiset semantics **required for efficient proof generation**

Probably should have been included in the definition

Unit clause deletion **simpler (but not necessarily faster) proof checking**

Is this really needed?

Proof generation and multiset semantics

Clauses learned by CDCL SAT solvers are always RUP clauses

Proof generation and multiset semantics

Clauses learned by CDCL SAT solvers are always RUP clauses

recording the sequence of learned clauses yields a DRAT proof

Proof generation and multiset semantics

Clauses learned by CDCL SAT solvers are always RUP clauses

recording the sequence of learned clauses yields a DRAT proof

Clauses derived by inprocessing techniques must be derived by *ad hoc* methods.

F

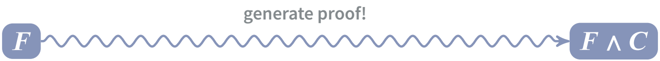
$F \wedge C$

Proof generation and multiset semantics

Clauses learned by CDCL SAT solvers are always RUP clauses

recording the sequence of learned clauses yields a DRAT proof

Clauses derived by inprocessing techniques must be derived by *ad hoc* methods.



Proof generation and multiset semantics

Clauses learned by CDCL SAT solvers are always RUP clauses

recording the sequence of learned clauses yields a DRAT proof

Clauses derived by inprocessing techniques must be derived by *ad hoc* methods.

F i: C $F \wedge C$

Proof generation and multiset semantics

Clauses learned by CDCL SAT solvers are always RUP clauses

recording the sequence of learned clauses yields a DRAT proof

Clauses derived by inprocessing techniques must be derived by *ad hoc* methods.

what if C is not
a RUP/RAT in F?

F i: C $F \wedge C$

Proof generation and multiset semantics

Clauses learned by CDCL SAT solvers are always RUP clauses

recording the sequence of learned clauses yields a DRAT proof

Clauses derived by inprocessing techniques must be derived by *ad hoc* methods.

F i: A_1, …, i: A_n, i: C, d: A_n, …, d: A_1 $F \wedge C$

Proof generation and multiset semantics

Clauses learned by CDCL SAT solvers are always RUP clauses

recording the sequence of learned clauses yields a DRAT proof

Clauses derived by inprocessing techniques must be derived by *ad hoc* methods.

F i: A_1, … , i: A_n, i: C, d: A_n, … , d: A_1 $F \wedge C$

Proof generation and multiset semantics

Clauses learned by CDCL SAT solvers are always RUP clauses

recording the sequence of learned clauses yields a DRAT proof

Clauses derived by inprocessing techniques must be derived by *ad hoc* methods.

F i: A_1, … , i: A_n, i: C, d: A_n, … , d: A_1 $F \wedge C$

Proof generation and multiset semantics

Clauses learned by CDCL SAT solvers are always RUP clauses

recording the sequence of learned clauses yields a DRAT proof

Clauses derived by inprocessing techniques must be derived by *ad hoc* methods.

F i: A_1, …, i: A_n, i: C, d: A_n, …, d: A_1 $F \wedge C$

Proof generation and multiset semantics

Clauses learned by CDCL SAT solvers are always RUP clauses

recording the sequence of learned clauses yields a DRAT proof

Clauses derived by inprocessing techniques must be derived by *ad hoc* methods.

careful there!

F i: A_1, …, i: A_n, i: C, d: A_n, …, d: A_1 $F \wedge C$

Proof generation and multiset semantics

Clauses learned by CDCL SAT solvers are always RUP clauses

recording the sequence of learned clauses yields a DRAT proof

Clauses derived by inprocessing techniques must be derived by *ad hoc* methods.

$F \wedge A_1$ i: A_1, … , i: A_n, i: C, d: A_n, … , d: A_1 $F \wedge A_1 \wedge C$

Proof generation and multiset semantics

Clauses learned by CDCL SAT solvers are always RUP clauses

recording the sequence of learned clauses yields a DRAT proof

Clauses derived by inprocessing techniques must be derived by *ad hoc* methods.

$F \wedge A_1$ i: A_1, …, i: A_n, i: C, d: A_n, …, d: A_1 $F \wedge A_1 \wedge C$

derived formula:
$F \wedge C$

Proof generation and multiset semantics

Clauses learned by CDCL SAT solvers are always RUP clauses

recording the sequence of learned clauses yields a DRAT proof

Clauses derived by inprocessing techniques must be derived by *ad hoc* methods.

$F \wedge A_1$ i: A_1, … , i: A_n, i: C, d: A_n, … , d: A_1 $F \wedge A_1 \wedge C$

Solution Consider CNF formulas as multisets of clauses

Proof generation and multiset semantics

Clauses learned by CDCL SAT solvers are always RUP clauses

recording the sequence of learned clauses yields a DRAT proof

Clauses derived by inprocessing techniques must be derived by *ad hoc* methods.

$F \wedge A_1$ i: A_1, … , i: A_n, i: C, d: A_n, … , d: A_1 $F \wedge A_1 \wedge C$

Solution Consider CNF formulas as multisets of clauses

DRAT checkers assume this, but it was not specified in the definition

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

$$
\begin{array}{cccc}
x_1 & \overline{x_1}x_2 & \overline{x_1}\,\overline{x_2}x_3 & \overline{x_1}\,\overline{x_3}x_4 \\
x_5x_6 & \overline{x_2}\,\overline{x_5}x_7 & \overline{x_1}\,\overline{x_5}x_6 & x_4\overline{x_5}\,\overline{x_6} \\
\overline{x_3}\,\overline{x_6}x_8 & x_3\overline{x_4}\,\overline{x_6} & x_5\overline{x_8} & \overline{x_3}x_9x_{10} \\
\overline{x_4}\,\overline{x_9}x_{10} & x_9\overline{x_{10}} & x_7\overline{x_9} & \overline{x_7}\,\overline{x_8}\,\overline{x_9}\,\overline{x_{10}}
\end{array}
$$

i: x_5, i: x_4, i: x_9, i: $\perp$

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

$$\begin{array}{cccc}
x_1 & \overline{x_1}\,x_2 & \overline{x_1}\,\overline{x_2}\,x_3 & \overline{x_1}\,\overline{x_3}\,x_4 \\
x_5\,x_6 & \overline{x_2}\,\overline{x_5}\,x_7 & \overline{x_1}\,\overline{x_5}\,x_6 & x_4\,\overline{x_5}\,\overline{x_6} \\
\overline{x_3}\,\overline{x_6}\,x_8 & x_3\,\overline{x_4}\,\overline{x_6} & x_5\,\overline{x_8} & \overline{x_3}\,x_9\,x_{10} \\
\overline{x_4}\,\overline{x_9}\,x_{10} & x_9\,\overline{x_{10}} & x_7\,\overline{x_9} & \overline{x_7}\,\overline{x_8}\,\overline{x_9}\,\overline{x_{10}}
\end{array}$$

i: x_5, i: x_4, i: x_9, i: $\perp$

$x_1,\ x_2,\ x_3,\ x_4$

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

x_1	$\overline{x}_1 x_2$	$\overline{x}_1 \overline{x}_2 x_3$	$\overline{x}_1 \overline{x}_3 x_4$
$x_5 x_6$	$\overline{x}_2 \overline{x}_5 x_7$	$\overline{x}_1 \overline{x}_5 x_6$	$x_4 \overline{x}_5 \overline{x}_6$
$\overline{x}_3 \overline{x}_6 x_8$	$x_3 \overline{x}_4 \overline{x}_6$	$x_5 \overline{x}_8$	$\overline{x}_3 x_9 x_{10}$
$\overline{x}_4 \overline{x}_9 x_{10}$	$x_9 \overline{x}_{10}$	$x_7 \overline{x}_9$	$\overline{x}_7 \overline{x}_8 \overline{x}_9 \overline{x}_{10}$

i: x_5, i: x_4, i: x_9, i: $\perp$

$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

x_1	$\overline{x}_1 x_2$	$\overline{x}_1 \overline{x}_2 x_3$	$\overline{x}_1 \overline{x}_3 x_4$
$x_5 x_6$	$\overline{x}_2 \overline{x}_5 x_7$	$\overline{x}_1 \overline{x}_5 x_6$	$x_4 \overline{x}_5 \overline{x}_6$
$\overline{x}_3 \overline{x}_6 x_8$	$x_3 \overline{x}_4 \overline{x}_6$	$x_5 \overline{x}_8$	$\overline{x}_3 x_9 x_{10}$
$\overline{x}_4 \overline{x}_9 x_{10}$	$x_9 \overline{x}_{10}$	$x_7 \overline{x}_9$	$\overline{x}_7 \overline{x}_8 \overline{x}_9 \overline{x}_{10}$

i: x_5, i: x_4, i: x_9, i: $\perp$

$x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7,\ x_8$

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

x_1	$\overline{x_1} x_2$	$\overline{x_1 x_2} x_3$	$\overline{x_1 x_3} x_4$
$x_5 x_6$	$\overline{x_2 x_5} x_7$	$\overline{x_1 x_5} x_6$	$x_4 \overline{x_5 x_6}$
$\overline{x_3 x_6} x_8$	$x_3 \overline{x_4 x_6}$	$x_5 \overline{x_8}$	$\overline{x_3} x_9 x_{10}$
$\overline{x_4 x_9} x_{10}$	$x_9 \overline{x_{10}}$	$x_7 \overline{x_9}$	$\overline{x_7 x_8 x_9 x_{10}}$

i: x_5, i: x_4, i: x_9, i: $\bot$

$x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7,\ x_8,\ x_9,\ x_{10},\ \overline{x_{10}}$

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

x_1	$\overline{x_1}x_2$	$\overline{x_1}\,\overline{x_2}x_3$	$\overline{x_1}\,\overline{x_3}x_4$
x_5x_6	$\overline{x_2}\,\overline{x_5}x_7$	$\overline{x_1}\,\overline{x_5}x_6$	$x_4\overline{x_5}\,\overline{x_6}$
$\overline{x_3}\,\overline{x_6}x_8$	$x_3\overline{x_4}\,\overline{x_6}$	$x_5\overline{x_8}$	$\overline{x_3}x_9x_{10}$
$\overline{x_4}\,\overline{x_9}x_{10}$	$x_9\overline{x_{10}}$	$x_7\overline{x_9}$	$\overline{x_7}\,\overline{x_8}\,\overline{x_9}\,\overline{x_{10}}$

i: x_5, d: $\overline{x_1}x_2$, i: x_4, i: x_9, i: $\perp$

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

x_1	$\overline{x_1}x_2$	$\overline{x_1 x_2}x_3$	$\overline{x_1 x_3}x_4$
$x_5 x_6$	$\overline{x_2 x_5}x_7$	$\overline{x_1 x_5}x_6$	$x_4\overline{x_5 x_6}$
$\overline{x_3 x_6}x_8$	$x_3\overline{x_4 x_6}$	$x_5\overline{x_8}$	$\overline{x_3}x_9 x_{10}$
$\overline{x_4 x_9}x_{10}$	$x_9\overline{x_{10}}$	$x_7\overline{x_9}$	$\overline{x_7 x_8 x_9 x_{10}}$

i: x_5, d: $\overline{x_1}x_2$, i: x_4, i: x_9, i: $\perp$

$x_1,\ x_2,\ x_3,\ x_4$

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

$$
\begin{array}{cccc}
x_1 & \overline{x}_1 x_2 & \overline{x}_1\overline{x}_2 x_3 & \overline{x}_1\overline{x}_3 x_4 \\
x_5 x_6 & \overline{x}_2\overline{x}_5 x_7 & \overline{x}_1\overline{x}_5 x_6 & x_4\overline{x}_5\overline{x}_6 \\
\overline{x}_3\overline{x}_6 x_8 & x_3\overline{x}_4\overline{x}_6 & x_5\overline{x}_8 & \overline{x}_3 x_9 x_{10} \\
\overline{x}_4\overline{x}_9 x_{10} & x_9\overline{x}_{10} & x_7\overline{x}_9 & \overline{x}_7\overline{x}_8\overline{x}_9\overline{x}_{10}
\end{array}
$$

i: x_5, d: $\overline{x}_1 x_2$, i: x_4, i: x_9, i: $\perp$

$x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7,\ x_8$

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

x_1	$\overline{x_1} x_2$	$\overline{x_1}\,\overline{x_2} x_3$	$\overline{x_1}\,\overline{x_3} x_4$
$x_5 x_6$	$\overline{x_2}\,\overline{x_5} x_7$	$\overline{x_1}\,\overline{x_5} x_6$	$x_4 \overline{x_5}\,\overline{x_6}$
$\overline{x_3}\,\overline{x_6} x_8$	$x_3 \overline{x_4}\,\overline{x_6}$	$x_5 \overline{x_8}$	$\overline{x_3} x_9 x_{10}$
$\overline{x_4}\,\overline{x_9} x_{10}$	$x_9 \overline{x_{10}}$	$x_7 \overline{x_9}$	$\overline{x_7}\,\overline{x_8}\,\overline{x_9}\,\overline{x_{10}}$

i: x_5, d: $\overline{x_1} x_2$, i: x_4, i: x_9, i: $\perp$

x_1, x_3, x_4, x_5, x_6, x_8

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

x_1	$\overline{x_1} x_2$	$\overline{x_1}\,\overline{x_2} x_3$	$\overline{x_1}\,\overline{x_3} x_4$
$x_5 x_6$	$\overline{x_2}\,\overline{x_5} x_7$	$\overline{x_1}\,\overline{x_5} x_6$	$x_4 \overline{x_5}\,\overline{x_6}$
$\overline{x_3}\,\overline{x_6} x_8$	$x_3 \overline{x_4}\,\overline{x_6}$	$x_5 \overline{x_8}$	$\overline{x_3} x_9 x_{10}$
$\overline{x_4}\,\overline{x_9} x_{10}$	$x_9 \overline{x_{10}}$	$x_7 \overline{x_9}$	$\overline{x_7}\,\overline{x_8}\,\overline{x_9}\,\overline{x_{10}}$

i: x_5, d: $\overline{x_1} x_2$, i: x_4, i: x_9, i: $\bot$

x_1, x_3, x_4, x_5, x_6, x_8

literals implied by unit propagation

Solution DRAT checkers ignore unit clause deletions [Heule '16]

Two flavors of DRAT

DRAT checkers are checking proofs with respect to a different proof system.

- Multiset semantics is justified by constraints in proof generation
  checking if a clause occurs in the formula is expensive
- Ignoring unit clause deletions is justified by constraints in proof checking
  No efficient unit propagation without two-watched literal schema

Multiset semantics should be included in the DRAT specification.
Should ignoring unit clause deletions be?

Specified DRAT
  Original definition + multiset semantics

Operational DRAT
  Original definition + multiset semantics + ignoring unit clause deletions

Operational DRAT as a proof system

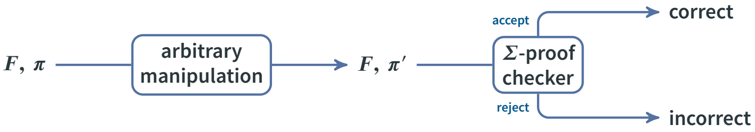
Is operational DRAT sound?

Remember: deletions may change whether RATs can be infered

Operational DRAT as a proof system

Is operational DRAT sound?

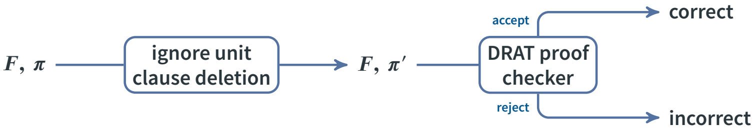
Remember: deletions may change whether RATs can be infered



Operational DRAT as a proof system

Is operational DRAT sound?

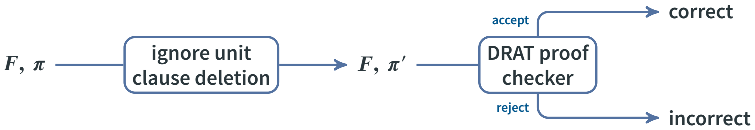
Remember: deletions may change whether RATs can be infered



Operational DRAT as a proof system

Is operational DRAT sound? **Yes!**

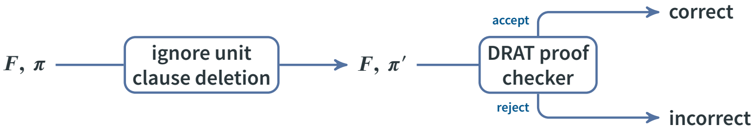
Remember: deletions may change whether RATs can be infered



Operational DRAT as a proof system

Is operational DRAT sound? Yes!

Remember: deletions may change whether RATs can be infered

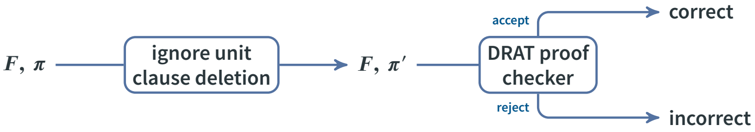


Is operational DRAT stronger or weaker than specified DRAT?

Operational DRAT as a proof system

Is operational DRAT sound? Yes!

Remember: deletions may change whether RATs can be infered

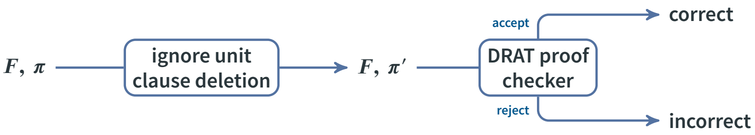


Is operational DRAT stronger or weaker than specified DRAT? Neither!

Operational DRAT as a proof system

Is operational DRAT sound? Yes!

Remember: deletions may change whether RATs can be infered



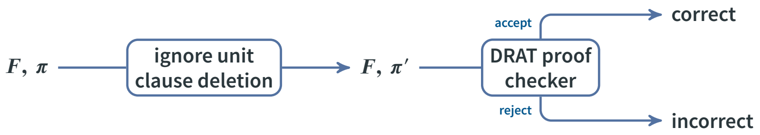
Is operational DRAT stronger or weaker than specified DRAT? Neither!

Can operational DRAT be formalized with inference rules?

Operational DRAT as a proof system

Is operational DRAT sound? Yes!

Remember: deletions may change whether RATs can be infered



Is operational DRAT stronger or weaker than specified DRAT? Neither!

Can operational DRAT be formalized with inference rules? Yes! (paper)

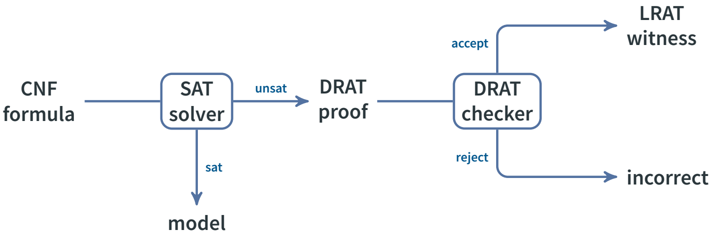
Discussion

What flavor should be used?

Discussion should operational DRAT or specified DRAT be used?

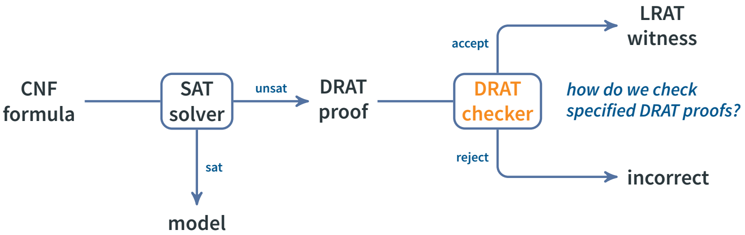
What flavor should be used?

Discussion should operational DRAT or specified DRAT be used?



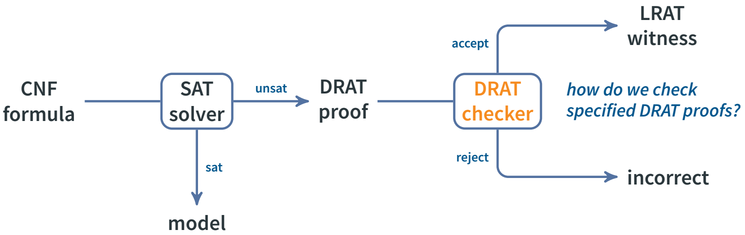
What flavor should be used?

Discussion should operational DRAT or specified DRAT be used?



What flavor should be used?

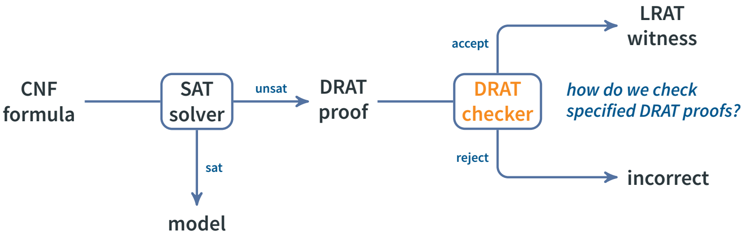
Discussion should operational DRAT or specified DRAT be used?



No publicly available specified DRAT checker

What flavor should be used?

Discussion should operational DRAT or specified DRAT be used?

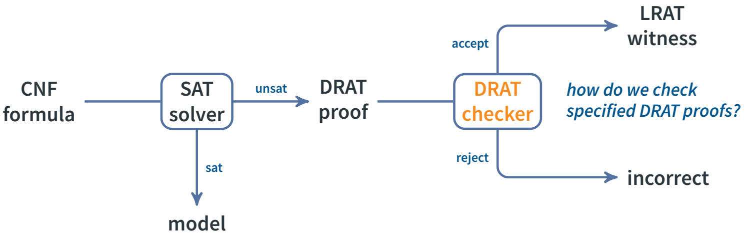


No publicly available specified DRAT checker

Two-watched literal invariants are hard to maintain

What flavor should be used?

Discussion should operational DRAT or specified DRAT be used?



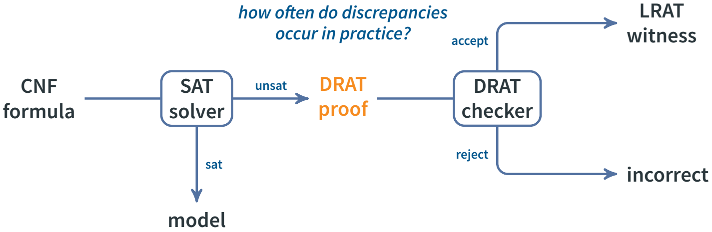
No publicly available specified DRAT checker

Two-watched literal invariants are hard to maintain

Under review first specified DRAT checker [RP, Cruz-Filipe]

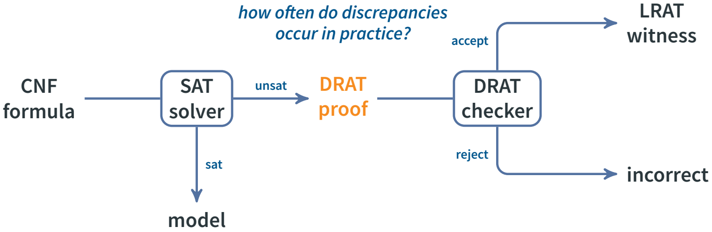
What flavor should be used?

Discussion should operational DRAT or specified DRAT be used?



What flavor should be used?

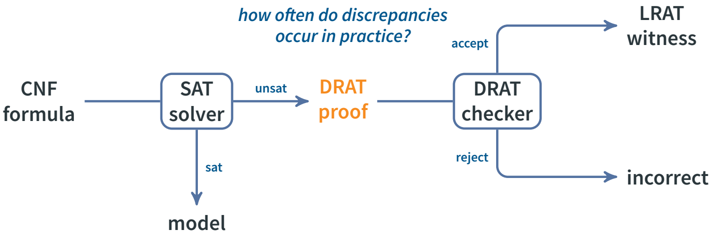
Discussion should operational DRAT or specified DRAT be used?



Potentially often 95% DRAT proofs contain unit deletions

What flavor should be used?

Discussion should operational DRAT or specified DRAT be used?

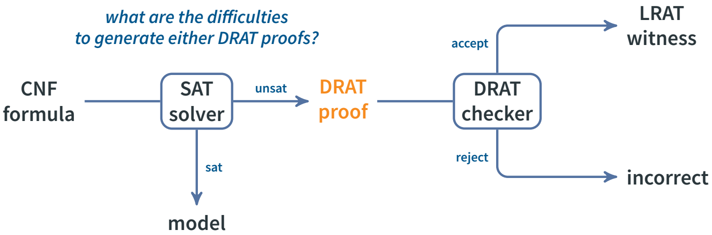


Potentially often 95% DRAT proofs contain unit deletions

Under review 59% discrepancies [RP, Cruz-Filipe]

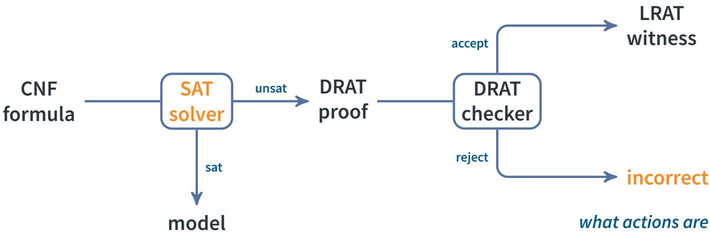
What flavor should be used?

Discussion should operational DRAT or specified DRAT be used?



What flavor should be used?

Discussion should operational DRAT or specified DRAT be used?

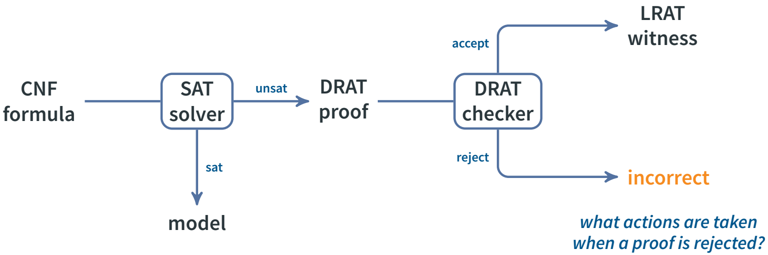


what actions are taken when a proof is rejected?

Solver debugging could lead to huge waste of time

What flavor should be used?

Discussion should operational DRAT or specified DRAT be used?

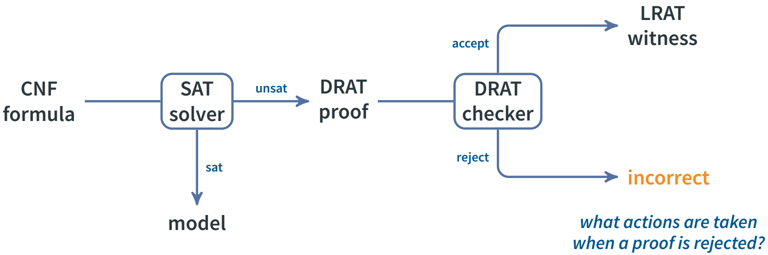


Solver debugging could lead to huge waste of time

Disqualifying solvers proofs rejected by DRAT-trim may be correct!

What flavor should be used?

Discussion should operational DRAT or specified DRAT be used?



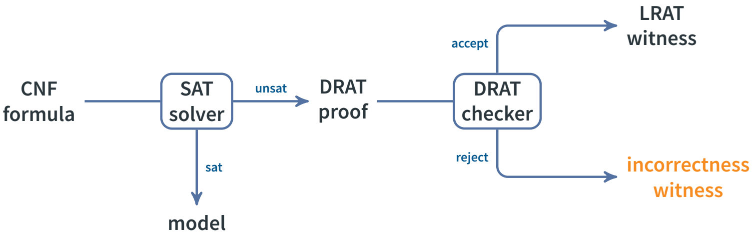
Solver debugging could lead to huge waste of time

Disqualifying solvers proofs rejected by DRAT-trim may be correct!

Future work verifying incorrectness results

What flavor should be used?

Discussion should operational DRAT or specified DRAT be used?



Solver debugging could lead to huge waste of time

Disqualifying solvers proofs rejected by DRAT-trim may be correct!

Future work verifying incorrectness results

Backup slides

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

$$
\begin{array}{cccc}
x_1 & \overline{x}_1 x_2 & \overline{x}_1 \overline{x}_2 x_3 & \overline{x}_1 \overline{x}_3 x_4 \\
x_5 x_6 & \overline{x}_2 \overline{x}_5 x_7 & \overline{x}_1 \overline{x}_5 x_6 & x_4 \overline{x}_5 \overline{x}_6 \\
\overline{x}_3 \overline{x}_6 x_8 & x_3 \overline{x}_4 \overline{x}_6 & x_5 \overline{x}_8 & \overline{x}_3 x_9 x_{10} \\
\overline{x}_4 \overline{x}_9 x_{10} & x_9 \overline{x}_{10} & x_7 \overline{x}_9 & \overline{x}_7 \overline{x}_8 \overline{x}_9 \overline{x}_{10}
\end{array}
$$

i: x_5, i: x_4, i: x_9, i: $\perp$

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

x_1	$\overline{x}_1 x_2$	$\overline{x}_1 \overline{x}_2 x_3$	$\overline{x}_1 \overline{x}_3 x_4$
$x_5 x_6$	$\overline{x}_2 \overline{x}_5 x_7$	$\overline{x}_1 \overline{x}_5 x_6$	$x_4 \overline{x}_5 \overline{x}_6$
$\overline{x}_3 \overline{x}_6 x_8$	$x_3 \overline{x}_4 \overline{x}_6$	$x_5 \overline{x}_8$	$\overline{x}_3 x_9 x_{10}$
$\overline{x}_4 \overline{x}_9 x_{10}$	$x_9 \overline{x}_{10}$	$x_7 \overline{x}_9$	$\overline{x}_7 \overline{x}_8 \overline{x}_9 \overline{x}_{10}$

i: x_5, i: x_4, i: x_9, i: $\perp$

$x_1,\ x_2,\ x_3,\ x_4$

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

$$\begin{array}{cccc}
x_1 & \overline{x_1}x_2 & \overline{x_1}\,\overline{x_2}x_3 & \overline{x_1}\,\overline{x_3}x_4 \\
x_5x_6 & \overline{x_2}\,\overline{x_5}x_7 & \overline{x_1}\,\overline{x_5}x_6 & x_4\overline{x_5}\,\overline{x_6} \\
\overline{x_3}\,\overline{x_6}x_8 & x_3\overline{x_4}\,\overline{x_6} & x_5\overline{x_8} & \overline{x_3}x_9x_{10} \\
\overline{x_4}\,\overline{x_9}x_{10} & x_9\overline{x_{10}} & x_7\overline{x_9} & \overline{x_7}\,\overline{x_8}\,\overline{x_9}\,\overline{x_{10}}
\end{array}$$

i: x_5, i: x_4, i: x_9, i: $\perp$

$x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7,\ x_8$

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

$$
\begin{array}{cccc}
x_1 & \overline{x_1} x_2 & \overline{x_1}\,\overline{x_2} x_3 & \overline{x_1}\,\overline{x_3} x_4 \\
x_5 x_6 & \overline{x_2}\,\overline{x_5} x_7 & \overline{x_1}\,\overline{x_5} x_6 & x_4 \overline{x_5}\,\overline{x_6} \\
\overline{x_3}\,\overline{x_6} x_8 & x_3 \overline{x_4}\,\overline{x_6} & x_5 \overline{x_8} & \overline{x_3} x_9 x_{10} \\
\overline{x_4}\,\overline{x_9} x_{10} & x_9 \overline{x_{10}} & x_7 \overline{x_9} & \overline{x_7}\,\overline{x_8}\,\overline{x_9}\,\overline{x_{10}}
\end{array}
$$

i: x_5, i: x_4, i: x_9, i: $\perp$

$x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7,\ x_8$

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

x_1	$\overline{x_1} x_2$	$\overline{x_1}\,\overline{x_2} x_3$	$\overline{x_1}\,\overline{x_3} x_4$
$x_5 x_6$	$\overline{x_2}\,\overline{x_5} x_7$	$\overline{x_1}\,\overline{x_5} x_6$	$x_4 \overline{x_5}\,\overline{x_6}$
$\overline{x_3}\,\overline{x_6} x_8$	$x_3 \overline{x_4}\,\overline{x_6}$	$x_5 \overline{x_8}$	$\overline{x_3} x_9 x_{10}$
$\overline{x_4}\,\overline{x_9} x_{10}$	$x_9 \overline{x_{10}}$	$x_7 \overline{x_9}$	$\overline{x_7}\,\overline{x_8}\,\overline{x_9}\,\overline{x_{10}}$

i: x_5, i: x_4, i: x_9, i: $\bot$

$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, \overline{x_{10}}$

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

x_1	$\overline{x_1} x_2$	$\overline{x_1}\,\overline{x_2} x_3$	$\overline{x_1}\,\overline{x_3} x_4$
$x_5 x_6$	$\overline{x_2}\,\overline{x_5} x_7$	$\overline{x_1}\,\overline{x_5} x_6$	$x_4 \overline{x_5}\,\overline{x_6}$
$\overline{x_3}\,\overline{x_6} x_8$	$x_3 \overline{x_4}\,\overline{x_6}$	$x_5 \overline{x_8}$	$\overline{x_3} x_9 x_{10}$
$\overline{x_4}\,\overline{x_9} x_{10}$	$x_9 \overline{x_{10}}$	$x_7 \overline{x_9}$	$\overline{x_7}\,\overline{x_8}\,\overline{x_9}\,\overline{x_{10}}$

i: x_5, d: $\overline{x_1} x_2$, i: x_4, i: x_9, i: $\bot$

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

$$
\begin{array}{cccc}
x_1 & \overline{x_1} x_2 & \overline{x_1 x_2} x_3 & \overline{x_1 x_3} x_4 \\
x_5 x_6 & \overline{x_2 x_5} x_7 & \overline{x_1 x_5} x_6 & x_4 \overline{x_5 x_6} \\
\overline{x_3 x_6} x_8 & x_3 \overline{x_4 x_6} & x_5 \overline{x_8} & \overline{x_3} x_9 x_{10} \\
\overline{x_4 x_9} x_{10} & x_9 \overline{x_{10}} & x_7 \overline{x_9} & \overline{x_7 x_8 x_9 x_{10}}
\end{array}
$$

i: x_5, d: $\overline{x_1} x_2$, i: x_4, i: x_9, i: $\perp$

x_1, x_2, x_3, x_4

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

$$
\begin{array}{cccc}
x_1 & \overline{x_1} x_2 & \overline{x_1}\,\overline{x_2} x_3 & \overline{x_1}\,\overline{x_3} x_4 \\
x_5 x_6 & \overline{x_2}\,\overline{x_5} x_7 & \overline{x_1}\,\overline{x_5} x_6 & x_4 \overline{x_5}\,\overline{x_6} \\
\overline{x_3}\,\overline{x_6} x_8 & x_3 \overline{x_4}\,\overline{x_6} & x_5 \overline{x_8} & \overline{x_3} x_9 x_{10} \\
\overline{x_4}\,\overline{x_9} x_{10} & x_9 \overline{x_{10}} & x_7 \overline{x_9} & \overline{x_7}\,\overline{x_8}\,\overline{x_9}\,\overline{x_{10}}
\end{array}
$$

i: x_5, d: $\overline{x_1} x_2$, i: x_4, i: x_9, i: $\perp$

$x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7,\ x_8$

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

x_1	$\overline{x_1} x_2$	$\overline{x_1 x_2} x_3$	$\overline{x_1 x_3} x_4$
$x_5 x_6$	$\overline{x_2 x_5} x_7$	$\overline{x_1 x_5} x_6$	$x_4 \overline{x_5 x_6}$
$\overline{x_3 x_6} x_8$	$x_3 \overline{x_4 x_6}$	$x_5 \overline{x_8}$	$\overline{x_3} x_9 x_{10}$
$\overline{x_4 x_9} x_{10}$	$x_9 \overline{x_{10}}$	$x_7 \overline{x_9}$	$\overline{x_7 x_8 x_9 x_{10}}$

i: x_5, d: $\overline{x_1} x_2$, i: x_4, i: x_9, i: $\perp$

x_1, x_3, x_4, x_5, x_6, x_8

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

Two-watched literal schema

if one watched literal is assigned to false, then the other watched literal must be assigned to true

x_1	$\overline{x_1} x_2$	$\overline{x_1}\,\overline{x_2} x_3$	$\overline{x_1}\,\overline{x_3} x_4$
$x_5 x_6$	$\overline{x_2}\,\overline{x_5} x_7$	$\overline{x_1}\,\overline{x_5} x_6$	$x_4 \overline{x_5}\,\overline{x_6}$
$\overline{x_3}\,\overline{x_6} x_8$	$x_3 \overline{x_4}\,\overline{x_6}$	$x_5 \overline{x_8}$	$\overline{x_3} x_9 x_{10}$
$\overline{x_4}\,\overline{x_9} x_{10}$	$x_9 \overline{x_{10}}$	$x_7 \overline{x_9}$	$\overline{x_7}\,\overline{x_8}\,\overline{x_9}\,\overline{x_{10}}$

i: x_5, d: $\overline{x_1} x_2$, i: x_4, i: x_9, i: $\bot$

x_1, x_3, x_4, x_5, x_6, x_8

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

Two-watched literal schema

if one watched literal is assigned to false, then the other watched literal must be assigned to true

$$
\begin{array}{cccc}
x_1 & \overline{x}_1 x_2 & \overline{x}_1 \overline{x}_2 x_3 & \overline{x}_1 \overline{x}_3 x_4 \\
x_5 x_6 & \overline{x}_2 \overline{x}_5 x_7 & \overline{x}_1 \overline{x}_5 x_6 & x_4 \overline{x}_5 \overline{x}_6 \\
\overline{x}_3 \overline{x}_6 x_8 & x_3 \overline{x}_4 \overline{x}_6 & x_5 \overline{x}_8 & \overline{x}_3 x_9 x_{10} \\
\overline{x}_4 \overline{x}_9 x_{10} & x_9 \overline{x}_{10} & x_7 \overline{x}_9 & \overline{x}_7 \overline{x}_8 \overline{x}_9 \overline{x}_{10}
\end{array}
$$

i: x_5, d: $\overline{x}_1 x_2$, i: x_4, i: x_9, i: $\perp$

x_1, x_3, x_4, x_5, x_6, x_8

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

Two-watched literal schema

if one watched literal is assigned to false, then the other watched literal must be assigned to true

$\overline{x_2}\ \ \overline{x_5}\ \ x_7$

i: x_5, d: $\overline{x_1} x_2$, i: x_4, i: x_9, i: $\perp$

$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

Two-watched literal schema

if one watched literal is assigned to false, then the other watched literal must be assigned to true

$\overline{x_2}\ \ \overline{x_5}\ \ x_7$

i: x_5, d: $\overline{x_1} x_2$, i: x_4, i: x_9, i: $\bot$

$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

Two-watched literal schema

if one watched literal is assigned to false, then the other watched literal must be assigned to true

$$\overline{x_2} \quad \overline{x_5} \quad x_7$$

i: x_5, d: $\overline{x_1} x_2$, i: x_4, i: x_9, i: $\perp$

x_1, x_3, x_4, x_5, x_6, x_8

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

Two-watched literal schema

if one watched literal is assigned to false, then the other watched literal must be assigned to true

$\overline{x_2}\ \ \overline{x_5}\ \ x_7$

two-watched literal invariant is broken!

i: x_5, d: $\overline{x_1} x_2$, i: x_4, i: x_9, i: $\perp$

x_1, x_3, x_4, x_5, x_6, x_8

literals implied by unit propagation

Proof checking and unit clause deletion

For efficiency, DRAT checkers keep track of literals implied by unit propagation.

Two-watched literal schema

if one watched literal is assigned to false, then the other watched literal must be assigned to true

$\overline{x_2}\ \overline{x_5}\ x_7$

i: x_5, d: $\overline{x_1} x_2$, i: x_4, i: x_9, i: $\perp$

x_1, x_3, x_4, x_5, x_6, x_8

literals implied by unit propagation

Solution DRAT checkers ignore unit clause deletions

clauses whose literals are all falsified except for one satisfied literal