Towards a Semantics of Unsatisfiability Proofs with Inprocessing

Tobias Philipp (TU Dresden)

Adrián Rebola-Pardo (TU Wien)

LPAR-21
Maun, Botswana
May 8th, 2017
Supported by FWF W1255-N23, WWTF VRG11-005, Microsoft Research and Graduate Academy TU Dresden

$$
\begin{array}{ccccc}
xy & xz & xt & yz & yt \\
zt & \bar{x}\,\bar{y}\,\bar{z} & \bar{x}\,\bar{y}\,\bar{t} & \bar{x}\,\bar{z}\,\bar{t} & \bar{y}\,\bar{z}\,\bar{t}
\end{array}
$$

Symmetry breaking clause $\bar{x}y$ can be added without affecting satisfiability

Inprocessing for SAT Solving

xy	xz	xt	yz	yt
zt	$\overline{x}\,\overline{y}\,\overline{z}$	$\overline{x}\,\overline{y}\,\overline{t}$	$\overline{x}\,\overline{z}\,\overline{t}$	$\overline{y}\,\overline{z}\,\overline{t}$

Symmetry breaking clause $\overline{x}y$ can be added without affecting satisfiability

In-/preprocessing techniques replace the formula by an equisatisfiable one

symmetry breaking

clause elimination

parity reasoning

bounded variable addition

cardinality resolution

How to **certify** the result of a SAT solver with inprocessing?

Satisfiability-preserving proof systems

How to certify the result of a SAT solver with inprocessing?

RUP cannot be used!

Theorem

If C is a RAT in F, then whenever F is satisfiable, so is $F \cup \{C\}$.

Satisfiability-preserving proof systems

How to certify the result of a SAT solver with inprocessing?

RUP cannot be used!

Theorem

If C is a RAT in F, then whenever F is satisfiable, so is $F \cup \{C\}$.

$$F \vDash_{\text{sat}} F \cup \{C\}$$

Satisfiability-preserving proof systems

How to certify the result of a SAT solver with inprocessing?

RUP cannot be used!

Theorem

If C is a RAT in F, then whenever F is satisfiable, so is $F \cup \{C\}$.

$$F \vDash_{\text{sat}} F \cup \{C\}$$

Idea build a proof system with $\vDash_{\text{sat}}$ as an invariant

- If C can be inferred, then $F \vDash_{\text{sat}} C$.
- Rules must be simple and efficient to check.
- Inprocessing rules should be easy to express.

The RAT property satisfies these conditions!

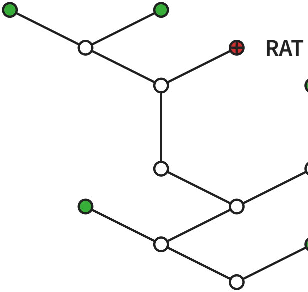
The semantics of DRAT proofs

Clause C is a RAT in F

RAT

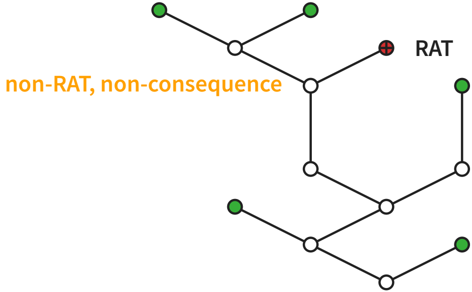
The semantics of DRAT proofs

Clause C is a RAT in F



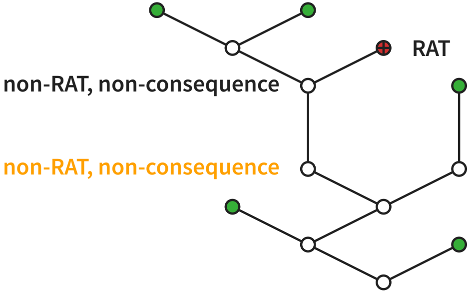
The semantics of DRAT proofs

Clause C is a RAT in F



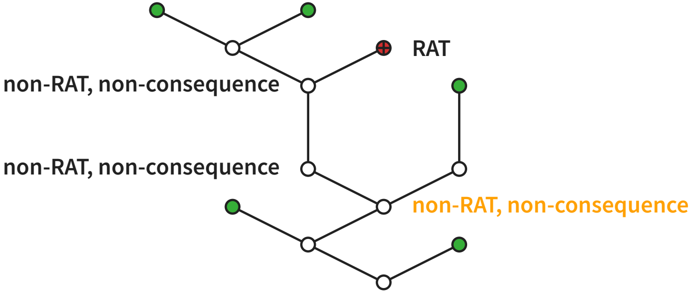
The semantics of DRAT proofs

Clause C is a RAT in F



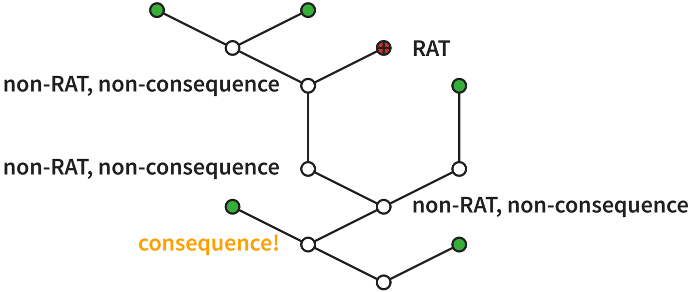
The semantics of DRAT proofs

Clause C is a RAT in F



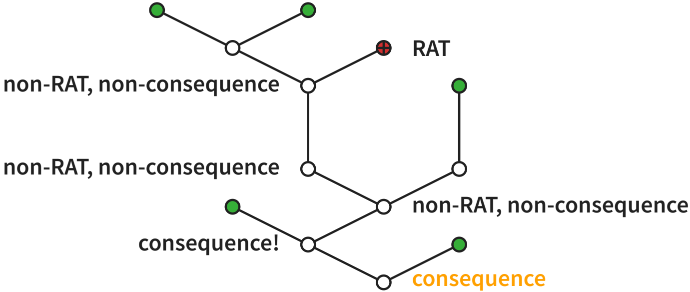
The semantics of DRAT proofs

Clause C is a RAT in F



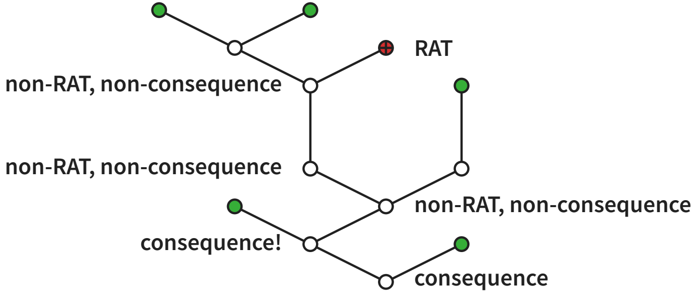
The semantics of DRAT proofs

Clause C is a RAT in F



The semantics of DRAT proofs

Clause C is a RAT in F



Some invariant seems to be preserved...

Question what is the invariant preserved along a DRAT proof?

if derived clauses are not consequences, then what are they?

DRAT proofs

RUP Introduction

Most derived clauses are simply consequences of previously introduced clauses.

Definition

A clause C is a reverse unit propagation (RUP) clause in a CNF formula F if boolean constraint propagation over $F \cup \overline{C}$ leads to contradiction.

RUP Introduction

Most derived clauses are simply consequences of previously introduced clauses.

Definition

A clause C is a reverse unit propagation (RUP) clause in a CNF formula F if boolean constraint propagation over $F \cup \overline{C}$ leads to contradiction.

Characterization

C is a RUP in F if and only if $\Box \in F$, or the least fixed point of ala_F contains complementary clauses, where:

$$\mathrm{ala}_F(C) = C \cup \{\bar{l} \mid \text{for some } D \in F,\ l \in D \text{ and } D \setminus \{l\} \subseteq C\}$$

RUP Introduction

Most derived clauses are simply consequences of previously introduced clauses.

Definition

A clause C is a reverse unit propagation (RUP) clause in a CNF formula F if boolean constraint propagation over $F \cup \overline{C}$ leads to contradiction.

Characterization

C is a RUP in F if and only if $\Box \in F$, or the least fixed point of ala_F contains complementary clauses, where:

$$\mathrm{ala}_F(C) = C \cup \{\bar{l} \mid \text{for some } D \in F,\ l \in D \text{ and } D \setminus \{l\} \subseteq C\}$$

Characterization

C is a RUP in F if and only if it can be proved from F through an SSSR chain:

$$\cfrac{\cfrac{\cfrac{C_0}{D_0}\,\text{sub} \qquad C_1}{D_1}\,\text{ssr}}{\begin{array}{c}\ddots \\ \cfrac{D_{n-1} \qquad C_n}{D_n}\,\text{ssr}\end{array}} = C$$

RAT Introduction and Clause Deletion

Introducing a RAT clause preserves satisfiability

Definition

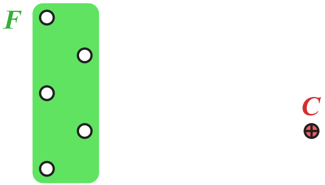
A clause C is a resolution asymmetric tautology (RAT) in a CNF formula F upon a literal $l \in C$ if every resolvent $C \otimes_l D$ for $D \in F$ is a RUP in F.

RAT Introduction and Clause Deletion

Introducing a RAT clause preserves satisfiability

Definition

A clause C is a resolution asymmetric tautology (RAT) in a CNF formula F upon a literal $l \in C$ if every resolvent $C \otimes_l D$ for $D \in F$ is a RUP in F.

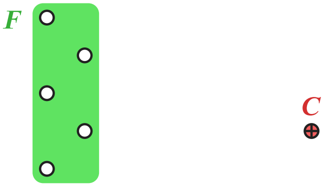


RAT Introduction and Clause Deletion

Introducing a RAT clause preserves satisfiability

Definition

A clause C is a resolution asymmetric tautology (RAT) in a CNF formula F upon a literal $l \in C$ if every resolvent $C \otimes_l D$ for $D \in F$ is a RUP in F.

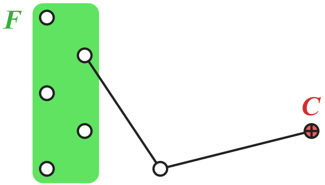


RAT Introduction and Clause Deletion

Introducing a RAT clause preserves satisfiability

Definition

A clause C is a resolution asymmetric tautology (RAT) in a CNF formula F upon a literal $l \in C$ if every resolvent $C \otimes_l D$ for $D \in F$ is a RUP in F.

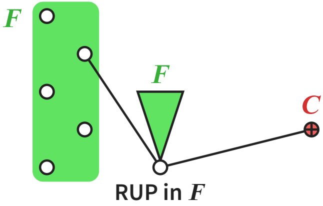


RAT Introduction and Clause Deletion

Introducing a RAT clause preserves satisfiability

Definition

A clause C is a resolution asymmetric tautology (RAT) in a CNF formula F upon a literal $l \in C$ if every resolvent $C \otimes_l D$ for $D \in F$ is a RUP in F.

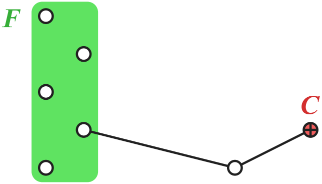


RAT Introduction and Clause Deletion

Introducing a RAT clause preserves satisfiability

Definition

A clause C is a resolution asymmetric tautology (RAT) in a CNF formula F upon a literal $l \in C$ if every resolvent $C \otimes_l D$ for $D \in F$ is a RUP in F.

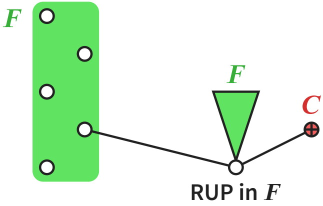


RAT Introduction and Clause Deletion

Introducing a RAT clause preserves satisfiability

Definition

A clause C is a resolution asymmetric tautology (RAT) in a CNF formula F upon a literal $l \in C$ if every resolvent $C \otimes_l D$ for $D \in F$ is a RUP in F.



RAT Introduction and Clause Deletion

Introducing a RAT clause preserves satisfiability

Definition

A clause C is a resolution asymmetric tautology (RAT) in a CNF formula F upon a literal $l \in C$ if every resolvent $C \otimes_l D$ for $D \in F$ is a RUP in F.

Introducing a RAT clause preserves satisfiability

Definition

A clause C is a resolution asymmetric tautology (RAT) in a CNF formula F upon a literal $l \in C$ if every resolvent $C \otimes_l D$ for $D \in F$ is a RUP in F.

Clause deletion preserves satisfiability for arbitrary clauses

- Deletion information is needed for efficiency in BCP
- Arbitrary clause deletion can be expressed

DRAT proofs

Delete Resolution Asymmetric Tautology (DRAT) proof system

$$F \Rightarrow_{\text{DRAT}} G \quad \begin{cases} G = F \cup \{C\} & \begin{cases} C \text{ is a RUP in } F \\ C \text{ is a RAT in } F \end{cases} \\ G = F \setminus \{C\} \end{cases}$$

A DRAT proof of G from F is a sequence of DRAT inferences:

$$F = F_0 \Rightarrow_{\text{DRAT}} F_1 \Rightarrow_{\text{DRAT}} \cdots \Rightarrow_{\text{DRAT}} F_{n-1} \Rightarrow_{\text{DRAT}} F_n = G$$

RATs as Definitions

Defining a Definition

Clauses containing variable x can be regarded as a definition:

$$x \lor C_1 \qquad \dots \qquad x \lor C_n \qquad \overline{x} \lor D_1 \qquad \dots \qquad \overline{x} \lor D_m$$

Defining a Definition

Clauses containing variable x can be regarded as a definition:

$$x \leftarrow \overline{C_1} \qquad \dots \qquad x \leftarrow \overline{C_n} \qquad \overline{x} \leftarrow \overline{D_1} \qquad \dots \qquad \overline{x} \leftarrow \overline{D_m}$$

Defining a Definition

Clauses containing variable x can be regarded as a definition:

$$x \leftarrow \overline{C_1} \qquad \ldots \qquad x \leftarrow \overline{C_n} \qquad \overline{x} \leftarrow \overline{D_1} \qquad \ldots \qquad \overline{x} \leftarrow \overline{D_m}$$

$$x = \begin{cases} 1 & \text{if } \overline{C_1} \\ \vdots & \\ 1 & \text{if } \overline{C_n} \\ 0 & \text{if } \overline{D_1} \\ \vdots & \\ 0 & \text{if } \overline{D_m} \\ \text{don't care} & \text{otherwise} \end{cases}$$

Defining a Definition

Clauses containing variable x can be regarded as a definition:

$$x \leftarrow \overline{C_1} \qquad \ldots \qquad x \leftarrow \overline{C_n} \qquad \overline{x} \leftarrow \overline{D_1} \qquad \ldots \qquad \overline{x} \leftarrow \overline{D_m}$$

$$x = \begin{cases} 1 & \text{if } \overline{C_1} \\ \vdots & \\ 1 & \text{if } \overline{C_n} \\ 0 & \text{if } \overline{D_1} \\ \vdots & \\ 0 & \text{if } \overline{D_m} \\ \text{don't care} & \text{otherwise} \end{cases}$$

Definition refinement to add rule $x \leftarrow C_{n+1}$, we require:

$$\overline{C_{n+1}} \wedge \overline{D_i} \text{ is unsatisfiable for } 1 \leq i \leq m$$

Defining a Definition

Clauses containing variable x can be regarded as a definition:

$$x \leftarrow \overline{C_1} \qquad \dots \qquad x \leftarrow \overline{C_n} \qquad \overline{x} \leftarrow \overline{D_1} \qquad \dots \qquad \overline{x} \leftarrow \overline{D_m}$$

$$x = \begin{cases} 1 & \text{if } \overline{C_1} \\ \vdots & \\ 1 & \text{if } \overline{C_n} \\ 0 & \text{if } \overline{D_1} \\ \vdots & \\ 0 & \text{if } \overline{D_m} \\ \text{don't care} & \text{otherwise} \end{cases}$$

Definition refinement to add rule $x \leftarrow C_{n+1}$, we require:

$$F \models \neg(\overline{C_{n+1}} \wedge \overline{D_i}) \text{ for } 1 \leq i \leq m$$

Defining a Definition

Clauses containing variable x can be regarded as a definition:

$$x \leftarrow \overline{C_1} \qquad \dots \qquad x \leftarrow \overline{C_n} \qquad \overline{x} \leftarrow \overline{D_1} \qquad \dots \qquad \overline{x} \leftarrow \overline{D_m}$$

$$x = \begin{cases} 1 & \text{if } \overline{C_1} \\ \vdots & \\ 1 & \text{if } \overline{C_n} \\ 0 & \text{if } \overline{D_1} \\ \vdots & \\ 0 & \text{if } \overline{D_m} \\ \text{don't care} & \text{otherwise} \end{cases}$$

Definition refinement to add rule $x \leftarrow C_{n+1}$, we require:

$$F \models \overline{C_{n+1}} \vee \overline{D_i} \text{ for } 1 \leq i \leq m$$

Defining a Definition

Clauses containing variable x can be regarded as a definition:

$$x \leftarrow \overline{C_1} \qquad \dots \qquad x \leftarrow \overline{C_n} \qquad \overline{x} \leftarrow \overline{D_1} \qquad \dots \qquad \overline{x} \leftarrow \overline{D_m}$$

$$x = \begin{cases} 1 & \text{if } \overline{C_1} \\ \vdots & \\ 1 & \text{if } \overline{C_n} \\ 0 & \text{if } \overline{D_1} \\ \vdots & \\ 0 & \text{if } \overline{D_m} \\ \text{don't care} & \text{otherwise} \end{cases}$$

Definition refinement to add rule $x \leftarrow C_{n+1}$, we require:

$$F \models (x \vee \overline{C_{n+1}}) \otimes_x (\overline{x} \vee \overline{D_i}) \text{ for } 1 \leq i \leq m$$

Defining a Definition

Clauses containing variable x can be regarded as a definition:

$$x \leftarrow \overline{C_1} \qquad \ldots \qquad x \leftarrow \overline{C_n} \qquad \overline{x} \leftarrow \overline{D_1} \qquad \ldots \qquad \overline{x} \leftarrow \overline{D_m}$$

$$x = \begin{cases} 1 & \text{if } \overline{C_1} \\ \vdots & \\ 1 & \text{if } \overline{C_n} \\ 0 & \text{if } \overline{D_1} \\ \vdots & \\ 0 & \text{if } \overline{D_m} \\ \text{don't care} & \text{otherwise} \end{cases}$$

Definition refinement to add rule $x \leftarrow C_{n+1}$, we require:

$$F \vDash (x \vee \overline{C_{n+1}}) \otimes_x D \text{ for every clause } D \in F$$

Defining a Definition

Clauses containing variable x can be regarded as a definition:

$$x \leftarrow \overline{C_1} \qquad \ldots \qquad x \leftarrow \overline{C_n} \qquad \overline{x} \leftarrow \overline{D_1} \qquad \ldots \qquad \overline{x} \leftarrow \overline{D_m}$$

$$x = \begin{cases} 1 & \text{if } \overline{C_1} \\ \vdots & \\ 1 & \text{if } \overline{C_n} \\ 0 & \text{if } \overline{D_1} \\ \vdots & \\ 0 & \text{if } \overline{D_m} \\ \text{don't care} & \text{otherwise} \end{cases}$$

Definition refinement to add rule $x \leftarrow C_{n+1}$, we require:

$$F \vDash (x \vee \overline{C_{n+1}}) \otimes_x D \text{ for every clause } D \in F$$

Definition

A clause C is a resolution asymmetric tautology (RAT) in a CNF formula F upon a literal $l \in C$ if every resolvent $C \otimes_l D$ for $D \in F$ is a RUP in F.

Defining a Definition

Clauses containing variable x can be regarded as a definition:

$$x \leftarrow \overline{C_1} \qquad \ldots \qquad x \leftarrow \overline{C_n} \qquad \overline{x} \leftarrow \overline{D_1} \qquad \ldots \qquad \overline{x} \leftarrow \overline{D_m}$$

$$x = \begin{cases} 1 & \text{if } \overline{C_1} \\ \vdots & \\ 1 & \text{if } \overline{C_n} \\ 0 & \text{if } \overline{D_1} \\ \vdots & \\ 0 & \text{if } \overline{D_m} \\ \text{don't care} & \text{otherwise} \end{cases}$$

Definition refinement to add rule $x \leftarrow C_{n+1}$, we require:

$$F \vDash (x \vee \overline{C_{n+1}}) \otimes_x D \text{ for every clause } D \in F$$

Definition

A clause C is a ~~resolution asymmetric tautology (RAT)~~ resolution consequence (RC) in a CNF formula F upon a literal $l \in C$ if every resolvent $C \otimes_l D$ for $D \in F$ is a ~~RUP in F~~ consequence of F.

The Semantics of Definitions

Question when is a definition correct from the semantic perspective?

- A definition changes the interpretation to force a truth value under some conditions.
- After the change, the interpretation still satisfies the original constraints.

The Semantics of Definitions

Question when is a definition correct from the semantic perspective?

- A definition changes the interpretation to force a truth value under some conditions.
- After the change, the interpretation still satisfies the original constraints.

Mutation rule $l \coloneq\!\!- \; Q$

Interpretation mutation

$$I \lhd (l \coloneq\!\!- \; Q) = \begin{cases} I + l & \text{if } I \models Q \\ I & \text{if } I \not\models Q \end{cases}$$

Definition

$l \coloneq\!\!- \; Q$ is a definition refinement of F if, whenever $I \models F$ holds, then $I \lhd (l \coloneq\!\!- \; Q) \models F$ holds as well.

Definition Refinements are Resolution Consequences

Definition

A clause C is a resolution consequence in F upon l if every resolvent $C \otimes_l D$ for $D \in F$ is a consequence of F.

Definition

$l :- Q$ is a definition refinement of F if, whenever $I \models F$ holds, then $I \lhd (l :- Q) \models F$ holds as well.

Definition Refinements are Resolution Consequences

Definition

A clause C is a resolution consequence in F upon l if every resolvent $C \otimes_l D$ for $D \in F$ is a consequence of F.

Definition

$l :- Q$ is a definition refinement of F if, whenever $I \vDash F$ holds, then $I \triangleleft (l :- Q) \vDash F$ holds as well.

Theorem

$C \vee l$ is a RC in F upon l if and only if $l :- \overline{C}$ is a definition refinement.

Definition Refinements are Resolution Consequences

Definition

A clause C is a resolution consequence in F upon l if every resolvent $C \otimes_l D$ for $D \in F$ is a consequence of F.

Definition

$l :\!- Q$ is a definition refinement of F if, whenever $I \models F$ holds, then $I \lhd (l :\!- Q) \models F$ holds as well.

Theorem

$C \vee l$ is a RC in F upon l if and only if $l :\!- \overline{C}$ is a definition refinement.

Example z is defined as true when x holds, and false when y holds

correct only if $\overline{x} \wedge \overline{y}$ is disallowed

$$\overline{x}\,\overline{y} \xrightarrow{z \,:\!-\, \overline{x}} zx \xrightarrow{\overline{z} \,:\!-\, \overline{y}} \overline{z}y$$

Definition Refinements are Resolution Consequences

Definition

A clause C is a resolution consequence in F upon l if every resolvent $C \otimes_l D$ for $D \in F$ is a consequence of F.

Definition

$l \mathrel{:\!-} Q$ is a definition refinement of F if, whenever $I \vDash F$ holds, then $I \lhd (l \mathrel{:\!-} Q) \vDash F$ holds as well.

Theorem

$C \vee l$ is a RC in F upon l if and only if $l \mathrel{:\!-} \overline{C}$ is a definition refinement.

Example z is defined as true when x holds, and false when y holds

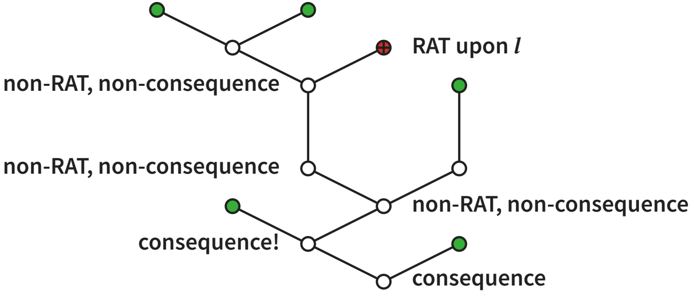
correct only if $\overline{x} \wedge \overline{y}$ is disallowed

$$\overline{x}\,\overline{y} \xrightarrow{\;z \mathrel{:\!-} \overline{x}\;} z\,x \xrightarrow{\;\overline{z} \mathrel{:\!-} \overline{y}\;} \overline{z}\,y$$

$$I = \{x, \overline{y}, z\} \qquad I' = \{x, \overline{y}, z\} \qquad I'' = \{x, \overline{y}, \overline{z}\}$$

Definitional Formulas

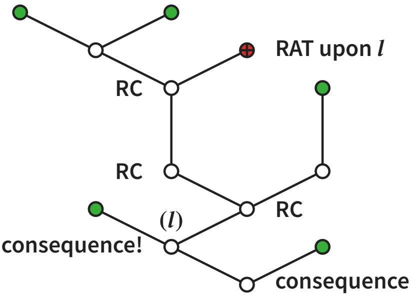
Resolution Consequences as Proof Invariants



Theorem If C is a RC in F upon l, and $F \wedge C \vDash D$, then either:

- $F \vDash D$
- D is an RC in F upon l

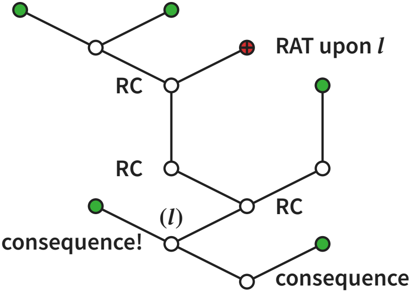
Resolution Consequences as Proof Invariants



Theorem If C is a RC in F upon l, and $F \wedge C \vDash D$, then either:

- $F \vDash D$
- D is an RC in F upon l

Resolution Consequences as Proof Invariants



Theorem If C is a RC in F upon l, and $F \wedge C \vDash D$, then either:

- $F \vDash D$
- D is an RC in F upon l

Question which definitions are required for each derived clause?

Definitional formulas

Formula trees

$$\begin{array}{cc} xyz & \overline{x}yz \\ x\overline{y}z & \overline{x}\,\overline{y}z \end{array} \xrightarrow{\quad w \coloneq \overline{y} \quad} \begin{array}{c} wy \\ wz \end{array} \xrightarrow{\quad \overline{w} \coloneq \overline{z} \quad} \overline{w}z$$

Definitional formulas

Formula trees

$$I \models \begin{matrix} xyz & \overline{x}yz \\ x\overline{y}z & \overline{x}\,\overline{y}z \end{matrix} \xrightarrow{\;w \coloneq \overline{y}\;} \begin{matrix} wy \\ wz \end{matrix} \xrightarrow{\;\overline{w} \coloneq \overline{z}\;} \overline{w}z$$

Definitional formulas

Formula trees

$$I \models \boxed{\begin{matrix} xyz & \overline{x}yz \\ x\overline{y}z & \overline{x}\,\overline{y}z \end{matrix}} \xrightarrow{w \;:\!-\; \overline{y}} \begin{matrix} wy \\ wz \end{matrix} \xrightarrow{\overline{w} \;:\!-\; \overline{z}} \overline{w}z$$

Definitional formulas

Formula trees

$$I \lhd (w \coloneq \overline{y}) \quad \models \quad \boxed{\begin{array}{cc} xyz & \overline{x}yz \\ x\overline{y}z & \overline{x}\,\overline{y}z \end{array} \xrightarrow{\;w \,\coloneq\, \overline{y}\;} \begin{array}{c} wy \\ wz \end{array}} \xrightarrow{\;\overline{w} \,\coloneq\, \overline{z}\;} \overline{w}z$$

Definitional formulas

Formula trees

$$I \triangleleft (w :- \overline{y}) \triangleleft (\overline{w} :- \overline{z}) \quad \models \quad \begin{matrix} xyz & \overline{x}yz \\ x\overline{y}z & \overline{x}\,\overline{y}z \end{matrix} \xrightarrow{w \,:-\, \overline{y}} \begin{matrix} wy \\ wz \end{matrix} \xrightarrow{\overline{w} \,:-\, \overline{z}} \overline{w}z$$

Definitional formulas

Formula trees

$$\begin{matrix} xyz & \bar{x}yz \\ x\bar{y}z & \bar{x}\bar{y}z \end{matrix} \xrightarrow{\quad w \;\coloneq\; \bar{y} \quad} \begin{matrix} wy \\ wz \end{matrix} \xrightarrow{\quad \bar{w} \;\coloneq\; \bar{z} \quad} \bar{w}z$$

Stratification derived clauses can be split based on which definitions they depend upon

Example $z = wz \otimes_w \bar{w}z$

Definitional formulas

Formula trees

$$\begin{matrix} xyz & \bar{x}yz \\ x\bar{y}z & \bar{x}\bar{y}z \end{matrix} \xrightarrow{\quad w \;:\!-\; \bar{y} \quad} \begin{matrix} wy \\ wz \end{matrix} \xrightarrow{\quad \bar{w} \;:\!-\; \bar{z} \quad} \begin{matrix} \bar{w}z \\ z \end{matrix}$$

Stratification derived clauses can be split based on which definitions they depend upon

Example $z = wz \otimes_w \bar{w}z$

Definitional formulas

Formula trees

$$\begin{matrix} xyz & \overline{x}yz \\ x\overline{y}z & \overline{x}\,\overline{y}z \end{matrix} \xrightarrow{\;w \;\coloneq\; \overline{y}\;} \begin{matrix} wy \\ wz \end{matrix} \xrightarrow{\;\overline{w} \;\coloneq\; \overline{z}\;} \begin{matrix} \overline{w}z \\ z \end{matrix}$$

Stratification derived clauses can be split based on which definitions they depend upon

Example $z = wz \otimes_w \overline{w}z$ actually requires no definition

Definitional formulas

Formula trees

$$\begin{matrix} xyz & \overline{x}yz \\ x\overline{y}z & \overline{x}\,\overline{y}z \end{matrix} \xrightarrow{\;w \coloneq \overline{y}\;} \begin{matrix} wy & \\ wz & z \end{matrix} \xrightarrow{\;\overline{w} \coloneq \overline{z}\;} \overline{w}z$$

Stratification derived clauses can be split based on which definitions they depend upon

Example $z = wz \otimes_w \overline{w}z$ actually requires no definition

Definitional formulas

Formula trees

$$\begin{matrix} xyz & \bar{x}yz \\ x\bar{y}z & \bar{x}\bar{y}z \end{matrix} \quad z \xrightarrow{w \;:\!-\; \bar{y}} \begin{matrix} wy \\ wz \end{matrix} \xrightarrow{\bar{w} \;:\!-\; \bar{z}} \bar{w}z$$

Stratification derived clauses can be split based on which definitions they depend upon

Example $z = wz \otimes_w \bar{w}z$ actually requires no definition

Definitional formulas

Formula trees

$$\begin{matrix} xyz & \overline{x}yz \\ x\overline{y}z & \overline{x}\,\overline{y}z \end{matrix} \xrightarrow{\;w \;\coloneq\; \overline{y}\;} \begin{matrix} wy \\ wz \end{matrix} \xrightarrow{\;\overline{w} \;\coloneq\; \overline{z}\;} \overline{w}z$$

Stratification derived clauses can be split based on which definitions they depend upon

Example $z = wz \otimes_w \overline{w}z$ actually requires no definition

Equivalence- and definitionality-preserving inferences

- Consequence introduction
  simulates RUP introduction
- Definition extension
  simulates RAT introduction
- Clause upgrade
  applied on resolvents upon pivot literals

Definitions and deletion

Clause deletion can make correct definitions incorrect:

$$\begin{matrix} xyz & \overline{x}yz \\ x\overline{y}z & \overline{x}\,\overline{y}z \end{matrix} \xrightarrow{\;w \;\coloneq\; \overline{y}\;} \begin{matrix} wy \\ wz \end{matrix} \xrightarrow{\;\overline{w} \;\coloneq\; \overline{z}\;} \begin{matrix} \overline{w}z \\ z \end{matrix}$$

Definitions and deletion

Clause deletion can make correct definitions incorrect:

$$\begin{matrix} xyz & \overline{x}yz \\ x\overline{y}z & \end{matrix} \xrightarrow{\quad w \coloneq \overline{y} \quad} \begin{matrix} wy \\ wz \end{matrix} \xrightarrow{\quad \overline{w} \coloneq \overline{z} \quad} \begin{matrix} \overline{w}z \\ z \end{matrix}$$

Definitions and deletion

Clause deletion can make correct definitions incorrect:

$$\begin{array}{l} xyz \quad \overline{x}yz \\ x\overline{y}z \end{array} \xrightarrow{w \;\coloneq\; \overline{y}} \begin{array}{l} wy \\ wz \end{array} \xrightarrow{\overline{w} \;\coloneq\; \overline{z}} \begin{array}{c} \overline{w}z \\ z \end{array}$$

$$I = \{x, \overline{y}, \overline{z}, \overline{w}\} \qquad I' = \{x, \overline{y}, \overline{z}, w\} \qquad I = \{x, \overline{y}, \overline{z}, \overline{w}\}$$

Definitions and deletion

Clause deletion can make correct definitions incorrect:

$$\begin{array}{ll} xyz & \overline{x}yz \\ x\overline{y}z & \end{array} \xrightarrow{\;w \,:\!-\, \overline{y}\;} \begin{array}{l} wy \\ wz \end{array} \xrightarrow{\;\overline{w} \,:\!-\, \overline{z}\;} \begin{array}{c} \overline{w}z \\ z \end{array}$$

$$I = \{x, \overline{y}, \overline{z}, \overline{w}\} \qquad I' = \{x, \overline{y}, \overline{z}, w\} \qquad I = \{x, \overline{y}, \overline{z}, \overline{w}\}$$

Question is there any invariant throughout DRAT proofs stronger than $\vDash_{\text{sat}}$?

Theorem G is DRAT-derivable from F if and only if $F \vDash_{\text{sat}} G$

no stronger invariant within propositional logic and unrestricted deletion

Definitions and deletion

Clause deletion can make correct definitions incorrect:

$$\begin{array}{ll} xyz & \bar{x}yz \\ x\bar{y}z & \end{array} \xrightarrow{w \,:\!-\, \bar{y}} \begin{array}{l} wy \\ wz \end{array} \xrightarrow{\bar{w} \,:\!-\, \bar{z}} \begin{array}{c} \bar{w}z \\ z \end{array}$$

$$I = \{x, \bar{y}, \bar{z}, \bar{w}\} \qquad I' = \{x, \bar{y}, \bar{z}, w\} \qquad I = \{x, \bar{y}, \bar{z}, \bar{w}\}$$

Question is there any invariant throughout DRAT proofs stronger than $\vDash_{\text{sat}}$?

Theorem G is DRAT-derivable from F if and only if $F \vDash_{\text{sat}} G$

no stronger invariant within propositional logic and unrestricted deletion

Possible workarounds

- Restrict deletion to realistic situations
- Construe a DRAT proof as a derivation on DQBF clauses.
- Change mutation rules upon deletion.

Conclusion

Conclusion

DRAT proofs are widely used for validation of SAT solvers' results, but its semantics are not well understood.

Conclusion

DRAT proofs are widely used for validation of SAT solvers' results, but its semantics are not well understood.

In the absence of deletion, proofs correspond to derivations on definitional formulas. In particular, resolution consequences correspond exactly to a form of permissive definition.

Conclusion

DRAT proofs are widely used for validation of SAT solvers' results, but its semantics are not well understood.

In the absence of deletion, proofs correspond to derivations on definitional formulas. In particular, resolution consequences correspond exactly to a form of permissive definition.

Future work

can we generate resolution/RUP proofs from a DRAT proof? yes!

Conclusion

DRAT proofs are widely used for validation of SAT solvers' results, but its semantics are not well understood.

In the absence of deletion, proofs correspond to derivations on definitional formulas. In particular, resolution consequences correspond exactly to a form of permissive definition.

Future work

can we generate resolution/RUP proofs from a DRAT proof? yes!

Actual future work

can we find a semantics stronger than satisfiability preservation for DRAT proofs?