Short proofs without interference

Adrián Rebola-Pardo

TU Wien, JKU Linz

Orsay, France
14 September 2022

Supported by FWF 10.55776/COE12

Trying to merge proofs...

all unsat, over the same variables

F_1 F_2 F_3

Trying to merge proofs...

$u_1 \vee F_1$ $\quad$ $u_2 \vee F_2$ $\quad$ $u_3 \vee F_3$

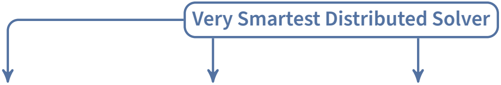
Trying to merge proofs...

still unsat!

$u_1 \vee F_1$ $\quad u_2 \vee F_2$ $\quad u_3 \vee F_3$ $\quad \overline{u_1} \vee \overline{u_2} \vee \overline{u_3}$

Trying to merge proofs...

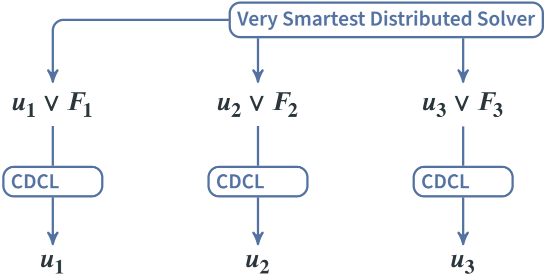
$u_1 \lor F_1$ $\quad$ $u_2 \lor F_2$ $\quad$ $u_3 \lor F_3$ $\quad$ $\overline{u_1} \lor \overline{u_2} \lor \overline{u_3}$



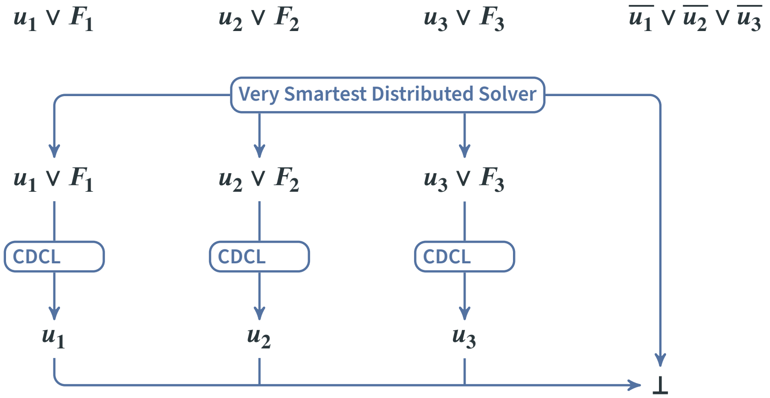
$u_1 \lor F_1$ $\quad$ $u_2 \lor F_2$ $\quad$ $u_3 \lor F_3$

Trying to merge proofs...

$u_1 \vee F_1$ $\qquad$ $u_2 \vee F_2$ $\qquad$ $u_3 \vee F_3$ $\qquad$ $\overline{u_1} \vee \overline{u_2} \vee \overline{u_3}$

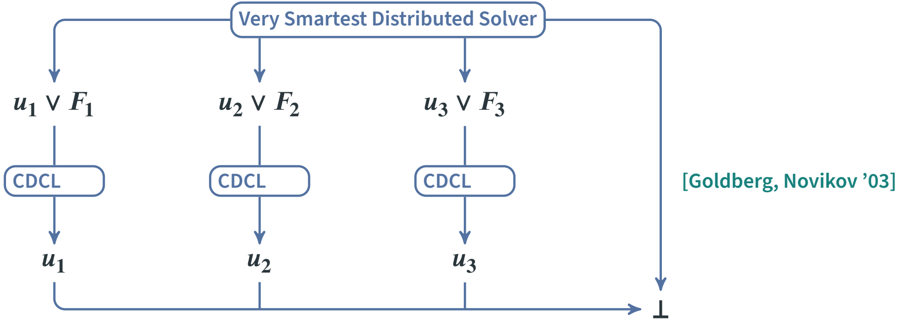


Trying to merge proofs...



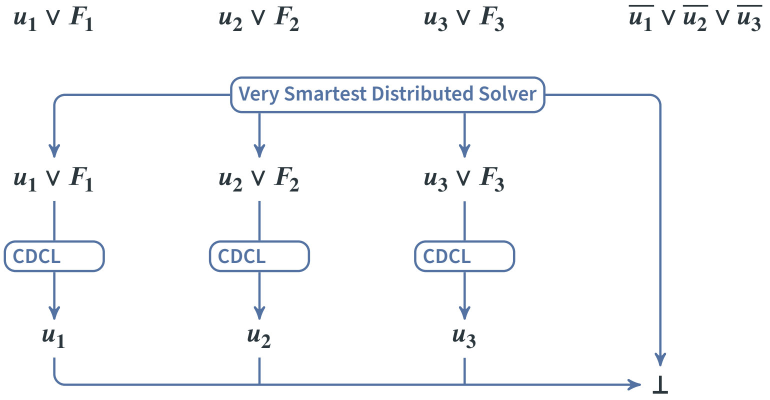
Trying to merge proofs...

$u_1 \vee F_1 \qquad u_2 \vee F_2 \qquad u_3 \vee F_3 \qquad \overline{u_1} \vee \overline{u_2} \vee \overline{u_3}$



$$\pi_1 : u_1 \vee F_1 \vdash u_1$$
$$\pi_2 : u_2 \vee F_2 \vdash u_2$$
$$\pi_3 : u_3 \vee F_3 \vdash u_3$$

Trying to merge proofs...



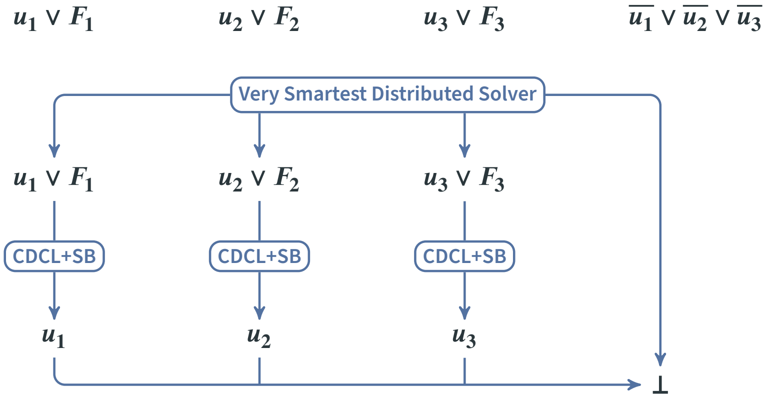
$\pi_1 : u_1 \lor F_1 \vdash u_1$

$\pi_2 : u_2 \lor F_2 \vdash u_2$

$\pi_3 : u_3 \lor F_3 \vdash u_3$

$\pi : u_1 \land u_2 \land u_3 \land (\overline{u_1} \lor \overline{u_2} \lor \overline{u_3}) \vdash \bot$

Trying to merge proofs...



$\pi_1 : u_1 \vee F_1 \vdash u_1$

$\pi_2 : u_2 \vee F_2 \vdash u_2$

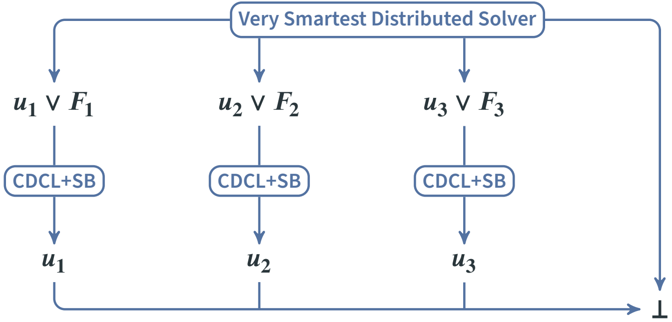
$\pi_3 : u_3 \vee F_3 \vdash u_3$

$\pi : u_1 \wedge u_2 \wedge u_3 \wedge (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3}) \vdash \bot$

Trying to merge proofs...

$\sigma_1 = \{x_1 \leftrightarrow y_1\}$ $\sigma_2 = \{x_2 \leftrightarrow y_2\}$ $\sigma_3 = \{x_3 \leftrightarrow y_3\}$

$u_1 \lor F_1$ $u_2 \lor F_2$ $u_3 \lor F_3$ $\overline{u_1} \lor \overline{u_2} \lor \overline{u_3}$



$\pi_1 : u_1 \lor F_1 \vdash u_1$

$\pi_2 : u_2 \lor F_2 \vdash u_2$

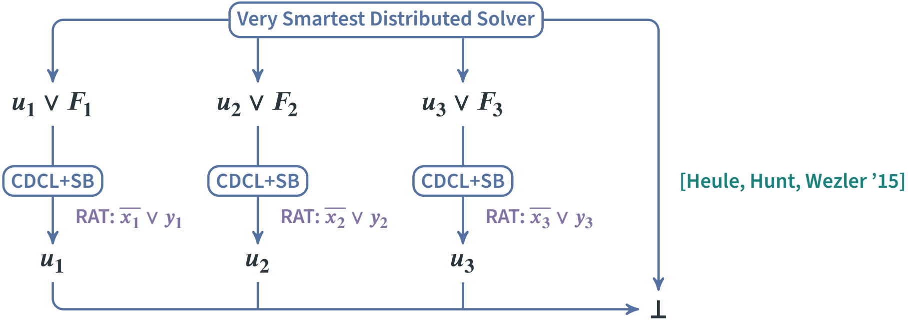
$\pi_3 : u_3 \lor F_3 \vdash u_3$

$\pi : u_1 \land u_2 \land u_3 \land (\overline{u_1} \lor \overline{u_2} \lor \overline{u_3}) \vdash \bot$

Trying to merge proofs...

$\sigma_1 = \{x_1 \leftrightarrow y_1\}$ $\sigma_2 = \{x_2 \leftrightarrow y_2\}$ $\sigma_3 = \{x_3 \leftrightarrow y_3\}$

$u_1 \vee F_1$ $u_2 \vee F_2$ $u_3 \vee F_3$ $\overline{u_1} \vee \overline{u_2} \vee \overline{u_3}$



$\pi_1 : u_1 \vee F_1 \vdash u_1$

$\pi_2 : u_2 \vee F_2 \vdash u_2$

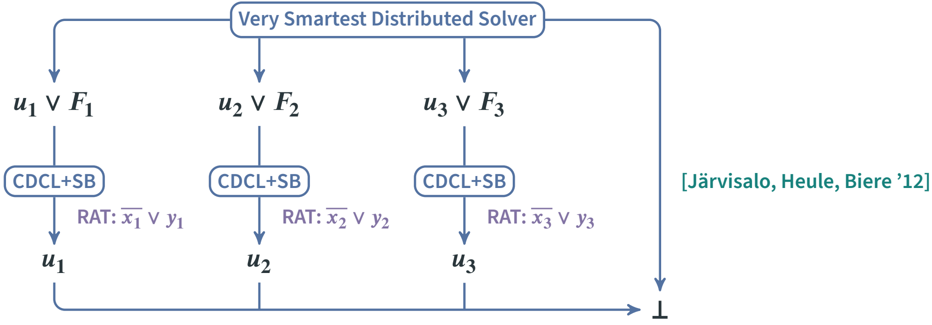
$\pi_3 : u_3 \vee F_3 \vdash u_3$

$\pi : u_1 \wedge u_2 \wedge u_3 \wedge (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3}) \vdash \bot$

Trying to merge proofs...

$\sigma_1 = \{x_1 \leftrightarrow y_1\}$ $\sigma_2 = \{x_2 \leftrightarrow y_2\}$ $\sigma_3 = \{x_3 \leftrightarrow y_3\}$

$u_1 \vee F_1$ $u_2 \vee F_2$ $u_3 \vee F_3$ $\overline{u_1} \vee \overline{u_2} \vee \overline{u_3}$



$\pi_1 : u_1 \vee F_1 \vdash u_1$

$\pi_2 : u_2 \vee F_2 \vdash u_2$

$\pi_3 : u_3 \vee F_3 \vdash u_3$

$\pi : u_1 \wedge u$ NOPE $\vee \overline{u_3}) \vdash \bot$

… and failing because of interference

What are DRAT proofs really doing?

$\pi : F \vdash G$ proves that for each $I \models F$ we have $\mathrm{mut}(I) \models F$

… and failing because of interference

What are DRAT proofs really doing?

$\pi : F \vdash G$ proves that for each $I \models F$ we have $\mathrm{mut}(I) \models F$

mut is a sequence of operations like if $I \models T$, then $I \coloneqq I \circ \sigma$ [RP, Suda '18]

What are DRAT proofs really doing?

$\pi : F \vdash G$ proves that for each $I \vDash F$ we have $\mathrm{mut}(I) \vDash F$

mut is a sequence of operations like if $I \vDash T$, then $I := I \circ \sigma$ [RP, Suda '18]

each SR addition of C upon σ introduces a new operation with $T = \overline{C}$

… and failing because of interference

What are DRAT proofs really doing?

$\pi : F \vdash G$ proves that for each $I \models F$ we have $\mathrm{mut}(I) \models F$

mut is a sequence of operations like if $I \models T$, then $I := I \circ \sigma$ [RP, Suda '18]

each SR addition of C upon σ introduces a new operation with $T = \overline{C}$

$$\pi_1 : u_1 \vee F_1 \vdash (u_1 \vee F_1) \wedge (\overline{x_1} \vee y_1) \vdash u_1$$

… and failing because of interference

What are DRAT proofs really doing?

$\pi : F \vdash G$ proves that for each $I \models F$ we have $\mathrm{mut}(I) \models F$

mut is a sequence of operations like if $I \models T$, then $I \coloneqq I \circ \sigma$ [RP, Suda '18]

each SR addition of C upon σ introduces a new operation with $T = \overline{C}$

$\pi_1 : u_1 \vee F_1 \vdash (u_1 \vee F_1) \wedge (\overline{x_1} \vee y_1) \vdash u_1$ $I \models u_1 \vee F_1 \Rightarrow \mathrm{mut}_1(I) \models u_1$

… and failing because of interference

What are DRAT proofs really doing?

$\pi : F \vdash G$ proves that for each $I \models F$ we have $\mathrm{mut}(I) \models F$

mut is a sequence of operations like if $I \models T$, then $I := I \circ \sigma$ [RP, Suda '18]

each SR addition of C upon σ introduces a new operation with $T = \overline{C}$

$\pi_1 : u_1 \lor F_1 \vdash (u_1 \lor F_1) \land (\overline{x_1} \lor y_1) \vdash u_1$ $\quad I \models u_1 \lor F_1 \Rightarrow \mathrm{mut}_1(I) \models u_1$

mut_1 is "if $I \models x_1 \land \overline{y_1}$ then $I := I \circ \{x_1 \leftrightarrow y_1\}$"

… and failing because of interference

What are DRAT proofs really doing?

$\pi : F \vdash G$ proves that for each $I \models F$ we have $\mathrm{mut}(I) \models F$

mut is a sequence of operations like if $I \models T$, then $I := I \circ \sigma$ [RP, Suda '18]

each SR addition of C upon σ introduces a new operation with $T = \overline{C}$

$$\pi_1 : u_1 \vee F_1 \vdash (u_1 \vee F_1) \wedge (\overline{x_1} \vee y_1) \vdash u_1 \qquad I \models u_1 \vee F_1 \Rightarrow \mathrm{mut}_1(I) \models u_1$$
$$\pi_2 : u_2 \vee F_2 \vdash (u_2 \vee F_2) \wedge (\overline{x_2} \vee y_2) \vdash u_2 \qquad I \models u_2 \vee F_2 \Rightarrow \mathrm{mut}_2(I) \models u_2$$
$$\pi_3 : u_3 \vee F_3 \vdash (u_3 \vee F_3) \wedge (\overline{x_3} \vee y_3) \vdash u_3 \qquad I \models u_3 \vee F_3 \Rightarrow \mathrm{mut}_3(I) \models u_3$$

... and failing because of interference

What are DRAT proofs really doing?

$\pi : F \vdash G$ proves that for each $I \models F$ we have $\mathrm{mut}(I) \models F$

mut is a sequence of operations like if $I \models T$, then $I := I \circ \sigma$ [RP, Suda '18]

each SR addition of C upon σ introduces a new operation with $T = \overline{C}$

$\pi_1 : u_1 \vee F_1 \vdash (u_1 \vee F_1) \wedge (\overline{x_1} \vee y_1) \vdash u_1$ $\quad I \models u_1 \vee F_1 \Rightarrow \quad I \models u_1$

$\pi_2 : u_2 \vee F_2 \vdash (u_2 \vee F_2) \wedge (\overline{x_2} \vee y_2) \vdash u_2$ $\quad I \models u_2 \vee F_2 \Rightarrow \quad I \models u_2$

$\pi_3 : u_3 \vee F_3 \vdash (u_3 \vee F_3) \wedge (\overline{x_3} \vee y_3) \vdash u_3$ $\quad I \models u_3 \vee F_3 \Rightarrow \quad I \models u_3$

what we need is this!

… and failing because of interference

What are DRAT proofs really doing?

$\pi : F \vdash G$ proves that for each $I \models F$ we have $\mathrm{mut}(I) \models F$

mut is a sequence of operations like if $I \models T$, then $I := I \circ \sigma$ [RP, Suda '18]

each SR addition of C upon σ introduces a new operation with $T = \overline{C}$

$\pi_1 : u_1 \vee F_1 \vdash (u_1 \vee F_1) \wedge (\overline{x_1} \vee y_1) \vdash u_1$ $\quad I \models u_1 \vee F_1 \Rightarrow \mathrm{mut}_1(I) \models u_1$

$\pi_2 : u_2 \vee F_2 \vdash (u_2 \vee F_2) \wedge (\overline{x_2} \vee y_2) \vdash u_2$ $\quad I \models u_2 \vee F_2 \Rightarrow \mathrm{mut}_2(I) \models u_2$

$\pi_3 : u_3 \vee F_3 \vdash (u_3 \vee F_3) \wedge (\overline{x_3} \vee y_3) \vdash u_3$ $\quad I \models u_3 \vee F_3 \Rightarrow \mathrm{mut}_3(I) \models u_3$

Solution 1 break the symmetries before splitting

This is not even sound in general!

… and failing because of interference

What are DRAT proofs really doing?

$\pi : F \vdash G$ **proves that** for each $I \models F$ we have $\mathrm{mut}(I) \models F$

mut **is a sequence of operations like** if $I \models T$, then $I := I \circ \sigma$ [RP, Suda '18]

each SR addition of C upon σ introduces a new operation with $T = \overline{C}$

$\pi_1 : u_1 \vee F_1 \vdash (u_1 \vee F_1) \wedge (\overline{x_1} \vee y_1) \vdash u_1$ $I \models u_1 \vee F_1 \Rightarrow \mathrm{mut}_1(I) \models u_1$

$\pi_2 : u_2 \vee F_2 \vdash (u_2 \vee F_2) \wedge (\overline{x_2} \vee y_2) \vdash u_2$ $I \models u_2 \vee F_2 \Rightarrow \mathrm{mut}_2(I) \models u_2$

$\pi_3 : u_3 \vee F_3 \vdash (u_3 \vee F_3) \wedge (\overline{x_3} \vee y_3) \vdash u_3$ $I \models u_3 \vee F_3 \Rightarrow \mathrm{mut}_3(I) \models u_3$

Solution 1 **break the symmetries before splitting**

This is not even sound in general!

Solution 2 **allow nesting DRAT refutations**

A refutation $(u_1 \vee F_1) \wedge \overline{u_1} \vdash \bot$ **is also a proof** $u_1 \vee F_1 \vdash u_1$

… and failing because of interference

What are DRAT proofs really doing?

$\pi : F \vdash G$ proves that for each $I \vDash F$ we have $\mathrm{mut}(I) \vDash F$

mut is a sequence of operations like if $I \vDash T$, then $I := I \circ \sigma$ [RP, Suda '18]

each SR addition of C upon σ introduces a new operation with $T = \overline{C}$

$$\pi_1 : u_1 \lor F_1 \vdash (u_1 \lor F_1) \land (\overline{x_1} \lor y_1) \vdash u_1 \qquad I \vDash u_1 \lor F_1 \Rightarrow \mathrm{mut}_1(I) \vDash u_1$$
$$\pi_2 : u_2 \lor F_2 \vdash (u_2 \lor F_2) \land (\overline{x_2} \lor y_2) \vdash u_2 \qquad I \vDash u_2 \lor F_2 \Rightarrow \mathrm{mut}_2(I) \vDash u_2$$
$$\pi_3 : u_3 \lor F_3 \vdash (u_3 \lor F_3) \land (\overline{x_3} \lor y_3) \vdash u_3 \qquad I \vDash u_3 \lor F_3 \Rightarrow \mathrm{mut}_3(I) \vDash u_3$$

Solution 1 break the symmetries before splitting

This is not even sound in general!

Solution 2 allow nesting DRAT **refutations**

A refutation $(u_1 \lor F_1) \land \overline{u_1} \vdash \bot$ is also a proof $u_1 \lor F_1 \vdash u_1$

- requires solver to know conclusion in advance
- repeated work if multiple clauses are derived

Accumulated formulas are not your friend

The mutation operator $\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \models T$, or I otherwise.

Accumulated formulas are not your friend

The mutation operator $\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \models T$, or I otherwise.

$I \models \nabla(T :- \sigma).C$ iff $\nabla(T :- \sigma)(I) \models C$ [RP, Suda ’18] [RP ’23]

Accumulated formulas are not your friend

The mutation operator $\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \vDash T$, or I otherwise.

$I \vDash \nabla(T :- \sigma).C$ iff $\nabla(T :- \sigma)(I) \vDash C$ [RP, Suda '18] [RP '23]

F

Accumulated formulas are not your friend

The mutation operator $\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \models T$, or I otherwise.

$I \models \nabla(T :- \sigma).C$ iff $\nabla(T :- \sigma)(I) \models C$ [RP, Suda '18] [RP '23]

F

$u_1 \lor F_1$ (by deletion)

Accumulated formulas are not your friend

The mutation operator $\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \models T$, or I otherwise.

$I \models \nabla(T :- \sigma).C$ iff $\nabla(T :- \sigma)(I) \models C$ [RP, Suda '18] [RP '23]

F

$u_1 \vee F_1$ (by deletion)

$\nabla(\overline{B_1} :- \sigma_1).u_1 \vee F_1$ (by SR)

[Buss, Thapen '19]

Accumulated formulas are not your friend

The mutation operator $\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \models T$, or I otherwise.

$$I \models \nabla(T :- \sigma).C \quad \text{iff} \quad \nabla(T :- \sigma)(I) \models C$$

[RP, Suda '18] [RP '23]

F

$u_1 \vee F_1$ (by deletion)

$\nabla(\overline{B_1} :- \sigma_1).\, u_1 \vee F_1$ (by SR)

$\nabla(\overline{B_1} :- \sigma_1).\, B_1$ (by SR)

Accumulated formulas are not your friend

The mutation operator $\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \models T$, or I otherwise.

$I \models \nabla(T :- \sigma).C$ iff $\nabla(T :- \sigma)(I) \models C$ [RP, Suda '18] [RP '23]

F	
$u_1 \vee F_1$	(by deletion)
$\nabla(\overline{B_1} :- \sigma_1). u_1 \vee F_1$	(by SR)
$\nabla(\overline{B_1} :- \sigma_1). B_1$	(by SR)
$\nabla(\overline{B_1} :- \sigma_1). u_1$	(by resolution)

Accumulated formulas are not your friend

The mutation operator $\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \models T$, or I otherwise.

$I \models \nabla(T :- \sigma).C$ iff $\nabla(T :- \sigma)(I) \models C$ [RP, Suda '18] [RP '23]

F	
$u_1 \vee F_1$	(by deletion)
$\nabla(\overline{B_1} :- \sigma_1).\, u_1 \vee F_1$	(by SR)
$\nabla(\overline{B_1} :- \sigma_1).\, B_1$	(by SR)
$\nabla(\overline{B_1} :- \sigma_1).\, u_1$	(by resolution)
u_1	(by cleanliness)

[Fazekas, Biere, Scholl '19] [Fazekas, Pollitt, Fleury, Biere '24]

Accumulated formulas are not your friend

The mutation operator $\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \models T$, or I otherwise.

$$I \models \nabla(T :- \sigma).C \quad \text{iff} \quad \nabla(T :- \sigma)(I) \models C$$ [RP, Suda ’18] [RP ’23]

F		F	F
$u_1 \vee F_1$	(by deletion)	$u_2 \vee F_2$	$u_3 \vee F_3$
$\nabla(\overline{B_1} :- \sigma_1).u_1 \vee F_1$	(by SR)	$\nabla(\overline{B_2} :- \sigma_2).u_2 \vee F_2$	$\nabla(\overline{B_2} :- \sigma_3).u_3 \vee F_3$
$\nabla(\overline{B_1} :- \sigma_1).B_1$	(by SR)	$\nabla(\overline{B_2} :- \sigma_2).B_2$	$\nabla(\overline{B_2} :- \sigma_3).B_3$
$\nabla(\overline{B_1} :- \sigma_1).u_1$	(by resolution)	$\nabla(\overline{B_2} :- \sigma_2).u_2$	$\nabla(\overline{B_2} :- \sigma_3).u_3$
u_1	(by cleanliness)	u_2	u_3

Accumulated formulas are not your friend

The mutation operator $\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \models T$, or I otherwise.

$$I \models \nabla(T :- \sigma).C \quad \text{iff} \quad \nabla(T :- \sigma)(I) \models C$$ [RP, Suda ’18] [RP ’23]

F		F	F
$u_1 \lor F_1$	(by deletion)	$u_2 \lor F_2$	$u_3 \lor F_3$
$\nabla(\overline{B_1} :- \sigma_1).\, u_1 \lor F_1$	(by SR)	$\nabla(\overline{B_2} :- \sigma_2).\, u_2 \lor F_2$	$\nabla(\overline{B_2} :- \sigma_3).\, u_3 \lor F_3$
$\nabla(\overline{B_1} :- \sigma_1).\, B_1$	(by SR)	$\nabla(\overline{B_2} :- \sigma_2).\, B_2$	$\nabla(\overline{B_2} :- \sigma_3).\, B_3$
$\nabla(\overline{B_1} :- \sigma_1).\, u_1$	(by resolution)	$\nabla(\overline{B_2} :- \sigma_2).\, u_2$	$\nabla(\overline{B_2} :- \sigma_3).\, u_3$
u_1	(by cleanliness)	u_2	u_3

$\overline{u_1} \lor \overline{u_2} \lor \overline{u_3}$ (by *some arcane magic*)

Accumulated formulas are not your friend

The mutation operator $\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \vDash T$, or I otherwise.

$I \vDash \nabla(T :- \sigma).C$ iff $\nabla(T :- \sigma)(I) \vDash C$ [RP, Suda ’18] [RP ’23]

F		F	F
$u_1 \vee F_1$	(by deletion)	$u_2 \vee F_2$	$u_3 \vee F_3$
$\nabla(\overline{B_1} :- \sigma_1).u_1 \vee F_1$	(by SR)	$\nabla(\overline{B_2} :- \sigma_2).u_2 \vee F_2$	$\nabla(\overline{B_2} :- \sigma_3).u_3 \vee F_3$
$\nabla(\overline{B_1} :- \sigma_1).B_1$	(by SR)	$\nabla(\overline{B_2} :- \sigma_2).B_2$	$\nabla(\overline{B_2} :- \sigma_3).B_3$
$\nabla(\overline{B_1} :- \sigma_1).u_1$	(by resolution)	$\nabla(\overline{B_2} :- \sigma_2).u_2$	$\nabla(\overline{B_2} :- \sigma_3).u_3$
u_1	(by cleanliness)	u_2	u_3

$\overline{u_1} \vee \overline{u_2} \vee \overline{u_3}$ (by *some arcane magic*)

$\bot$ (by resolution)

Accumulated formulas are not your friend

The mutation operator $\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \models T$, or I otherwise.

$$I \models \nabla(T :- \sigma).C \quad \text{iff} \quad \nabla(T :- \sigma)(I) \models C$$ [RP, Suda '18] [RP '23]

F		F	F
$u_1 \lor F_1$	(by deletion)	$u_2 \lor F_2$	$u_3 \lor F_3$
$\nabla(\overline{B_1} :- \sigma_1).u_1 \lor F_1$	(by SR)	$\nabla(\overline{B_2} :- \sigma_2).u_2 \lor F_2$	$\nabla(\overline{B_2} :- \sigma_3).u_3 \lor F_3$
$\nabla(\overline{B_1} :- \sigma_1).B_1$	(by SR)	$\nabla(\overline{B_2} :- \sigma_2).B_2$	$\nabla(\overline{B_2} :- \sigma_3).B_3$
$\nabla(\overline{B_1} :- \sigma_1).u_1$	(by resolution)	$\nabla(\overline{B_2} :- \sigma_2).u_2$	$\nabla(\overline{B_2} :- \sigma_3).u_3$
u_1	(by cleanliness)	u_2	u_3

$\overline{u_1} \lor \overline{u_2} \lor \overline{u_3}$ (by *some arcane magic*)

$\bot$ (by resolution)

why can't we just do this?

Accumulated formulas are not your friend

The mutation operator $\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \models T$, or I otherwise.

$I \models \nabla(T :- \sigma).C$ iff $\nabla(T :- \sigma)(I) \models C$ [RP, Suda '18] [RP '23]

F		F	F
$u_1 \vee F_1$	(by deletion)	$u_2 \vee F_2$	$u_3 \vee F_3$
$\nabla(\overline{B_1} :- \sigma_1).u_1 \vee F_1$	(by SR)	$\nabla(\overline{B_2} :- \sigma_2).u_2 \vee F_2$	$\nabla(\overline{B_2} :- \sigma_3).u_3 \vee F_3$
$\nabla(\overline{B_1} :- \sigma_1).B_1$	(by SR)	$\nabla(\overline{B_2} :- \sigma_2).B_2$	$\nabla(\overline{B_2} :- \sigma_3).B_3$
$\nabla(\overline{B_1} :- \sigma_1).u_1$	(by resolution)	$\nabla(\overline{B_2} :- \sigma_2).u_2$	$\nabla(\overline{B_2} :- \sigma_3).u_3$
u_1	(by cleanliness)	u_2	u_3

$\overline{u_1} \vee \overline{u_2} \vee \overline{u_3}$ this is a premise!

$\bot$ (by resolution)

why can't we just do this?

Compositionality is all you need: trimming

$$F = \{C_1, C_2, C_3\}$$

$$F$$

Compositionality is all you need: trimming

$F = \{C_1, C_2, C_3\}$

$$F \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, F \land B$$

Compositionality is all you need: trimming

$$F = \{C_1, C_2, C_3\}$$

$$F \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, F \wedge B$$

$$\{C_1, \overline{B}\} \vdash C_1\big|_\sigma$$

$$\{C_2, \overline{B}\} \vdash C_2\big|_\sigma$$

$$\{C_1, C_3, \overline{B}\} \vdash C_3\big|_\sigma$$

$$\{C_1, C_2, \overline{B}\} \vdash B|_\sigma$$

Compositionality is all you need: trimming

$$F = \{C_1, C_2, C_3\}$$

$$F \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, F \wedge B \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, \bot$$

$$\{C_1, \overline{B}\} \vdash C_1\big|_\sigma$$

$$\{C_2, \overline{B}\} \vdash C_2\big|_\sigma$$

$$\{C_1, C_3, \overline{B}\} \vdash C_3\big|_\sigma$$

$$\{C_1, C_2, \overline{B}\} \vdash B|_\sigma$$

Compositionality is all you need: trimming

$$F = \{C_1, C_2, C_3\}$$

$$F \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, F \land B \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, \bot$$

$$\{C_1, \overline{B}\} \vdash C_1\big|_\sigma$$

$$\{C_2, \overline{B}\} \vdash C_2\big|_\sigma$$

$$\{C_1, C_3, \overline{B}\} \vdash C_3\big|_\sigma$$

$$\{C_1, C_2, \overline{B}\} \vdash B|_\sigma$$

$$\{C_1, B\} \vdash \bot$$

Compositionality is all you need: trimming

$F = \{C_1, C_2, C_3\}$

$$F \vdash \nabla(\overline{B} :- \sigma).\, F \wedge B \vdash \nabla(\overline{B} :- \sigma).\, \bot$$

$\{C_1, \overline{B}\} \vdash C_1|_\sigma$ $\{C_1\} \vdash \nabla(\overline{B} :- \sigma).C_1$ $\{C_1, B\} \vdash \bot$

$\{C_2, \overline{B}\} \vdash C_2|_\sigma$

$\{C_1, C_3, \overline{B}\} \vdash C_3|_\sigma$

$\{C_1, C_2, \overline{B}\} \vdash B|_\sigma$

Compositionality is all you need: trimming

$F = \{C_1, C_2, C_3\}$

$$F \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, F \wedge B \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, \bot$$

$\{C_1, \overline{B}\} \vdash C_1|_\sigma$ $\quad \{C_1\} \vdash \nabla(\overline{B} :- \sigma).C_1$ $\quad \{C_1, B\} \vdash \bot$

$\{C_2, \overline{B}\} \vdash C_2|_\sigma$ $\quad \{C_2\} \vdash \nabla(\overline{B} :- \sigma).C_2$

$\{C_1, C_3, \overline{B}\} \vdash C_3|_\sigma$ $\quad \{C_1, C_3\} \vdash \nabla(\overline{B} :- \sigma).C_3$

$\{C_1, C_2, \overline{B}\} \vdash B|_\sigma$ $\quad \{C_1, C_2\} \vdash \nabla(\overline{B} :- \sigma).B$

Compositionality is all you need: trimming

$F = \{C_1, C_2, C_3\}$

$$F \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, F \wedge B \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, \bot$$

$\{C_1, \overline{B}\} \vdash C_1|_\sigma$ $\quad \{C_1\} \vdash \nabla(\overline{B} :- \sigma).C_1$ $\quad \{C_1, B\} \vdash \bot$

$\{C_2, \overline{B}\} \vdash C_2|_\sigma$ $\quad \{C_2\} \vdash \nabla(\overline{B} :- \sigma).C_2$

$\{C_1, C_3, \overline{B}\} \vdash C_3|_\sigma$ $\quad \{C_1, C_3\} \vdash \nabla(\overline{B} :- \sigma).C_3$

$\{C_1, C_2, \overline{B}\} \vdash B|_\sigma$ $\quad \{C_1, C_2\} \vdash \nabla(\overline{B} :- \sigma).B$

marked: $\nabla(\overline{B} :- \sigma).\bot$

[Heule, Hunt, Wetzler ’13]

Compositionality is all you need: trimming

$F = \{C_1, C_2, C_3\}$

$$F \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, F \wedge B \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, \bot$$

$\{C_1, \overline{B}\} \vdash C_1\big\|_\sigma$	$\{C_1\} \vdash \nabla(\overline{B} :- \sigma).C_1$	$\{C_1, B\} \vdash \bot$
$\{C_2, \overline{B}\} \vdash C_2\big\|_\sigma$	$\{C_2\} \vdash \nabla(\overline{B} :- \sigma).C_2$	
$\{C_1, C_3, \overline{B}\} \vdash C_3\big\|_\sigma$	$\{C_1, C_3\} \vdash \nabla(\overline{B} :- \sigma).C_3$	
$\{C_1, C_2, \overline{B}\} \vdash B\|_\sigma$	$\{C_1, C_2\} \vdash \nabla(\overline{B} :- \sigma).B$	

marked: $\nabla(\overline{B} :- \sigma).\bot$ $\nabla(\overline{B} :- \sigma).B$ $\nabla(\overline{B} :- \sigma).C_1$

Compositionality is all you need: trimming

$F = \{C_1, C_2, C_3\}$

$$F \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, F \wedge B \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, \bot$$

$\{C_1, \overline{B}\} \vdash C_1\big\|_\sigma$	$\{C_1\} \vdash \nabla(\overline{B} :- \sigma).C_1$	$\{C_1, B\} \vdash \bot$
$\{C_2, \overline{B}\} \vdash C_2\big\|_\sigma$	$\{C_2\} \vdash \nabla(\overline{B} :- \sigma).C_2$	
$\{C_1, C_3, \overline{B}\} \vdash C_3\big\|_\sigma$	$\{C_1, C_3\} \vdash \nabla(\overline{B} :- \sigma).C_3$	
$\{C_1, C_2, \overline{B}\} \vdash B\|_\sigma$	$\{C_1, C_2\} \vdash \nabla(\overline{B} :- \sigma).B$	

marked: $\nabla(\overline{B} :- \sigma).\bot$ $\nabla(\overline{B} :- \sigma).B$ $\nabla(\overline{B} :- \sigma).C_1$

Compositionality is all you need: trimming

$F = \{C_1, C_2, C_3\}$

$$F \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, F \land B \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, \bot$$

$\{C_1, \overline{B}\} \vdash C_1|_\sigma$ $\quad \{C_1\} \vdash \nabla(\overline{B} :- \sigma).C_1$ $\quad \{C_1, B\} \vdash \bot$

$\{C_2, \overline{B}\} \vdash C_2|_\sigma$ $\quad \{C_2\} \vdash \nabla(\overline{B} :- \sigma).C_2$

$\{C_1, C_3, \overline{B}\} \vdash C_3|_\sigma$ $\quad \{C_1, C_3\} \vdash \nabla(\overline{B} :- \sigma).C_3$

$\{C_1, C_2, \overline{B}\} \vdash B|_\sigma$ $\quad \{C_1, C_2\} \vdash \nabla(\overline{B} :- \sigma).B$

marked: $\nabla(\overline{B} :- \sigma).\bot$ $\quad \nabla(\overline{B} :- \sigma).B$ $\quad \nabla(\overline{B} :- \sigma).C_1$ $\quad C_1$

Compositionality is all you need: trimming

$$F = \{C_1, C_2, C_3\}$$

$$F \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, F \wedge B \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, \bot$$

$$\{C_1, \overline{B}\} \vdash C_1\big|_\sigma \qquad \{C_1\} \vdash \nabla(\overline{B} :- \sigma).C_1 \qquad \{C_1, B\} \vdash \bot$$

$$\{C_2, \overline{B}\} \vdash C_2\big|_\sigma \qquad \{C_2\} \vdash \nabla(\overline{B} :- \sigma).C_2$$

$$\{C_1, C_3, \overline{B}\} \vdash C_3\big|_\sigma \qquad \{C_1, C_3\} \vdash \nabla(\overline{B} :- \sigma).C_3$$

$$\{C_1, C_2, \overline{B}\} \vdash B|_\sigma \qquad \{C_1, C_2\} \vdash \nabla(\overline{B} :- \sigma).B$$

marked: $\nabla(\overline{B} :- \sigma).\bot \quad \nabla(\overline{B} :- \sigma).B \quad \nabla(\overline{B} :- \sigma).C_1 \quad C_1 \quad C_2$

$F = \{C_1, C_2, C_3\}$

$$F \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, F \wedge B \quad \vdash \quad \nabla(\overline{B} :- \sigma).\, \bot$$

$\{C_1, \overline{B}\} \vdash C_1\big\vert_\sigma$	$\{C_1\} \vdash \nabla(\overline{B} :- \sigma).C_1$	$\{C_1, B\} \vdash \bot$
$\{C_2, \overline{B}\} \vdash C_2\big\vert_\sigma$	$\{C_2\} \vdash \nabla(\overline{B} :- \sigma).C_2$	
$\{C_1, C_3, \overline{B}\} \vdash C_3\big\vert_\sigma$	$\{C_1, C_3\} \vdash \nabla(\overline{B} :- \sigma).C_3$	
$\{C_1, C_2, \overline{B}\} \vdash B\vert_\sigma$	$\{C_1, C_2\} \vdash \nabla(\overline{B} :- \sigma).B$	

marked: $\nabla(\overline{B} :- \sigma).\bot$ $\nabla(\overline{B} :- \sigma).B$ $\nabla(\overline{B} :- \sigma).C_1$ C_1 C_2 unsat core!

Compositionality is all you need: satisfiability

$$C_1: \quad x \lor y \lor z$$
$$C_2: \quad y \lor \overline{z}$$
$$C_3: \quad x \lor \overline{y} \lor z$$
$$C_4: \quad \overline{x} \lor \overline{y} \lor z$$

Compositionality is all you need: satisfiability

$C_1 : \quad x \lor y \lor z$

$C_2 : \quad y \lor \overline{z}$

$C_3 : \quad x \lor \overline{y} \lor z$

$C_4 : \quad \overline{x} \lor \overline{y} \lor z$

delete C_1 by $\sigma_1 = \{x \mapsto \top\}$

[Järvisalo, Heule, Biere ’12]

Compositionality is all you need: satisfiability

$C_1 : \quad x \vee y \vee z$

$C_2 : \quad y \vee \overline{z}$

$C_3 : \quad x \vee \overline{y} \vee z$

$C_4 : \quad \overline{x} \vee \overline{y} \vee z$

delete C_1 by $\sigma_1 = \{x \mapsto \top\}$

delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$

Compositionality is all you need: satisfiability

$C_1 : \quad x \vee y \vee z$

$C_2 : \quad y \vee \overline{z}$

$C_3 : \quad x \vee \overline{y} \vee z$

$C_4 : \quad \overline{x} \vee \overline{y} \vee z$

delete C_1 by $\sigma_1 = \{x \mapsto \top\}$

delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$

delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$

Compositionality is all you need: satisfiability

C_1 :	$x \vee y \vee z$	delete C_1 by $\sigma_1 = \{x \mapsto \top\}$
C_2 :	$y \vee \overline{z}$	delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$
C_3 :	$x \vee \overline{y} \vee z$	delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$
C_4 :	$\overline{x} \vee \overline{y} \vee z$	SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$

Compositionality is all you need: satisfiability

C_1 :	$x \vee y \vee z$	delete C_1 by $\sigma_1 = \{x \mapsto \top\}$
C_2 :	$y \vee \overline{z}$	delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$
C_3 :	$x \vee \overline{y} \vee z$	delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$
C_4 :	$\overline{x} \vee \overline{y} \vee z$	SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$

Redundant clause deletion **I can transform models of $F \setminus C$ into models of F**

[Järvisalo, Biere '10] [Järvisalo, Heule, Biere '12]

Compositionality is all you need: satisfiability

C_1 :	$x \vee y \vee z$	delete C_1 by $\sigma_1 = \{x \mapsto \top\}$
C_2 :	$y \vee \overline{z}$	delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$
C_3 :	$x \vee \overline{y} \vee z$	delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$
C_4 :	$\overline{x} \vee \overline{y} \vee z$	SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$

Redundant clause deletion I can transform models of $F \setminus C$ into models of F

[Järvisalo, Biere '10] [Järvisalo, Heule, Biere '12]

Satisfiability I can transform models of $\top$ into models of F

[Philipp, RP '16]

Compositionality is all you need: satisfiability

$C_1 : \quad x \vee y \vee z$ — delete C_1 by $\sigma_1 = \{x \mapsto \top\}$

$C_2 : \quad y \vee \overline{z}$ — delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$

$C_3 : \quad x \vee \overline{y} \vee z$ — delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$

$C_4 : \quad \overline{x} \vee \overline{y} \vee z$ — SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$

Redundant clause deletion I can transform models of $F \setminus C$ into models of F

[Järvisalo, Biere '10] [Järvisalo, Heule, Biere '12]

Satisfiability I can transform models of $\top$ into models of F

[Philipp, RP '16]

$$\varnothing \vdash \sigma_4 . C_4$$

Compositionality is all you need: satisfiability

C_1 :	$x \lor y \lor z$	delete C_1 by $\sigma_1 = \{x \mapsto \top\}$
C_2 :	$y \lor \overline{z}$	delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$
C_3 :	$x \lor \overline{y} \lor z$	delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$
C_4 :	$\overline{x} \lor \overline{y} \lor z$	SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$

Redundant clause deletion **I can transform models of $F \setminus C$ into models of F**

[Järvisalo, Biere '10] [Järvisalo, Heule, Biere '12]

Satisfiability **I can transform models of $\top$ into models of F**

[Philipp, RP '16]

$$\emptyset \vdash \sigma_4 . C_4$$
$$\vdash \sigma_4 \nabla (C_3 :- \sigma_3) . (C_3 \land C_4)$$

Compositionality is all you need: satisfiability

C_1 : $x \lor y \lor z$ — delete C_1 by $\sigma_1 = \{x \mapsto \top\}$

C_2 : $y \lor \overline{z}$ — delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$

C_3 : $x \lor \overline{y} \lor z$ — delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$

C_4 : $\overline{x} \lor \overline{y} \lor z$ — SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$

Redundant clause deletion I can transform models of $F \setminus C$ into models of F

[Järvisalo, Biere '10] [Järvisalo, Heule, Biere '12]

Satisfiability I can transform models of $\top$ into models of F

[Philipp, RP '16]

$$\varnothing \vdash \sigma_4 . C_4$$
$$\vdash \sigma_4 \nabla(C_3 :- \sigma_3).(C_3 \land C_4)$$
$$\vdash \sigma_4 \nabla(C_3 :- \sigma_3) \nabla(C_2 :- \sigma_2).(C_2 \land C_3 \land C_4)$$
$$\vdash \sigma_4 \nabla(C_3 :- \sigma_3) \nabla(C_2 :- \sigma_2) \nabla(C_1 :- \sigma_1).(C_1 \land C_2 \land C_3 \land C_4)$$

Compositionality is all you need: incremental solving

$C_1 : \quad x \vee y \vee z$ — delete C_1 by $\sigma_1 = \{x \mapsto \top\}$

$C_2 : \quad y \vee \overline{z}$ — delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$

$C_3 : \quad x \vee \overline{y} \vee z$ — delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$

$C_4 : \quad \overline{x} \vee \overline{y} \vee z$ — SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$

$$C_4 \vdash \nabla(C_3 :- \sigma_3)\, \nabla(C_2 :- \sigma_2)\, \nabla(C_1 :- \sigma_1).\, (C_1 \wedge C_2 \wedge C_3 \wedge C_4)$$

Compositionality is all you need: incremental solving

C_1 :	$x \vee y \vee z$	delete C_1 by $\sigma_1 = \{x \mapsto \top\}$
C_2 :	$y \vee \overline{z}$	delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$
C_3 :	$x \vee \overline{y} \vee z$	delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$
C_4 :	$\overline{x} \vee \overline{y} \vee z$	SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$
		insert $C_5 = \overline{y} \vee \overline{z}$ (clean on $\sigma_1, \sigma_2, \sigma_3$)

[Fazekas, Biere, Scholl '19]

[Fazekas, Pollitt, Fleury, Biere '24]

$$C_4 \vdash \nabla(C_3 :- \sigma_3)\, \nabla(C_2 :- \sigma_2)\, \nabla(C_1 :- \sigma_1).\, (C_1 \wedge C_2 \wedge C_3 \wedge C_4)$$

Compositionality is all you need: incremental solving

C_1 :	$x \vee y \vee z$	delete C_1 by $\sigma_1 = \{x \mapsto \top\}$
C_2 :	$y \vee \overline{z}$	delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$
C_3 :	$x \vee \overline{y} \vee z$	delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$
C_4 :	$\overline{x} \vee \overline{y} \vee z$	SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$
		insert $C_5 = \overline{y} \vee \overline{z}$ (clean on $\sigma_1, \sigma_2, \sigma_3$)

$$C_4 \vdash \nabla(C_3 :- \sigma_3)\,\nabla(C_2 :- \sigma_2)\,\nabla(C_1 :- \sigma_1).(C_1 \wedge C_2 \wedge C_3 \wedge C_4)$$

$$C_5 \vdash \nabla(C_3 :- \sigma_3)\,\nabla(C_2 :- \sigma_2)\,\nabla(C_1 :- \sigma_1).\,C_5$$

Compositionality is all you need: incremental solving

C_1 :	$x \vee y \vee z$	delete C_1 by $\sigma_1 = \{x \mapsto \top\}$
C_2 :	$y \vee \overline{z}$	delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$
C_3 :	$x \vee \overline{y} \vee z$	delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$
C_4 :	$\overline{x} \vee \overline{y} \vee z$	SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$
C_5 :	$\overline{y} \vee \overline{z}$	insert $C_5 = \overline{y} \vee \overline{z}$ (clean on $\sigma_1, \sigma_2, \sigma_3$)

$$C_4 \vdash \nabla(C_3 :- \sigma_3)\, \nabla(C_2 :- \sigma_2)\, \nabla(C_1 :- \sigma_1). (C_1 \wedge C_2 \wedge C_3 \wedge C_4)$$

$$C_5 \vdash \nabla(C_3 :- \sigma_3)\, \nabla(C_2 :- \sigma_2)\, \nabla(C_1 :- \sigma_1). C_5$$

Compositionality is all you need: incremental solving

$C_1 : \quad x \vee y \vee z$	delete C_1 by $\sigma_1 = \{x \mapsto \top\}$
$C_2 : \quad y \vee \overline{z}$	delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$
$C_3 : \quad x \vee \overline{y} \vee z$	delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$
$C_4 : \quad \overline{x} \vee \overline{y} \vee z$	SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$
$C_5 : \quad \overline{y} \vee \overline{z}$	insert $C_5 = \overline{y} \vee \overline{z}$ (clean on $\sigma_1, \sigma_2, \sigma_3$)
	SAT by $\sigma_5 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \bot\}$

$$C_4 \vdash \nabla(C_3 :- \sigma_3)\,\nabla(C_2 :- \sigma_2)\,\nabla(C_1 :- \sigma_1).(C_1 \wedge C_2 \wedge C_3 \wedge C_4)$$

$$C_5 \vdash \nabla(C_3 :- \sigma_3)\,\nabla(C_2 :- \sigma_2)\,\nabla(C_1 :- \sigma_1).C_5$$

Compositionality is all you need: incremental solving

C_1 :	$x \vee y \vee z$	delete C_1 by $\sigma_1 = \{x \mapsto \top\}$
C_2 :	$y \vee \overline{z}$	delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$
C_3 :	$x \vee \overline{y} \vee z$	delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$
C_4 :	$\overline{x} \vee \overline{y} \vee z$	SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$
C_5 :	$\overline{y} \vee \overline{z}$	insert $C_5 = \overline{y} \vee \overline{z}$ (clean on $\sigma_1, \sigma_2, \sigma_3$)
		SAT by $\sigma_5 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \bot\}$

$$C_4 \vdash \nabla(C_3 :- \sigma_3)\, \nabla(C_2 :- \sigma_2)\, \nabla(C_1 :- \sigma_1).\, (C_1 \wedge C_2 \wedge C_3 \wedge C_4)$$

$$C_5 \vdash \nabla(C_3 :- \sigma_3)\, \nabla(C_2 :- \sigma_2)\, \nabla(C_1 :- \sigma_1).\, C_5$$

$$\varnothing \vdash \sigma_5.\, (C_4 \wedge C_5)$$

Compositionality is all you need: incremental solving

$C_1 : \quad x \vee y \vee z$
$C_2 : \quad y \vee \overline{z}$
$C_3 : \quad x \vee \overline{y} \vee z$
$C_4 : \quad \overline{x} \vee \overline{y} \vee z$
$C_5 : \quad \overline{y} \vee \overline{z}$

delete C_1 by $\sigma_1 = \{x \mapsto \top\}$
delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$
delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$
SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$
insert $C_5 = \overline{y} \vee \overline{z}$ (clean on $\sigma_1, \sigma_2, \sigma_3$)
SAT by $\sigma_5 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \bot\}$
insert $C_6 = z$ (clean on σ_1, σ_3)

$$C_4 \vdash \nabla(C_3 :- \sigma_3)\, \nabla(C_2 :- \sigma_2)\, \nabla(C_1 :- \sigma_1). (C_1 \wedge C_2 \wedge C_3 \wedge C_4)$$

$$C_5 \vdash \nabla(C_3 :- \sigma_3)\, \nabla(C_2 :- \sigma_2)\, \nabla(C_1 :- \sigma_1). C_5$$

$$\varnothing \vdash \sigma_5. (C_4 \wedge C_5)$$

Compositionality is all you need: incremental solving

$C_1 : \quad x \vee y \vee z$

$C_2 : \quad y \vee \overline{z}$

$C_3 : \quad x \vee \overline{y} \vee z$

$C_4 : \quad \overline{x} \vee \overline{y} \vee z$

$C_5 : \quad \overline{y} \vee \overline{z}$

delete C_1 by $\sigma_1 = \{x \mapsto \top\}$

delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$

delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$

SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$

insert $C_5 = \overline{y} \vee \overline{z}$ (clean on $\sigma_1, \sigma_2, \sigma_3$)

SAT by $\sigma_5 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \bot\}$

insert $C_6 = z$ (clean on σ_1, σ_3)

$$C_4 \vdash \nabla(C_3 :- \sigma_3)\, \nabla(C_2 :- \sigma_2)\, \nabla(C_1 :- \sigma_1).\, (C_1 \wedge C_2 \wedge C_3 \wedge C_4)$$

$$C_5 \vdash \nabla(C_3 :- \sigma_3)\, \nabla(C_2 :- \sigma_2)\, \nabla(C_1 :- \sigma_1).\, C_5$$

$$\emptyset \vdash \sigma_5.\, (C_4 \wedge C_5)$$

$$C_6 \vdash \nabla(C_1 :- \sigma_1).\, C_6$$

Compositionality is all you need: incremental solving

C_1 :	$x \vee y \vee z$	delete C_1 by $\sigma_1 = \{x \mapsto \top\}$
C_2 :	$y \vee \overline{z}$	delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$
C_3 :	$x \vee \overline{y} \vee z$	delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$
C_4 :	$\overline{x} \vee \overline{y} \vee z$	SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$
C_5 :	$\overline{y} \vee \overline{z}$	insert $C_5 = \overline{y} \vee \overline{z}$ (clean on $\sigma_1, \sigma_2, \sigma_3$)
C_6 :	z	SAT by $\sigma_5 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \bot\}$
		insert $C_6 = z$ (clean on σ_1, σ_3)

$$C_4 \vdash \nabla(C_3 :- \sigma_3)\, \nabla(C_2 :- \sigma_2)\, \nabla(C_1 :- \sigma_1).\, (C_1 \wedge C_2 \wedge C_3 \wedge C_4)$$

$$C_5 \vdash \nabla(C_3 :- \sigma_3)\, \nabla(C_2 :- \sigma_2)\, \nabla(C_1 :- \sigma_1).\, C_5$$

$$\varnothing \vdash \sigma_5.\, (C_4 \wedge C_5)$$

$$C_6 \vdash \nabla(C_1 :- \sigma_1).\, C_6$$

Compositionality is all you need: incremental solving

$C_1 : \quad x \vee y \vee z$ — delete C_1 by $\sigma_1 = \{x \mapsto \top\}$

$C_2 : \quad y \vee \overline{z}$ — delete C_2 by $\sigma_2 = \{z \mapsto \bot\}$

$C_3 : \quad x \vee \overline{y} \vee z$ — delete C_3 by $\sigma_3 = \{y \mapsto \bot\}$

$C_4 : \quad \overline{x} \vee \overline{y} \vee z$ — SAT by $\sigma_4 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \top\}$

$C_5 : \quad \overline{y} \vee \overline{z}$ — insert $C_5 = \overline{y} \vee \overline{z}$ (clean on $\sigma_1, \sigma_2, \sigma_3$)

$C_6 : \quad z$ — SAT by $\sigma_5 = \{x \mapsto \top, y \mapsto \bot, z \mapsto \bot\}$

insert $C_6 = z$ (clean on σ_1, σ_3)

UNSAT

$$C_4 \vdash \nabla(C_3 :- \sigma_3)\, \nabla(C_2 :- \sigma_2)\, \nabla(C_1 :- \sigma_1).\,(C_1 \wedge C_2 \wedge C_3 \wedge C_4)$$

$$C_5 \vdash \nabla(C_3 :- \sigma_3)\, \nabla(C_2 :- \sigma_2)\, \nabla(C_1 :- \sigma_1).\, C_5$$

$$\varnothing \vdash \sigma_5.\,(C_4 \wedge C_5)$$

$$C_6 \vdash \nabla(C_1 :- \sigma_1).\, C_6$$

$$C_2 \wedge \cdots \wedge C_6 \vdash \bot$$

What’s in the box?

What about dominance?

this requires a *huge* detour through modal logic

[Fischer, Ladner ’79] [Babiani, Herzig, Troquard ’13]

TL;DR: ∇ is really a box modality in PDL, dominance corresponds to the Kleene star

What's in the box?

What about dominance?

this requires a *huge* detour through modal logic

[Fischer, Ladner '79] [Babiani, Herzig, Troquard '13]

TL;DR: ∇ is really a box modality in PDL, dominance corresponds to the Kleene star

What about deletion in unsat proofs?

They now know their place (non-semantic performance annotations)

What's in the box?

What about dominance?

this requires a *huge* detour through modal logic

[Fischer, Ladner '79] [Babiani, Herzig, Troquard '13]

TL;DR: ∇ is really a box modality in PDL, dominance corresponds to the Kleene star

What about deletion in unsat proofs?

They now know their place (non-semantic performance annotations)

But how many rules do you need?

not that many: RUP can be (carefully) extended to (much of) PDL

What's in the box?

What about dominance?

this requires a *huge* detour through modal logic

[Fischer, Ladner '79] [Babiani, Herzig, Troquard '13]

TL;DR: ∇ is really a box modality in PDL, dominance corresponds to the Kleene star

What about deletion in unsat proofs?

They now know their place (non-semantic performance annotations)

But how many rules do you need?

not that many: RUP can be (carefully) extended to (much of) PDL

Wouldn't proofs be very long?

this is really a matter of format engineering

if done right, comparable to DRAT/VeriPB

What's in the box?

What about dominance?

this requires a *huge* detour through modal logic

[Fischer, Ladner '79] [Babiani, Herzig, Troquard '13]

TL;DR: ∇ is really a box modality in PDL, dominance corresponds to the Kleene star

What about deletion in unsat proofs?

They now know their place (non-semantic performance annotations)

But how many rules do you need?

not that many: RUP can be (carefully) extended to (much of) PDL

Wouldn't proofs be very long?

this is really a matter of format engineering

if done right, comparable to DRAT/VeriPB

Wouldn't the checks be too complex?

not if adequately restricted; distributed/parallelized checking is trivial

for RAT/SR-equivalent checks, same as DRAT/DSR

What's in the box?

What about dominance?

this requires a *huge* detour through modal logic

[Fischer, Ladner '79] [Babiani, Herzig, Troquard '13]

TL;DR: ∇ is really a box modality in PDL, dominance corresponds to the Kleene star

What about deletion in unsat proofs?

They now know their place (non-semantic performance annotations)

But how many rules do you need?

not that many: RUP can be (carefully) extended to (much of) PDL

Wouldn't proofs be very long?

this is really a matter of format engineering

if done right, comparable to DRAT/VeriPB

Wouldn't the checks be too complex?

not if adequately restricted; distributed/parallelized checking is trivial

for RAT/SR-equivalent checks, same as DRAT/DSR

Does this yield new redundance rules?

so many I stopped bothering giving them names

Kripke semantics for Kripkean stuff

$\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \models T$, or I otherwise.

Kripke semantics for Kripkean stuff

$\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \vDash T$, or I otherwise.

I maps variables to bits $\rightsquigarrow$ memory states

Kripke semantics for Kripkean stuff

$\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \vDash T$, or I otherwise.

I maps variables to bits $\rightsquigarrow$ memory states

$\nabla(T :- \sigma)$ transforms a memory state into a memory state $\rightsquigarrow$ programs

Kripke semantics for Kripkean stuff

$\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \vDash T$, or I otherwise.

I maps variables to bits $\rightsquigarrow$ memory states

$\nabla(T :- \sigma)$ transforms a memory state into a memory state $\rightsquigarrow$ programs

if we want to make this work for dominance, we must be even more general:

- programs may be partial maps (to allow while loops)
- programs may be non-deterministic (to encode preorders)

Kripke semantics for Kripkean stuff

$\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \vDash T$, or I otherwise.

I maps variables to bits $\rightsquigarrow$ memory states

$\nabla(T :- \sigma)$ transforms a memory state into a memory state $\rightsquigarrow$ programs

if we want to make this work for dominance, we must be even more general:

- programs may be partial maps (to allow while loops)
- programs may be non-deterministic (to encode preorders)

Constraints semantics given by a set of (satisfying) assignments

$$J \vDash C$$

Kripke semantics for Kripkean stuff

$\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \vDash T$, or I otherwise.

I maps variables to bits $\rightsquigarrow$ memory states

$\nabla(T :- \sigma)$ transforms a memory state into a memory state $\rightsquigarrow$ programs

if we want to make this work for dominance, we must be even more general:

- programs may be partial maps (to allow while loops)
- programs may be non-deterministic (to encode preorders)

Constraints semantics given by a set of (satisfying) assignments

Programs semantics given by a binary relation of (transitioning) assignments

$$J \vDash C \qquad\qquad I \otimes J \vDash \varepsilon$$

Kripke semantics for Kripkean stuff

$\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \vDash T$, or I otherwise.

I maps variables to bits $\rightsquigarrow$ memory states

$\nabla(T :- \sigma)$ transforms a memory state into a memory state $\rightsquigarrow$ programs

if we want to make this work for dominance, we must be even more general:

- programs may be partial maps (to allow while loops)
- programs may be non-deterministic (to encode preorders)

Constraints semantics given by a set of (satisfying) assignments

Programs semantics given by a binary relation of (transitioning) assignments

$I \vDash \varepsilon.C$ iff $J \vDash C$ for all J such that $I \otimes J \vDash \varepsilon$

Kripke semantics for Kripkean stuff

$\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \vDash T$, or I otherwise.

I maps variables to bits $\rightsquigarrow$ memory states

$\nabla(T :- \sigma)$ transforms a memory state into a memory state $\rightsquigarrow$ programs

if we want to make this work for dominance, we must be even more general:

- programs may be partial maps (to allow while loops)
- programs may be non-deterministic (to encode preorders)

Constraints semantics given by a set of (satisfying) assignments

Programs semantics given by a binary relation of (transitioning) assignments

$$I \vDash \varepsilon.C \quad \text{iff} \quad J \vDash C \text{ for all } J \text{ such that } I \otimes J \vDash \varepsilon$$

Theorem (necessitation) if $F \vDash G$ then $\varepsilon.F \vDash \varepsilon.G$

Kripke semantics for Kripkean stuff

$\nabla(T :- \sigma)(I)$ is $I \circ \sigma$ if $I \vDash T$, or I otherwise.

I maps variables to bits $\rightsquigarrow$ memory states

$\nabla(T :- \sigma)$ transforms a memory state into a memory state $\rightsquigarrow$ programs

if we want to make this work for dominance, we must be even more general:

- programs may be partial maps (to allow while loops)
- programs may be non-deterministic (to encode preorders)

Constraints semantics given by a set of (satisfying) assignments

Programs semantics given by a binary relation of (transitioning) assignments

$$I \vDash \varepsilon.C \quad \text{iff} \quad J \vDash C \text{ for all } J \text{ such that } I \otimes J \vDash \varepsilon$$

Theorem (necessitation) if $F \vDash G$ then $\varepsilon.F \vDash \varepsilon.G$

right out of the bat: parametric lemmas!

I heard you like programs

$\langle\sigma\rangle$ assignments (set/clear/swap/flip bits)

I heard you like programs

$\langle \sigma \rangle$	assignments (set/clear/swap/flip bits)
$\varepsilon_1 \ldots \varepsilon_n$	sequential composition

I heard you like programs

$\langle \sigma \rangle$	assignments (set/clear/swap/flip bits)
$\varepsilon_1 \ldots \varepsilon_n$	sequential composition
$\varepsilon_1 \sqcup \cdots \sqcup \varepsilon_n$	non-deterministic choice

I heard you like programs

$\langle \sigma \rangle$	assignments (set/clear/swap/flip bits)
$\varepsilon_1 \ldots \varepsilon_n$	sequential composition
$\varepsilon_1 \sqcup \cdots \sqcup \varepsilon_n$	non-deterministic choice
$T?$	assertion

I heard you like programs

$\langle \sigma \rangle$	assignments (set/clear/swap/flip bits)
$\varepsilon_1 \dots \varepsilon_n$	sequential composition
$\varepsilon_1 \sqcup \cdots \sqcup \varepsilon_n$	non-deterministic choice
$T?$	assertion
ε^*	non-deterministic repetition

I heard you like programs

$\langle \sigma \rangle$	assignments (set/clear/swap/flip bits)
$\varepsilon_1 \dots \varepsilon_n$	sequential composition
$\varepsilon_1 \sqcup \cdots \sqcup \varepsilon_n$	non-deterministic choice
$T?$	assertion
ε^*	non-deterministic repetition
$\Diamond(V : \varepsilon_1 \parallel \varepsilon_0)$	concurrency

I heard you like programs

$\langle \sigma \rangle$	assignments (set/clear/swap/flip bits)
$\varepsilon_1 \ldots \varepsilon_n$	sequential composition
$\varepsilon_1 \sqcup \cdots \sqcup \varepsilon_n$	non-deterministic choice
$T?$	assertion
ε^*	non-deterministic repetition
$\lozenge(V : \varepsilon_1 \parallel \varepsilon_0)$	concurrency
$[R]$	run a SAT solver

I heard you like programs

$\langle\sigma\rangle$	assignments (set/clear/swap/flip bits)
$\varepsilon_1 \ldots \varepsilon_n$	sequential composition
$\varepsilon_1 \sqcup \cdots \sqcup \varepsilon_n$	non-deterministic choice
$T?$	assertion
ε^*	non-deterministic repetition
$\Diamond(V : \varepsilon_1 \parallel \varepsilon_0)$	concurrency
$[R]$	run a SAT solver

Constructing new programs

$\nabla(T : \varepsilon_1 \parallel \varepsilon_0) = (T?\,\varepsilon_1) \sqcup (\overline{T}?\,\varepsilon_0)$ (branching)

I heard you like programs

$\langle \sigma \rangle$	assignments (set/clear/swap/flip bits)
$\varepsilon_1 \dots \varepsilon_n$	sequential composition
$\varepsilon_1 \sqcup \cdots \sqcup \varepsilon_n$	non-deterministic choice
$T?$	assertion
ε^*	non-deterministic repetition
$\Diamond(V : \varepsilon_1 \parallel \varepsilon_0)$	concurrency
$[R]$	run a SAT solver

Constructing new programs

$$\nabla(T : \varepsilon_1 \parallel \varepsilon_0) = (T?\,\varepsilon_1) \sqcup (\overline{T}?\,\varepsilon_0) \qquad \text{(branching)}$$

$$\Box(T : \varepsilon) = (\overline{T}?\varepsilon)^*\,T? \qquad \text{(while loops)}$$

I heard you like programs

$\langle\sigma\rangle$	assignments (set/clear/swap/flip bits)
$\varepsilon_1 \ldots \varepsilon_n$	sequential composition
$\varepsilon_1 \sqcup \cdots \sqcup \varepsilon_n$	non-deterministic choice
$T?$	assertion
ε^*	non-deterministic repetition
$\lozenge(V : \varepsilon_1 \parallel \varepsilon_0)$	concurrency
$[R]$	run a SAT solver

Constructing new programs

$\nabla(T : \varepsilon_1 \parallel \varepsilon_0) = (T?\,\varepsilon_1) \sqcup (\overline{T}?\,\varepsilon_0)$ (branching)

$\square(T : \varepsilon) = (\overline{T}?\varepsilon)^*\,T?$ (while loops)

$\mathbf{0} = [\bot]$ (block)

I heard you like programs

$\langle\sigma\rangle$	assignments (set/clear/swap/flip bits)
$\varepsilon_1 \dots \varepsilon_n$	sequential composition
$\varepsilon_1 \sqcup \cdots \sqcup \varepsilon_n$	non-deterministic choice
$T?$	assertion
ε^*	non-deterministic repetition
$\Diamond(V : \varepsilon_1 \parallel \varepsilon_0)$	concurrency
$[R]$	run a SAT solver

Constructing new programs

$\nabla(T : \varepsilon_1 \parallel \varepsilon_0) = (T?\,\varepsilon_1) \sqcup (\overline{T}?\,\varepsilon_0)$ (branching)

$\Box(T : \varepsilon) = (\overline{T}?\varepsilon)^*\,T?$ (while loops)

$\mathbf{0} = [\bot]$ (block)

$\clubsuit = [\top]$ (nondet)

I heard you like programs

$\langle\sigma\rangle$	assignments (set/clear/swap/flip bits)
$\varepsilon_1 \dots \varepsilon_n$	sequential composition
$\varepsilon_1 \sqcup \cdots \sqcup \varepsilon_n$	non-deterministic choice
$T?$	assertion
ε^*	non-deterministic repetition
$\Diamond(V : \varepsilon_1 \parallel \varepsilon_0)$	concurrency
$[R]$	run a SAT solver

Constructing new programs

$\nabla(T : \varepsilon_1 \parallel \varepsilon_0) = (T?\,\varepsilon_1) \sqcup (\overline{T}?\,\varepsilon_0)$ (branching)

$\Box(T : \varepsilon) = (\overline{T}?\varepsilon)^*\, T?$ (while loops)

$\mathbf{0} = [\bot]$ (block)

$\clubsuit = [\top]$ (nondet)

$\forall V = \Diamond(V : \clubsuit \parallel 1)$ (universal quantification)

Dynamic proofs

Proving unsatisfiability F is unsatisfiable if $F \vdash \varepsilon.\bot$ and $\varepsilon.\bot \vdash \bot$

Dynamic proofs

Proving unsatisfiability F is unsatisfiable if $F \vdash \varepsilon.\bot$ and $\varepsilon.\bot \vdash \bot$

Proving satisfiability F is satisfiable if $\top \vdash \varepsilon.F$ and $\varepsilon.\bot \vdash \bot$

Dynamic proofs

Proving unsatisfiability F is unsatisfiable if $F \vdash \varepsilon.\bot$ and $\varepsilon.\bot \vdash \bot$

Proving satisfiability F is satisfiable if $\top \vdash \varepsilon.F$ and $\varepsilon.\bot \vdash \bot$

Proving a safety property P always holds assumming A if $A \vdash \varepsilon^*.P$

Dynamic proofs

Proving unsatisfiability F is unsatisfiable if $F \vdash \varepsilon.\bot$ and $\varepsilon.\bot \vdash \bot$

Proving satisfiability F is satisfiable if $\top \vdash \varepsilon.F$ and $\varepsilon.\bot \vdash \bot$

Proving a safety property P always holds assumming A if $A \vdash \varepsilon^*.P$

Proving a liveness property P eventually holds assumming A if $A \wedge \varepsilon^*.\overline{P} \vdash \bot$

Dynamic proofs

Proving unsatisfiability F is unsatisfiable if $F \vdash \varepsilon.\bot$ and $\varepsilon.\bot \vdash \bot$

Proving satisfiability F is satisfiable if $\top \vdash \varepsilon.F$ and $\varepsilon.\bot \vdash \bot$

Proving a safety property P always holds assumming A if $A \vdash \varepsilon^*.P$

Proving a liveness property P eventually holds assumming A if $A \wedge \varepsilon^*.\overline{P} \vdash \bot$

So where are we at the moment?

- An interference-free logical framework where trimming, distribution and incrementality work out of the box by design

Dynamic proofs

Proving unsatisfiability F is unsatisfiable if $F \vdash \varepsilon.\bot$ and $\varepsilon.\bot \vdash \bot$

Proving satisfiability F is satisfiable if $\top \vdash \varepsilon.F$ and $\varepsilon.\bot \vdash \bot$

Proving a safety property P always holds assumming A if $A \vdash \varepsilon^*.P$

Proving a liveness property P eventually holds assumming A if $A \wedge \varepsilon^*.\overline{P} \vdash \bot$

So where are we at the moment?

- An interference-free logical framework where trimming, distribution and incrementality work out of the box by design
- An interference-free, fully composable proof system with autoproving (of complexity similar to DSR) with assignment, choice and test, covering all of DRAT/DPR/DSR/WSR (SYNASC 2025)

Dynamic proofs

Proving unsatisfiability F is unsatisfiable if $F \vdash \varepsilon.\bot$ and $\varepsilon.\bot \vdash \bot$

Proving satisfiability F is satisfiable if $\top \vdash \varepsilon.F$ and $\varepsilon.\bot \vdash \bot$

Proving a safety property P always holds assumming A if $A \vdash \varepsilon^*.P$

Proving a liveness property P eventually holds assumming A if $A \wedge \varepsilon^*.\overline{P} \vdash \bot$

So where are we at the moment?

- An interference-free logical framework where trimming, distribution and incrementality work out of the box by design
- An interference-free, fully composable proof system with autoproving (of complexity similar to DSR) with assignment, choice and test, covering all of DRAT/DPR/DSR/WSR (SYNASC 2025)
- Proof rules to handle VeriPB-like dominance without interference or accumulated formulas; autoproving is only partially possible (but includes the VeriPB case)

Dynamic proofs

Proving unsatisfiability F is unsatisfiable if $F \vdash \varepsilon.\bot$ and $\varepsilon.\bot \vdash \bot$

Proving satisfiability F is satisfiable if $\top \vdash \varepsilon.F$ and $\varepsilon.\bot \vdash \bot$

Proving a safety property P always holds assumming A if $A \vdash \varepsilon^*.P$

Proving a liveness property P eventually holds assumming A if $A \wedge \varepsilon^*.\overline{P} \vdash \bot$

So where are we at the moment?

- An interference-free logical framework where trimming, distribution and incrementality work out of the box by design
- An interference-free, fully composable proof system with autoproving (of complexity similar to DSR) with assignment, choice and test, covering all of DRAT/DPR/DSR/WSR (SYNASC 2025)
- Proof rules to handle VeriPB-like dominance without interference or accumulated formulas; autoproving is only partially possible (but includes the VeriPB case)
- Still ironing some kinks out for dominance with full generality, beyond VeriPB-like dominance

Dynamic proofs

Proving unsatisfiability F is unsatisfiable if $F \vdash \varepsilon.\bot$ and $\varepsilon.\bot \vdash \bot$

Proving satisfiability F is satisfiable if $\top \vdash \varepsilon.F$ and $\varepsilon.\bot \vdash \bot$

Proving a safety property P always holds assumming A if $A \vdash \varepsilon^*.P$

Proving a liveness property P eventually holds assumming A if $A \wedge \varepsilon^*.\overline{P} \vdash \bot$

So where are we at the moment?

- An interference-free logical framework where trimming, distribution and incrementality work out of the box by design
- An interference-free, fully composable proof system with autoproving (of complexity similar to DSR) with assignment, choice and test, covering all of DRAT/DPR/DSR/WSR (SYNASC 2025)
- Proof rules to handle VeriPB-like dominance without interference or accumulated formulas; autoproving is only partially possible (but includes the VeriPB case)
- Still ironing some kinks out for dominance with full generality, beyond VeriPB-like dominance
- Nothing implemented yet!