# Look Ma! No Interference!

Adrián Rebola-Pardo

TU Wien, JKU Linz

Linz, Austria
15 October 2025

Supported by FWF 10.55776/COE12

# Proofs in SAT solving: the good ol' times

**Resolution** [Davis, Putnam '60]

$$\frac{C \vee x \qquad\qquad D \vee \overline{x}}{C \vee D} \qquad\qquad \{C \vee x, D \vee \overline{x}\} \vDash C \vee D$$

# Proofs in SAT solving: the good ol' times

**Resolution** [Davis, Putnam '60]

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D} \qquad \{C \vee x, D \vee \overline{x}\} \vDash C \vee D$$

$\pi = C_1, \ldots, C_n$ is a proof of $F \vDash G$ if

- $C_i$ is a resolvent of clauses in $F \cup \{C_1, \ldots, C_{i-1}\}$
- $G \subseteq F \cup \{C_1, \ldots, C_n\}$

# Proofs in SAT solving: the good ol' times

**Resolution** [Davis, Putnam '60]

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D} \qquad \{C \vee x, D \vee \overline{x}\} \vDash C \vee D$$

$\pi = C_1, \ldots, C_n$ is a proof of $F \vDash G$ if

- $C_i$ is a resolvent of clauses in $F \cup \{C_1, \ldots, C_{i-1}\}$
- $G \subseteq F \cup \{C_1, \ldots, C_n\}$

**Reverse unit propagation (RUP)** [Goldberg, Novikov '03]

$C$ is a RUP over $F$ if $C$ can be obtained by iterated resolution over clauses in $F$

# Proofs in SAT solving: the good ol' times

## Resolution [Davis, Putnam '60]

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D} \qquad \{C \vee x, D \vee \overline{x}\} \vDash C \vee D$$

$\pi = C_1, \ldots, C_n$ is a proof of $F \vDash G$ if

- $C_i$ is a resolvent of clauses in $F \cup \{C_1, \ldots, C_{i-1}\}$
- $G \subseteq F \cup \{C_1, \ldots, C_n\}$

## Reverse unit propagation (RUP) [Goldberg, Novikov '03]

$C$ is a RUP over $F$ if $C$ can be obtained by iterated resolution over clauses in $F$

$\pi = C_1, \ldots, C_n$ is a proof of $F \vDash G$ if

- $C_i$ is a RUP over $F \cup \{C_1, \ldots, C_{i-1}\}$
- $G \subseteq F \cup \{C_1, \ldots, C_n\}$

# Proofs in SAT solving: the good ol' times

## Resolution [Davis, Putnam '60]

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D} \qquad \{C \vee x, D \vee \overline{x}\} \vDash C \vee D$$

$\pi = C_1, \dots, C_n$ is a proof of $F \vDash G$ if

- $C_i$ is a resolvent of clauses in $F \cup \{C_1, \dots, C_{i-1}\}$
- $G \subseteq F \cup \{C_1, \dots, C_n\}$

## Reverse unit propagation (RUP) [Goldberg, Novikov '03]

$C$ is a RUP over $F$ if $C$ can be obtained by iterated resolution over clauses in $F$

$\pi = C_1, \dots, C_n$ is a proof of $F \vDash G$ if

- $C_i$ is a RUP over $F \cup \{C_1, \dots, C_{i-1}\}$
- $G \subseteq F \cup \{C_1, \dots, C_n\}$

## Deletions [Heule, Hunt, Wetzler '14]

Proofs are sequences of either introductions i: $C$ or deletions d: $C$.

At every point in the proof we have an accumulated formula $F_i$.

# Proofs in SAT solving: the good ol' times

## Resolution [Davis, Putnam '60]

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D} \qquad \{C \vee x, D \vee \overline{x}\} \vDash C \vee D$$

$\pi = C_1, \dots, C_n$ is a proof of $F \vDash G$ if

- $C_i$ is a resolvent of clauses in $F \cup \{C_1, \dots, C_{i-1}\}$
- $G \subseteq F \cup \{C_1, \dots, C_n\}$

## Reverse unit propagation (RUP) [Goldberg, Novikov '03]

$C$ is a RUP over $F$ if $C$ can be obtained by iterated resolution over clauses in $F$

$\pi = C_1, \dots, C_n$ is a proof of $F \vDash G$ if

- $C_i$ is a RUP over $F \cup \{C_1, \dots, C_{i-1}\}$
- $G \subseteq F \cup \{C_1, \dots, C_n\}$

## Deletions [Heule, Hunt, Wetzler '14]

Proofs are sequences of either introductions i: $C$ or deletions d: $C$.

At every point in the proof we have an accumulated formula $F_i$.

- If the $i$-th instruction is i: $C$, then $C$ must be RUP in $F_{i-1}$, and $F_i = F_{i-1} \cup \{C\}$

# Proofs in SAT solving: the good ol' times

## Resolution [Davis, Putnam '60]

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D} \qquad \{C \vee x, D \vee \overline{x}\} \vDash C \vee D$$

$\pi = C_1, \ldots, C_n$ is a proof of $F \vDash G$ if

- $C_i$ is a resolvent of clauses in $F \cup \{C_1, \ldots, C_{i-1}\}$
- $G \subseteq F \cup \{C_1, \ldots, C_n\}$

## Reverse unit propagation (RUP) [Goldberg, Novikov '03]

$C$ is a RUP over $F$ if $C$ can be obtained by iterated resolution over clauses in $F$

$\pi = C_1, \ldots, C_n$ is a proof of $F \vDash G$ if

- $C_i$ is a RUP over $F \cup \{C_1, \ldots, C_{i-1}\}$
- $G \subseteq F \cup \{C_1, \ldots, C_n\}$

## Deletions [Heule, Hunt, Wetzler '14]

Proofs are sequences of either introductions i: $C$ or deletions d: $C$.

At every point in the proof we have an accumulated formula $F_i$.

- If the $i$-th instruction is i: $C$, then $C$ must be RUP in $F_{i-1}$, and $F_i = F_{i-1} \cup \{C\}$
- If the $i$-th instruction is d: $C$, then $F_i = F_{i-1} \backslash \{C\}$

# Proofs in SAT solving: the redundance era

Proofs are sequences of either introductions i: $C$ or deletions d: $C$.

At every point in the proof we have an accumulated formula $F_i$.

- If the $i$-th instruction is i: $C$, then $C$ must be RUP in $F_{i-1}$, and $F_i = F_{i-1} \cup \{C\}$
- If the $i$-th instruction is d: $C$, then $F_i = F_{i-1} \setminus \{C\}$

# Proofs in SAT solving: the redundance era

Proofs are sequences of either introductions i: $C$ or deletions d: $C$.

At every point in the proof we have an accumulated formula $F_i$.

- If the $i$-th instruction is i: $C$, then $C$ must be RUP in $F_{i-1}$, and $F_i = F_{i-1} \cup \{C\}$
- If the $i$-th instruction is d: $C$, then $F_i = F_{i-1} \setminus \{C\}$

$$F = F_0 \vDash F_1 \vDash F_2 \vDash \cdots \vDash F_n \ni \bot$$

# Proofs in SAT solving: the redundance era

Proofs are sequences of either introductions i: $C$ or deletions d: $C$.

At every point in the proof we have an accumulated formula $F_i$.

- If the $i$-th instruction is i: $C$, then $C$ must be RUP in $F_{i-1}$, and $F_i = F_{i-1} \cup \{C\}$
- If the $i$-th instruction is d: $C$, then $F_i = F_{i-1} \setminus \{C\}$

$$F = F_0 \vDash F_1 \vDash F_2 \vDash \cdots \vDash F_n \ni \bot$$

... but why $\vDash$? [Järvisalo, Heule, Biere '12]

$C$ is redundant in $F$ if $F$ satisfiable implies $F \cup \{C\}$ satisfiable

If the proof ends in $\bot$, then we can replace RUP by any redundance criterion!

# Proofs in SAT solving: the redundance era

Proofs are sequences of either introductions i: $C$ or deletions d: $C$.

At every point in the proof we have an accumulated formula $F_i$.

- If the $i$-th instruction is i: $C$, then $C$ must be RUP in $F_{i-1}$, and $F_i = F_{i-1} \cup \{C\}$
- If the $i$-th instruction is d: $C$, then $F_i = F_{i-1} \setminus \{C\}$

$$F = F_0 \vDash F_1 \vDash F_2 \vDash \cdots \vDash F_n \ni \bot$$

... but why $\vDash$? [Järvisalo, Heule, Biere '12]

$C$ is redundant in $F$ if $F$ satisfiable implies $F \cup \{C\}$ satisfiable

If the proof ends in $\bot$, then we can replace RUP by any redundance criterion!

$$F = F_0 \vDash_{\text{sat}} F_1 \vDash_{\text{sat}} F_2 \vDash_{\text{sat}} \cdots \vDash_{\text{sat}} F_n \ni \bot$$

# Proofs in SAT solving: the redundance era

Proofs are sequences of either introductions i: $C$ or deletions d: $C$.

At every point in the proof we have an accumulated formula $F_i$.

- If the $i$-th instruction is i: $C$, then $C$ must be RUP in $F_{i-1}$, and $F_i = F_{i-1} \cup \{C\}$
- If the $i$-th instruction is d: $C$, then $F_i = F_{i-1} \setminus \{C\}$

$$F = F_0 \vDash F_1 \vDash F_2 \vDash \cdots \vDash F_n \ni \bot$$

... but why $\vDash$? [Järvisalo, Heule, Biere '12]

$C$ is redundant in $F$ if $F$ satisfiable implies $F \cup \{C\}$ satisfiable

If the proof ends in $\bot$, then we can replace RUP by any redundance criterion!

$$F = F_0 \vDash_{\text{sat}} F_1 \vDash_{\text{sat}} F_2 \vDash_{\text{sat}} \cdots \vDash_{\text{sat}} F_n \ni \bot$$

... why though? proof complexity considerations [Tseitin '83] [Haken '85]

# Proofs in SAT solving: the redundance era

Proofs are sequences of either introductions i: $C$ or deletions d: $C$.

At every point in the proof we have an accumulated formula $F_i$.

- If the $i$-th instruction is i: $C$, then $C$ must be RUP in $F_{i-1}$, and $F_i = F_{i-1} \cup \{C\}$
- If the $i$-th instruction is d: $C$, then $F_i = F_{i-1} \setminus \{C\}$

$$F = F_0 \vDash F_1 \vDash F_2 \vDash \cdots \vDash F_n \ni \bot$$

... but why $\vDash$? [Järvisalo, Heule, Biere '12]

$C$ is redundant in $F$ if $F$ satisfiable implies $F \cup \{C\}$ satisfiable

If the proof ends in $\bot$, then we can replace RUP by any redundance criterion!

$$F = F_0 \vDash_{\text{sat}} F_1 \vDash_{\text{sat}} F_2 \vDash_{\text{sat}} \cdots \vDash_{\text{sat}} F_n \ni \bot$$

... why though? proof complexity considerations [Tseitin '83] [Haken '85]

# Proofs in SAT solving: substitution redundancy and interference

**Substitution redundancy (SR)** [Buss, Thapen ’19] [Gocht, Nordström, ’21]

*but also DRAT [Wetzler, Heule, Hunt ’14], DPR [Heule, Kiesl, Biere ’17]*

$F$ is a CNF formula

$C$ is a clause

$\sigma$ is an atomic substitution (variables map to literals, ⊤ or ⊥)

# Proofs in SAT solving: substitution redundancy and interference

**Substitution redundancy (SR)** [Buss, Thapen '19] [Gocht, Nordström, '21]

*but also DRAT [Wetzler, Heule, Hunt '14], DPR [Heule, Kiesl, Biere '17]*

$F$ is a CNF formula

$C$ is a clause

$\sigma$ is an atomic substitution (variables map to literals, $\top$ or $\bot$)

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \vDash (F \cup \{C\})|_\sigma$ (simplified)

# Proofs in SAT solving: substitution redundancy and interference

**Substitution redundancy (SR)** [Buss, Thapen '19] [Gocht, Nordström, '21]

*but also DRAT [Wetzler, Heule, Hunt '14], DPR [Heule, Kiesl, Biere '17]*

$F$ is a CNF formula

$C$ is a clause

$\sigma$ is an atomic substitution (variables map to literals, $\top$ or $\bot$)

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \vDash (F \cup \{C\})|_\sigma$ (simplified)

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

# Proofs in SAT solving: substitution redundancy and interference

**Substitution redundancy (SR)** [Buss, Thapen '19] [Gocht, Nordström, '21]

*but also DRAT [Wetzler, Heule, Hunt '14], DPR [Heule, Kiesl, Biere '17]*

$F$ is a CNF formula

$C$ is a clause

$\sigma$ is an atomic substitution (variables map to literals, $\top$ or $\bot$)

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \vDash (F \cup \{C\})|_\sigma$ (simplified)

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

**Pros**

Succintly captures a lot (but not all!) of beyond-CDCL reasoning

# Proofs in SAT solving: substitution redundancy and interference

**Substitution redundancy (SR)** [Buss, Thapen '19] [Gocht, Nordström, '21]

*but also DRAT [Wetzler, Heule, Hunt '14], DPR [Heule, Kiesl, Biere '17]*

$F$ is a CNF formula

$C$ is a clause

$\sigma$ is an atomic substitution (variables map to literals, ⊤ or ⊥)

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \vDash (F \cup \{C\})|_\sigma$ (simplified)

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

**Pros**

Succintly captures a lot (but not all!) of beyond-CDCL reasoning

**Cons** [RP, Suda '17] [RP '23] [RP '25]

Fixpoints in trimming *huh, that's weird*

# Proofs in SAT solving: substitution redundancy and interference

## Substitution redundancy (SR) [Buss, Thapen '19] [Gocht, Nordström, '21]

*but also DRAT [Wetzler, Heule, Hunt '14], DPR [Heule, Kiesl, Biere '17]*

$F$ is a CNF formula

$C$ is a clause

$\sigma$ is an atomic substitution (variables map to literals, ⊤ or ⊥)

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \vDash (F \cup \{C\})|_\sigma$ (simplified)

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

## Pros

Succintly captures a lot (but not all!) of beyond-CDCL reasoning

## Cons [RP, Suda '17] [RP '23] [RP '25]

Fixpoints in trimming *huh, that's weird*

Subproofs cannot be merged *that's ok i guess?*

# Proofs in SAT solving: substitution redundancy and interference

**Substitution redundancy (SR)** [Buss, Thapen '19] [Gocht, Nordström, '21]

*but also DRAT [Wetzler, Heule, Hunt '14], DPR [Heule, Kiesl, Biere '17]*

$F$ is a CNF formula

$C$ is a clause

$\sigma$ is an atomic substitution (variables map to literals, ⊤ or ⊥)

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \vDash (F \cup \{C\})|_\sigma$ (simplified)

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

**Pros**

Succintly captures a lot (but not all!) of beyond-CDCL reasoning

**Cons** [RP, Suda '17] [RP '23] [RP '25]

Fixpoints in trimming *huh, that's weird*

Subproofs cannot be merged *that's ok i guess?*

Deriving lemmas prevents inferences *⋆slams desk in anger⋆*

# Proofs in SAT solving: substitution redundancy and interference

**Substitution redundancy (SR)** [Buss, Thapen '19] [Gocht, Nordström, '21]

*but also DRAT [Wetzler, Heule, Hunt '14], DPR [Heule, Kiesl, Biere '17]*

$F$ is a CNF formula

$C$ is a clause

$\sigma$ is an atomic substitution (variables map to literals, $\top$ or $\bot$)

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \models (F \cup \{C\})|_\sigma$ (simplified)

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

**Pros**

Succintly captures a lot (but not all!) of beyond-CDCL reasoning

**Cons** [RP, Suda '17] [RP '23] [RP '25]

Fixpoints in trimming *huh, that's weird*

Subproofs cannot be merged *that's ok i guess?*

Deriving lemmas prevents inferences *⋆slams desk in anger⋆*

Deletions are now semantically relevant *⋆yells at clouds⋆*

# Proofs in SAT solving: substitution redundancy and interference

**Substitution redundancy (SR)** [Buss, Thapen '19] [Gocht, Nordström, '21]

*but also DRAT [Wetzler, Heule, Hunt '14], DPR [Heule, Kiesl, Biere '17]*

$F$ is a CNF formula

$C$ is a clause

$\sigma$ is an atomic substitution (variables map to literals, $\top$ or $\bot$)

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \vDash (F \cup \{C\})|_\sigma$ (simplified)

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

**Pros**

Succinctly captures a lot (but not all!) of beyond-CDCL reasoning

**Cons** [RP, Suda '17] [RP '23] [RP '25]

Fixpoints in trimming *huh, that's weird*

Subproofs cannot be merged *that's ok i guess?*

Deriving lemmas prevents inferences *⋆slams desk in anger⋆*

Deletions are now semantically relevant *⋆yells at clouds⋆*

**The real issue** non-monotonicity and global dependence a.k.a. **interference**

# Accumulated formulas are not your friend

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \models (F \cup \{C\})|_\sigma$

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

# Accumulated formulas are not your friend

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \vDash (F \cup \{C\})|_{\sigma}$

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

If $\pi$ is a proof of $F \vdash G$ then, given any assignment $I$ satisfying $F$, I know how to construct $\mathrm{mut}(I)$ satisfying $G$

# Accumulated formulas are not your friend

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \vDash (F \cup \{C\})|_\sigma$

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

If $\pi$ is a proof of $F \vdash G$ then, given any assignment $I$ satisfying $F$, I know how to construct $\mathrm{mut}(I)$ satisfying $G$

This is a monotonic property! *⋆bangs head against the wall⋆*

Solution [RP, Suda '17] [RP '23]

Operate over constraints $\nabla(\sigma_1 :- T_1). \ldots \nabla(\sigma_n :- T_n).C$

# Accumulated formulas are not your friend

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \vDash (F \cup \{C\})|_\sigma$

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

If $\pi$ is a proof of $F \vdash G$ then, given any assignment $I$ satisfying $F$, I know how to construct $\mathrm{mut}(I)$ satisfying $G$

This is a monotonic property! *⋆bangs head against the wall⋆*

Solution [RP, Suda '17] [RP '23]

Operate over constraints $\nabla(\sigma_1 :- T_1). \dots \nabla(\sigma_n :- T_n).C$

$I \vDash \nabla(T :- \sigma).\varphi$ iff $I' \vDash \varphi$, where $I' = I \circ \sigma$ if $I \vDash T$, and $I' = I$ otherwise

# Accumulated formulas are not your friend

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \vDash (F \cup \{C\})|_\sigma$

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

If $\pi$ is a proof of $F \vdash G$ then, given any assignment $I$ satisfying $F$, I know how to construct $\mathrm{mut}(I)$ satisfying $G$

This is a monotonic property! *⋆bangs head against the wall⋆*

Solution [RP, Suda '17] [RP '23]

Operate over constraints $\nabla(\sigma_1 :- T_1) \ldots \nabla(\sigma_n :- T_n).C$

$I \vDash \nabla(T :- \sigma).\varphi$ iff $I' \vDash \varphi$, where $I' = I \circ \sigma$ if $I \vDash T$, and $I' = I$ otherwise

If $C$ is SR over $F$ upon $\sigma$ then $F \vDash \nabla(\overline{C} :- \sigma).F \cup \{C\}$

# Accumulated formulas are not your friend

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \vDash (F \cup \{C\})|_\sigma$

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

If $\pi$ is a proof of $F \vdash G$ then, given any assignment $I$ satisfying $F$, I know how to construct $\mathrm{mut}(I)$ satisfying $G$

This is a monotonic property! *⋆bangs head against the wall⋆*

Solution [RP, Suda '17] [RP '23]

Operate over constraints $\nabla(\sigma_1 :- T_1). \ldots \nabla(\sigma_n :- T_n).C$

$I \vDash \nabla(T :- \sigma).\varphi$ iff $I' \vDash \varphi$, where $I' = I \circ \sigma$ if $I \vDash T$, and $I' = I$ otherwise

If $C$ is SR over $F$ upon $\sigma$ then $F \vDash \nabla(\overline{C} :- \sigma).F \cup \{C\}$

Fixpoints in trimming are now gone!

# Accumulated formulas are not your friend

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \vDash (F \cup \{C\})|_\sigma$

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

If $\pi$ is a proof of $F \vdash G$ then, given any assignment $I$ satisfying $F$, I know how to construct $\mathrm{mut}(I)$ satisfying $G$

This is a monotonic property! *⋆bangs head against the wall⋆*

Solution [RP, Suda '17] [RP '23]

Operate over constraints $\nabla(\sigma_1 :- T_1). \ldots \nabla(\sigma_n :- T_n).C$

$I \vDash \nabla(T :- \sigma).\varphi$ iff $I' \vDash \varphi$, where $I' = I \circ \sigma$ if $I \vDash T$, and $I' = I$ otherwise

If $C$ is SR over $F$ upon $\sigma$ then $F \vDash \nabla(\overline{C} :- \sigma).F \cup \{C\}$

Fixpoints in trimming are now gone! Lemmas can now safely derived!

# Accumulated formulas are not your friend

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \models (F \cup \{C\})|_\sigma$

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

If $\pi$ is a proof of $F \vdash G$ then, given any assignment $I$ satisfying $F$, I know how to construct $\mathrm{mut}(I)$ satisfying $G$

This is a monotonic property! *⋆bangs head against the wall⋆*

Solution [RP, Suda '17] [RP '23]

Operate over constraints $\nabla(\sigma_1 :- T_1).\ldots\nabla(\sigma_n :- T_n).C$

$I \models \nabla(T :- \sigma).\varphi$ iff $I' \models \varphi$, where $I' = I \circ \sigma$ if $I \models T$, and $I' = I$ otherwise

If $C$ is SR over $F$ upon $\sigma$ then $F \models \nabla(\overline{C} :- \sigma).F \cup \{C\}$

Fixpoints in trimming are now gone! Lemmas can now safely derived! Deletions are now semantically irrelevant!

# Accumulated formulas are not your friend

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \vDash (F \cup \{C\})|_\sigma$

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

If $\pi$ is a proof of $F \vdash G$ then, given any assignment $I$ satisfying $F$, I know how to construct $\mathrm{mut}(I)$ satisfying $G$

This is a monotonic property! *⋆bangs head against the wall⋆*

Solution [RP, Suda '17] [RP '23]

Operate over constraints $\nabla(\sigma_1 :- T_1). \ldots \nabla(\sigma_n :- T_n).C$

$I \vDash \nabla(T :- \sigma).\varphi$ iff $I' \vDash \varphi$, where $I' = I \circ \sigma$ if $I \vDash T$, and $I' = I$ otherwise

If $C$ is SR over $F$ upon $\sigma$ then $F \vDash \nabla(\overline{C} :- \sigma).F \cup \{C\}$

Fixpoints in trimming are now gone! Lemmas can now safely derived! Deletions are now semantically irrelevant! All inferences have clearly defined dependencies!

# Accumulated formulas are not your friend

$C$ is SR over $F$ upon $\sigma$ if $F \cup \{\overline{C}\} \vDash (F \cup \{C\})|_\sigma$

If I had an assignment $I$ satisfying $F$, then check if it satisfies $C$.

If it does, then $I$ satisfies $F \cup \{C\}$; otherwise, $I \circ \sigma$ satisfies $F \cup \{C\}$.

If $\pi$ is a proof of $F \vdash G$ then, given any assignment $I$ satisfying $F$, I know how to construct $\mathrm{mut}(I)$ satisfying $G$

This is a monotonic property! *⋆bangs head against the wall⋆*

Solution [RP, Suda '17] [RP '23]

Operate over constraints $\nabla(\sigma_1 :- T_1). \ldots \nabla(\sigma_n :- T_n).C$

$I \vDash \nabla(T :- \sigma).\varphi$ iff $I' \vDash \varphi$, where $I' = I \circ \sigma$ if $I \vDash T$, and $I' = I$ otherwise

If $C$ is SR over $F$ upon $\sigma$ then $F \vDash \nabla(\overline{C} :- \sigma).F \cup \{C\}$

Fixpoints in trimming are now gone! Lemmas can now safely derived! Deletions are now semantically irrelevant! All inferences have clearly defined dependencies!

... but subproofs still can't be merged *I must be missing something...*

# Dominance: interference with a vengeance

Dominance needs three accumulated formulas *please stop...*

$K(x)$, $R(x)$, $O(x, x')$ are CNF formulas

# Dominance: interference with a vengeance

Dominance needs three accumulated formulas *please stop...*

$K(x), R(x), O(x, x')$ are CNF formulas

$O(x, x')$ encodes a preorder: $O(x, x') \equiv \sum_{i=0}^{n} 2^i x_i - \sum_{i=0}^{n} 2^i x'_i \leq 0$

$$F \overset{}{\underset{\kappa}{\rightsquigarrow}} K \underset{\delta}{\rightsquigarrow} K \cup R \qquad K(x) \vDash O(x, \delta(x))$$

RUP and SR add clauses in $R(x)$

SR now requires proving $K(x) \vDash O(x, \sigma(x))$, and $\sigma$ is added to $\delta$

# Dominance: interference with a vengeance

Dominance needs three accumulated formulas *please stop...*

$K(x), R(x), O(x, x')$ are CNF formulas

$O(x, x')$ encodes a preorder: $O(x, x') \equiv \sum_{i=0}^{n} 2^i x_i - \sum_{i=0}^{n} 2^i x'_i \leq 0$

$$F \underset{\kappa}{\rightsquigarrow} K \underset{\delta}{\rightsquigarrow} K \cup R \qquad K(x) \vDash O(x, \delta(x))$$

RUP and SR add clauses in $R(x)$

SR now requires proving $K(x) \vDash O(x, \sigma(x))$, and $\sigma$ is added to $\delta$

Dominance [Boggaerts, Gocht, McCreesh, Nordström '23]

$C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \vDash K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \vDash O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

# Dominance: interference with a vengeance

Dominance needs three accumulated formulas *please stop...*

$K(x), R(x), O(x, x')$ are CNF formulas

$O(x, x')$ encodes a preorder: $O(x, x') \equiv \sum_{i=0}^{n} 2^i x_i - \sum_{i=0}^{n} 2^i x'_i \leq 0$

$$F \underset{\kappa}{\rightsquigarrow} K \underset{\delta}{\rightsquigarrow} K \cup R \qquad K(x) \vDash O(x, \delta(x))$$

RUP and SR add clauses in $R(x)$

SR now requires proving $K(x) \vDash O(x, \sigma(x))$, and $\sigma$ is added to $\delta$

**Dominance** [Boggaerts, Gocht, McCreesh, Nordström ’23]

$C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \vDash K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \vDash O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

If is dominance-redundant over $K, R, O$ upon $\sigma$, then $C$ is redundant on $K \cup R$

# Look who's hiding under dominance

$K(x), R(x), O(x, x')$ are CNF formulas        $O$ encodes a preorder

$$F \overset{}{\underset{\kappa}{\rightsquigarrow}} K \underset{\delta}{\rightsquigarrow} K \cup R \qquad K(x) \vDash O(x, \delta(x))$$

**Dominance-based strengthening** *(very simplified)*   $C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \vDash K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \vDash O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

# Look who's hiding under dominance

$K(x), R(x), O(x, x')$ are CNF formulas $\qquad$ $O$ encodes a preorder

$$F \overset{}{\underset{\kappa}{\rightsquigarrow}} K \underset{\delta}{\rightsquigarrow} K \cup R \qquad K(x) \vDash O(x, \delta(x))$$

**Dominance-based strengthening** *(very simplified)* $\;C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \vDash K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \vDash O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

If I had an assignment $I_0$ satisfying $K$,

# Look who's hiding under dominance

$K(x), R(x), O(x, x')$ are CNF formulas $\qquad$ $O$ encodes a preorder

$$F \overset{}{\underset{\kappa}{\rightsquigarrow}} K \underset{\delta}{\rightsquigarrow} K \cup R \qquad K(x) \vDash O(x, \delta(x))$$

**Dominance-based strengthening** *(very simplified)* $\quad C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \vDash K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \vDash O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

If I had an assignment $I_0$ satisfying $K$,
then by $\delta$ I have an assignment $J_0$ satisfying $K \cup R$.

# Look who's hiding under dominance

$K(x), R(x), O(x,x')$ are CNF formulas

$O$ encodes a preorder

$$F \overset{}{\underset{\kappa}{\rightsquigarrow}} K \underset{\delta}{\rightsquigarrow} K \cup R \qquad K(x) \vDash O(x, \delta(x))$$

Dominance-based strengthening *(very simplified)* $C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \vDash K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \vDash O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

If I had an assignment $I_0$ satisfying $K$,
then by $\delta$ I have an assignment $J_0$ satisfying $K \cup R$.

If $J_0$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;

# Look who's hiding under dominance

$K(x), R(x), O(x, x')$ are CNF formulas $\qquad$ $O$ encodes a preorder

$$\begin{tikzcd} F \arrow[r, rightsquigarrow, "\kappa"'] & K \arrow[r, rightsquigarrow, "\delta"'] & K \cup R \arrow[l, rightsquigarrow, bend right, "\sigma"'] \end{tikzcd} \qquad K(x) \vDash O(x, \delta(x))$$

**Dominance-based strengthening** *(very simplified)* $C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \vDash K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \vDash O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

If I had an assignment $I_0$ satisfying $K$,
then by $\delta$ I have an assignment $J_0$ satisfying $K \cup R$.

If $J_0$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_1$ satisfying $K$,

# Look who's hiding under dominance

$K(x), R(x), O(x, x')$ are CNF formulas $\qquad$ $O$ encodes a preorder

$$\begin{tikzcd} F \arrow[r, squiggly, "\kappa"'] & K \arrow[r, squiggly, "\delta"'] & K \cup R \arrow[l, squiggly, bend right, "\sigma"'] \end{tikzcd} \qquad K(x) \models O(x, \delta(x))$$

**Dominance-based strengthening** *(very simplified)* $C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \models K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \models O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

If I had an assignment $I_0$ satisfying $K$, then by $\delta$ I have an assignment $J_0$ satisfying $K \cup R$.

If $J_0$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable; otherwise by $\sigma$ I have an assignment $I_1$ satisfying $K$, and by $\delta$ I have an assignment $J_1$ satisfying $K \cup R$.

# Look who's hiding under dominance

$K(x), R(x), O(x, x')$ are CNF formulas $\qquad$ $O$ encodes a preorder

$$\begin{tikzcd} F \arrow[r, squiggly, "\kappa"'] & K \arrow[r, squiggly, "\delta"'] & K \cup R \arrow[l, squiggly, bend right, "\sigma"'] \end{tikzcd} \qquad K(x) \vDash O(x, \delta(x))$$

**Dominance-based strengthening** *(very simplified)* $\;$ $C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \vDash K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \vDash O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

If I had an assignment $I_0$ satisfying $K$,
then by $\delta$ I have an assignment $J_0$ satisfying $K \cup R$.

If $J_0$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_1$ satisfying $K$,
and by $\delta$ I have an assignment $J_1$ satisfying $K \cup R$.

If $J_1$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;

# Look who's hiding under dominance

$K(x), R(x), O(x, x')$ are CNF formulas $\qquad$ $O$ encodes a preorder

$$\begin{tikzcd} F \arrow[r, squiggly, "\kappa"'] & K \arrow[r, squiggly, "\delta"'] & K \cup R \arrow[l, squiggly, bend right, "\sigma"'] \end{tikzcd} \qquad K(x) \vDash O(x, \delta(x))$$

Dominance-based strengthening *(very simplified)* $\;$ $C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \vDash K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \vDash O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

If I had an assignment $I_0$ satisfying $K$,
then by $\delta$ I have an assignment $J_0$ satisfying $K \cup R$.

If $J_0$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_1$ satisfying $K$,
and by $\delta$ I have an assignment $J_1$ satisfying $K \cup R$.

If $J_1$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_2$ blah blah blah.

# Look who's hiding under dominance

$K(x), R(x), O(x, x')$ are CNF formulas $\qquad$ $O$ encodes a preorder

$$\begin{tikzcd} F \arrow[r, squiggly, "\kappa"'] & K \arrow[r, squiggly, "\delta"'] & K \cup R \arrow[l, squiggly, bend right, "\sigma"'] \end{tikzcd} \qquad K(x) \vDash O(x, \delta(x))$$

Dominance-based strengthening *(very simplified)* $C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \vDash K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \vDash O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

If I had an assignment $I_0$ satisfying $K$,
then by $\delta$ I have an assignment $J_0$ satisfying $K \cup R$.

If $J_0$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_1$ satisfying $K$,
and by $\delta$ I have an assignment $J_1$ satisfying $K \cup R$.

If $J_1$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_2$ blah blah blah.

This process cannot continue forever because
$O$ is non-increasing on $\delta$ and strictly decreasing on $\sigma$.

# Look who's hiding under dominance

$K(x), R(x), O(x, x')$ are CNF formulas $\qquad$ $O$ encodes a preorder

$$\begin{tikzcd} F \arrow[r, squiggly, "\kappa"'] & K \arrow[r, squiggly, "\delta"'] & K \cup R \arrow[l, squiggly, bend right, "\sigma"'] \end{tikzcd} \qquad K(x) \vDash x \geq \delta(x)$$

**Dominance-based strengthening** *(very simplified)* $C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \vDash K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \vDash O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

If I had an assignment $I_0$ satisfying $K$,
then by $\delta$ I have an assignment $J_0$ satisfying $K \cup R$.

If $J_0$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_1$ satisfying $K$,
and by $\delta$ I have an assignment $J_1$ satisfying $K \cup R$.

If $J_1$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_2$ blah blah blah.

This process cannot continue forever because
$O$ is non-increasing on $\delta$ and strictly decreasing on $\sigma$.

# Look who's hiding under dominance

$K(x), R(x), O(x, x')$ are CNF formulas

$O$ encodes a preorder

$$\begin{tikzcd} F \arrow[r, squiggly, "\kappa"'] & K \arrow[r, squiggly, "\delta"'] & K \cup R \arrow[l, squiggly, bend right, "\sigma"'] \end{tikzcd}$$

$$K(x) \cup R(x) \cup \overline{C}(x) \vDash x > \sigma(x)$$
$$K(x) \vDash x \geq \delta(x)$$

**Dominance-based strengthening** *(very simplified)* $C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \vDash K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \vDash O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

If I had an assignment $I_0$ satisfying $K$,
then by $\delta$ I have an assignment $J_0$ satisfying $K \cup R$.

If $J_0$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_1$ satisfying $K$,
and by $\delta$ I have an assignment $J_1$ satisfying $K \cup R$.

If $J_1$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_2$ blah blah blah.

This process cannot continue forever because
$O$ is non-increasing on $\delta$ and strictly decreasing on $\sigma$.

# Look who's hiding under dominance

$K(x), R(x), O(x, x')$ are CNF formulas $\quad$ $O$ encodes a preorder

invariant

$$\begin{tikzcd} F \arrow[r, squiggly, "\kappa"'] & K \arrow[r, squiggly, "\delta"'] & K \cup R \arrow[l, squiggly, bend right, "\sigma"'] \end{tikzcd}$$

$K(x) \cup R(x) \cup \overline{C}(x) \vDash x > \sigma(x)$

$K(x) \vDash x \geq \delta(x)$

Dominance-based strengthening *(very simplified)* $\quad$ $C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \vDash K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \vDash O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

If I had an assignment $I_0$ satisfying $K$,
then by $\delta$ I have an assignment $J_0$ satisfying $K \cup R$.

If $J_0$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_1$ satisfying $K$,
and by $\delta$ I have an assignment $J_1$ satisfying $K \cup R$.

If $J_1$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_2$ blah blah blah.

This process cannot continue forever because
$O$ is non-increasing on $\delta$ and strictly decreasing on $\sigma$.

# Look who's hiding under dominance

$K(x), R(x), O(x, x')$ are CNF formulas $\qquad$ $O$ encodes a preorder

invariant

$$\begin{tikzcd} F \arrow[r, squiggly, "\kappa"'] & K \arrow[r, squiggly, "\delta"'] & K \cup R \arrow[l, squiggly, bend right, "\sigma"'] \end{tikzcd}$$

termination order

$$K(x) \cup R(x) \cup \overline{C}(x) \vDash x > \sigma(x)$$

$$K(x) \vDash x \geq \delta(x)$$

**Dominance-based strengthening** *(very simplified)* $C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \vDash K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \vDash O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

If I had an assignment $I_0$ satisfying $K$,
then by $\delta$ I have an assignment $J_0$ satisfying $K \cup R$.

If $J_0$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_1$ satisfying $K$,
and by $\delta$ I have an assignment $J_1$ satisfying $K \cup R$.

If $J_1$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_2$ blah blah blah.

This process cannot continue forever because
$O$ is non-increasing on $\delta$ and strictly decreasing on $\sigma$.

# Look who's hiding under dominance

$K(x), R(x), O(x, x')$ are CNF formulas $\qquad$ $O$ encodes a preorder


termination order

$K(x) \cup R(x) \cup \overline{C}(x) \vDash x > \sigma(x)$

$K(x) \vDash x \geq \delta(x)$

**Dominance-based strengthening** *(very simplified)* $\quad$ $C$ is dominance-redundant over $K, R, O$ upon $\sigma$ if:

- $K \cup R \cup \{\overline{C}\} \vDash K|_\sigma$
- $K(x) \cup R(x) \cup \{\overline{C(x)}\} \vDash O(x, \sigma(x)) \cup \overline{O(\sigma(x), x)}$

If I had an assignment $I_0$ satisfying $K$,
then by $\delta$ I have an assignment $J_0$ satisfying $K \cup R$.

If $J_0$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_1$ satisfying $K$,
and by $\delta$ I have an assignment $J_1$ satisfying $K \cup R$.

If $J_1$ satisfies $C$, then $K \cup R \cup \{C\}$ is satisfiable;
otherwise by $\sigma$ I have an assignment $I_2$ blah blah blah.

This process cannot continue forever because
$O$ is non-increasing on $\delta$ and strictly decreasing on $\sigma$.

# The swiss army knife of the non-non-monotonicist

Why do we keep having global conditions on classical reasoning when *we know* classical logic is monotonic? [Martin Suda over lunch in 2017]

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

# The swiss army knife of the non-non-monotonicist

Why do we keep having global conditions on classical reasoning when *we know* classical logic is monotonic? [Martin Suda over lunch in 2017]

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

$I$ maps variables to bits $\rightsquigarrow$ memory states

# The swiss army knife of the non-non-monotonicist

Why do we keep having global conditions on classical reasoning when *we know* classical logic is monotonic? [Martin Suda over lunch in 2017]

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

$I$ maps variables to bits $\rightsquigarrow$ memory states

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

# The swiss army knife of the non-non-monotonicist

Why do we keep having global conditions on classical reasoning when *we know* classical logic is monotonic? [Martin Suda over lunch in 2017]

$\nabla(T :- \sigma)(I)$ 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)

# The swiss army knife of the non-non-monotonicist

Why do we keep having global conditions on classical reasoning when *we know* classical logic is monotonic? [Martin Suda over lunch in 2017]

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

(Static) constraints semantics given by a set of (satisfying) assignments

$$J \models C$$

# The swiss army knife of the non-non-monotonicist

Why do we keep having global conditions on classical reasoning when *we know* classical logic is monotonic? [Martin Suda over lunch in 2017]

$\nabla(T :- \sigma)(I)$ 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)

(Static) constraints semantics given by a set of (satisfying) assignments

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

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

# The swiss army knife of the non-non-monotonicist

Why do we keep having global conditions on classical reasoning when *we know* classical logic is monotonic? [Martin Suda over lunch in 2017]

$\nabla(T :- \sigma)(I)$ 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)

(Static) constraints semantics given by a set of (satisfying) assignments

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

Dynamic constraints a static constraint must hold after executing a program

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

# The swiss army knife of the non-non-monotonicist

Why do we keep having global conditions on classical reasoning when *we know* classical logic is monotonic? [Martin Suda over lunch in 2017]

$\nabla(T :- \sigma)(I)$ 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)

(Static) constraints semantics given by a set of (satisfying) assignments

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

Dynamic constraints a static constraint must hold after executing a program

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

# What’s in the box? Programs!

# What's in the box? Programs!

**Noop** $I \otimes J \vDash \mathbf{1}$ iff $I = J$

# What's in the box? Programs!

**Noop** $I \otimes J \vDash \mathbf{1}$ iff $I = J$

**Composition** $I \otimes J \vDash \varepsilon_1\, \varepsilon_2$ iff $\exists K,\ I \otimes K \vDash \varepsilon_1$ and $K \otimes J \vDash \varepsilon_2$

# What's in the box? Programs!

**Noop** $I \otimes J \vDash 1$ iff $I = J$

**Composition** $I \otimes J \vDash \varepsilon_1 \, \varepsilon_2$ iff $\exists K,\ I \otimes K \vDash \varepsilon_1$ and $K \otimes J \vDash \varepsilon_2$

**Assignment** $I \otimes J \vDash \langle \sigma \rangle$ iff $J = I \circ \sigma$

# What's in the box? Programs!

**Noop** $I \otimes J \vDash 1$ iff $I = J$

**Composition** $I \otimes J \vDash \varepsilon_1\, \varepsilon_2$ iff $\exists K,\ I \otimes K \vDash \varepsilon_1$ and $K \otimes J \vDash \varepsilon_2$

**Assignment** $I \otimes J \vDash \langle \sigma \rangle$ iff $J = I \circ \sigma$

**Choice** $I \otimes J \vDash \varepsilon_1 \sqcup \varepsilon_2$ iff $I \otimes J \vDash \varepsilon_1$ or $I \otimes J \vDash \varepsilon_2$

# What's in the box? Programs!

**Noop** $I \otimes J \vDash 1$ iff $I = J$

**Composition** $I \otimes J \vDash \varepsilon_1\, \varepsilon_2$ iff $\exists K,\ I \otimes K \vDash \varepsilon_1$ and $K \otimes J \vDash \varepsilon_2$

**Assignment** $I \otimes J \vDash \langle\sigma\rangle$ iff $J = I \circ \sigma$

**Choice** $I \otimes J \vDash \varepsilon_1 \sqcup \varepsilon_2$ iff $I \otimes J \vDash \varepsilon_1$ or $I \otimes J \vDash \varepsilon_2$

**Test** $I \otimes J \vDash T?$ iff $I = J$ and $I \vDash T$

# What's in the box? Programs!

**Noop** $I \otimes J \vDash 1$ iff $I = J$

**Composition** $I \otimes J \vDash \varepsilon_1\, \varepsilon_2$ iff $\exists K,\ I \otimes K \vDash \varepsilon_1$ and $K \otimes J \vDash \varepsilon_2$

**Assignment** $I \otimes J \vDash \langle \sigma \rangle$ iff $J = I \circ \sigma$

**Choice** $I \otimes J \vDash \varepsilon_1 \sqcup \varepsilon_2$ iff $I \otimes J \vDash \varepsilon_1$ or $I \otimes J \vDash \varepsilon_2$

**Test** $I \otimes J \vDash T?$ iff $I = J$ and $I \vDash T$

**Branch** $I \otimes J \vDash \nabla(T : \varepsilon_1 \parallel \varepsilon_0)$ iff $I \otimes J \vDash (T?\, \varepsilon_1) \sqcup (\overline{T}?\, \varepsilon_0)$

# What's in the box? Programs!

**Noop** $I \otimes J \vDash 1$ iff $I = J$

**Composition** $I \otimes J \vDash \varepsilon_1\, \varepsilon_2$ iff $\exists K,\ I \otimes K \vDash \varepsilon_1$ and $K \otimes J \vDash \varepsilon_2$

**Assignment** $I \otimes J \vDash \langle \sigma \rangle$ iff $J = I \circ \sigma$

**Choice** $I \otimes J \vDash \varepsilon_1 \sqcup \varepsilon_2$ iff $I \otimes J \vDash \varepsilon_1$ or $I \otimes J \vDash \varepsilon_2$

**Test** $I \otimes J \vDash T?$ iff $I = J$ and $I \vDash T$

**Branch** $I \otimes J \vDash \nabla(T : \varepsilon_1 \parallel \varepsilon_0)$ iff $I \otimes J \vDash (T?\, \varepsilon_1) \sqcup (\overline{T}?\, \varepsilon_0)$

**Repeat** $I \otimes J \vDash \varepsilon^*$ iff $\exists I_i,\ I_0 = I,\ I_n = J,\ I_{i-1} \otimes I_i \vDash \varepsilon$

# What's in the box? Programs!

**Noop** $I \otimes J \vDash 1$ iff $I = J$

**Composition** $I \otimes J \vDash \varepsilon_1\, \varepsilon_2$ iff $\exists K,\ I \otimes K \vDash \varepsilon_1$ and $K \otimes J \vDash \varepsilon_2$

**Assignment** $I \otimes J \vDash \langle \sigma \rangle$ iff $J = I \circ \sigma$

**Choice** $I \otimes J \vDash \varepsilon_1 \sqcup \varepsilon_2$ iff $I \otimes J \vDash \varepsilon_1$ or $I \otimes J \vDash \varepsilon_2$

**Test** $I \otimes J \vDash T?$ iff $I = J$ and $I \vDash T$

**Branch** $I \otimes J \vDash \nabla(T : \varepsilon_1 \parallel \varepsilon_0)$ iff $I \otimes J \vDash (T?\, \varepsilon_1) \sqcup (\overline{T}?\, \varepsilon_0)$

**Repeat** $I \otimes J \vDash \varepsilon^*$ iff $\exists I_i,\ I_0 = I,\ I_n = J,\ I_{i-1} \otimes I_i \vDash \varepsilon$

**Loop** $I \otimes J \vDash \square(T : \varepsilon)$ iff $I \otimes J \vDash (\overline{T}?\varepsilon)^*\, T?$

# What’s in the box? Programs!

**Noop** $I \otimes J \vDash 1$ iff $I = J$

**Composition** $I \otimes J \vDash \varepsilon_1\, \varepsilon_2$ iff $\exists K,\ I \otimes K \vDash \varepsilon_1$ and $K \otimes J \vDash \varepsilon_2$

**Assignment** $I \otimes J \vDash \langle \sigma \rangle$ iff $J = I \circ \sigma$

**Choice** $I \otimes J \vDash \varepsilon_1 \sqcup \varepsilon_2$ iff $I \otimes J \vDash \varepsilon_1$ or $I \otimes J \vDash \varepsilon_2$

**Test** $I \otimes J \vDash T?$ iff $I = J$ and $I \vDash T$

**Branch** $I \otimes J \vDash \nabla(T : \varepsilon_1 \parallel \varepsilon_0)$ iff $I \otimes J \vDash (T?\, \varepsilon_1) \sqcup (\overline{T}?\, \varepsilon_0)$

**Repeat** $I \otimes J \vDash \varepsilon^*$ iff $\exists I_i,\ I_0 = I,\ I_n = J,\ I_{i-1} \otimes I_i \vDash \varepsilon$

**Loop** $I \otimes J \vDash \Box(T : \varepsilon)$ iff $I \otimes J \vDash (\overline{T}?\varepsilon)^*\, T?$

**Rendezvous** $I \otimes J \vDash \Diamond(V : \varepsilon_1 \parallel \varepsilon_0)$ iff $\exists J_1, J_0,\ I \otimes J_i \vDash \varepsilon_i$ and $J = J_1 +_V J_0$

# What's in the box? Programs!

**Noop** $I \otimes J \vDash 1$ iff $I = J$

**Composition** $I \otimes J \vDash \varepsilon_1\, \varepsilon_2$ iff $\exists K,\ I \otimes K \vDash \varepsilon_1$ and $K \otimes J \vDash \varepsilon_2$

**Assignment** $I \otimes J \vDash \langle \sigma \rangle$ iff $J = I \circ \sigma$

**Choice** $I \otimes J \vDash \varepsilon_1 \sqcup \varepsilon_2$ iff $I \otimes J \vDash \varepsilon_1$ or $I \otimes J \vDash \varepsilon_2$

**Test** $I \otimes J \vDash T?$ iff $I = J$ and $I \vDash T$

**Branch** $I \otimes J \vDash \nabla(T : \varepsilon_1 \parallel \varepsilon_0)$ iff $I \otimes J \vDash (T?\, \varepsilon_1) \sqcup (\overline{T}?\, \varepsilon_0)$

**Repeat** $I \otimes J \vDash \varepsilon^*$ iff $\exists I_i,\ I_0 = I,\ I_n = J,\ I_{i-1} \otimes I_i \vDash \varepsilon$

**Loop** $I \otimes J \vDash \Box(T : \varepsilon)$ iff $I \otimes J \vDash (\overline{T}?\varepsilon)^*\, T?$

**Rendezvous** $I \otimes J \vDash \Diamond(V : \varepsilon_1 \parallel \varepsilon_0)$ iff $\exists J_1, J_0,\ I \otimes J_i \vDash \varepsilon_i$ and $J = J_1 +_V J_0$

**Solve** $I \otimes J \vDash [R]$ iff $I \otimes J \vDash R$

# What's in the box? Programs!

**Noop** $I \otimes J \models 1$ iff $I = J$

**Composition** $I \otimes J \models \varepsilon_1\, \varepsilon_2$ iff $\exists K,\ I \otimes K \models \varepsilon_1$ and $K \otimes J \models \varepsilon_2$

**Assignment** $I \otimes J \models \langle \sigma \rangle$ iff $J = I \circ \sigma$

**Choice** $I \otimes J \models \varepsilon_1 \sqcup \varepsilon_2$ iff $I \otimes J \models \varepsilon_1$ or $I \otimes J \models \varepsilon_2$

**Test** $I \otimes J \models T?$ iff $I = J$ and $I \models T$

**Branch** $I \otimes J \models \nabla(T : \varepsilon_1 \parallel \varepsilon_0)$ iff $I \otimes J \models (T?\, \varepsilon_1) \sqcup (\overline{T}?\, \varepsilon_0)$

**Repeat** $I \otimes J \models \varepsilon^*$ iff $\exists I_i,\ I_0 = I,\ I_n = J,\ I_{i-1} \otimes I_i \models \varepsilon$

**Loop** $I \otimes J \models \square(T : \varepsilon)$ iff $I \otimes J \models (\overline{T}?\varepsilon)^*\, T?$

**Rendezvous** $I \otimes J \models \lozenge(V : \varepsilon_1 \parallel \varepsilon_0)$ iff $\exists J_1, J_0,\ I \otimes J_i \models \varepsilon_i$ and $J = J_1 +_V J_0$

**Solve** $I \otimes J \models [R]$ iff $I \otimes J \models R$

**Havoc** $I \otimes J \models \forall V$ iff $I \otimes J \models \lozenge(V : [\top] \parallel 1)$

# What's in the box? Programs!

Noop $I \otimes J \vDash 1$ iff $I = J$

Composition $I \otimes J \vDash \varepsilon_1\, \varepsilon_2$ iff $\exists K,\ I \otimes K \vDash \varepsilon_1$ and $K \otimes J \vDash \varepsilon_2$

Assignment $I \otimes J \vDash \langle \sigma \rangle$ iff $J = I \circ \sigma$

Choice $I \otimes J \vDash \varepsilon_1 \sqcup \varepsilon_2$ iff $I \otimes J \vDash \varepsilon_1$ or $I \otimes J \vDash \varepsilon_2$

Test $I \otimes J \vDash T?$ iff $I = J$ and $I \vDash T$

Branch $I \otimes J \vDash \nabla(T : \varepsilon_1 \parallel \varepsilon_0)$ iff $I \otimes J \vDash (T?\, \varepsilon_1) \sqcup (\overline{T}?\, \varepsilon_0)$

Repeat $I \otimes J \vDash \varepsilon^*$ iff $\exists I_i,\ I_0 = I,\ I_n = J,\ I_{i-1} \otimes I_i \vDash \varepsilon$

Loop $I \otimes J \vDash \square(T : \varepsilon)$ iff $I \otimes J \vDash (\overline{T}?\varepsilon)^*\, T?$

Rendezvous $I \otimes J \vDash \Diamond(V : \varepsilon_1 \parallel \varepsilon_0)$ iff $\exists J_1, J_0,\ I \otimes J_i \vDash \varepsilon_i$ and $J = J_1 +_V J_0$

Solve $I \otimes J \vDash [R]$ iff $I \otimes J \vDash R$

Havoc $I \otimes J \vDash \forall V$ iff $I \otimes J \vDash \Diamond(V : [\top] \parallel 1)$

Not even a new thing! [Fischer, Ladner '79] [Balbiani, Herzig, Troquard '13]

Propositional dynamic logic (PDL) defines modalities for each program

# What's in the box? Programs!

Noop $I \otimes J \vDash \mathbf{1}$ iff $I = J$

Composition $I \otimes J \vDash \varepsilon_1\, \varepsilon_2$ iff $\exists K,\ I \otimes K \vDash \varepsilon_1$ and $K \otimes J \vDash \varepsilon_2$

Assignment $I \otimes J \vDash \langle \sigma \rangle$ iff $J = I \circ \sigma$

Choice $I \otimes J \vDash \varepsilon_1 \sqcup \varepsilon_2$ iff $I \otimes J \vDash \varepsilon_1$ or $I \otimes J \vDash \varepsilon_2$

Test $I \otimes J \vDash T?$ iff $I = J$ and $I \vDash T$

Branch $I \otimes J \vDash \nabla(T : \varepsilon_1 \parallel \varepsilon_0)$ iff $I \otimes J \vDash (T?\, \varepsilon_1) \sqcup (\overline{T}?\, \varepsilon_0)$

Repeat $I \otimes J \vDash \varepsilon^*$ iff $\exists I_i,\ I_0 = I,\ I_n = J,\ I_{i-1} \otimes I_i \vDash \varepsilon$

Loop $I \otimes J \vDash \square(T : \varepsilon)$ iff $I \otimes J \vDash (\overline{T}?\varepsilon)^*\, T?$

Rendezvous $I \otimes J \vDash \lozenge(V : \varepsilon_1 \parallel \varepsilon_0)$ iff $\exists J_1, J_0,\ I \otimes J_i \vDash \varepsilon_i$ and $J = J_1 +_V J_0$

Solve $I \otimes J \vDash [R]$ iff $I \otimes J \vDash R$

Havoc $I \otimes J \vDash \forall V$ iff $I \otimes J \vDash \lozenge(V : [\top] \parallel \mathbf{1})$

Not even a new thing! [Fischer, Ladner '79] [Balbiani, Herzig, Troquard '13]

Propositional dynamic logic (PDL) defines modalities for each program

Necessitation law (a.k.a. I can apply programs to a proof)

If $F \vDash G$ holds, then $\varepsilon.F \vDash \varepsilon.G$ holds too

# What can I do with this?

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

# What can I do with this?

**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 $\varepsilon.\bot \vdash \bot$

# What can I do with this?

**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 $\varepsilon.\bot \vdash \bot$

**Parametric proofs** If I have proven $F \vdash G$, then I can prove $F|_\sigma \vdash G|_\sigma$ as $\langle\sigma\rangle.F \vdash \langle\sigma\rangle.G$

# What can I do with this?

**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 $\varepsilon.\bot \vdash \bot$

**Parametric proofs** If I have proven $F \vdash G$, then I can prove $F|_\sigma \vdash G|_\sigma$ as $\langle\sigma\rangle.F \vdash \langle\sigma\rangle.G$

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

# What can I do with this?

**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 $\varepsilon.\bot \vdash \bot$

**Parametric proofs** If I have proven $F \vdash G$, then I can prove $F|_\sigma \vdash G|_\sigma$ as $\langle\sigma\rangle.F \vdash \langle\sigma\rangle.G$

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

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

# What can I do with this?

**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 $\varepsilon.\bot \vdash \bot$

**Parametric proofs** If I have proven $F \vdash G$, then I can prove $F|_\sigma \vdash G|_\sigma$ as $\langle\sigma\rangle.F \vdash \langle\sigma\rangle.G$

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

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

**Refinements** $\varepsilon$ refines $\delta$ if $\varepsilon \vdash [R_\varepsilon]$ and $[R_\delta] \vDash \delta$ and $R_\varepsilon \vdash R_\delta$

# What can I do with this?

**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 $\varepsilon.\bot \vdash \bot$

**Parametric proofs** If I have proven $F \vdash G$, then I can prove $F|_\sigma \vdash G|_\sigma$ as $\langle\sigma\rangle.F \vdash \langle\sigma\rangle.G$

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

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

**Refinements** $\varepsilon$ refines $\delta$ if $\varepsilon \vdash [R_\varepsilon]$ and $[R_\delta] \vDash \delta$ and $R_\varepsilon \vdash R_\delta$

**I think this could be a good foundation for a general, versatile, unified certificate system for propositional reasoning beyond SAT**

# What can I do with this?

**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 $\varepsilon.\bot \vdash \bot$

**Parametric proofs** If I have proven $F \vdash G$, then I can prove $F|_\sigma \vdash G|_\sigma$ as $\langle\sigma\rangle.F \vdash \langle\sigma\rangle.G$

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

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

**Refinements** $\varepsilon$ refines $\delta$ if $\varepsilon \vdash [R_\varepsilon]$ and $[R_\delta] \vDash \delta$ and $R_\varepsilon \vdash R_\delta$

**I think this could be a good foundation for a general, versatile, unified certificate system for propositional reasoning beyond SAT**

An appetizer: [Rebola-Pardo ’25, SYNASC]

# Trying to merge proofs...

all unsat, over the same variables

$F_1$ $F_2$ $F_3$

# Trying to merge proofs...

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

# 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...

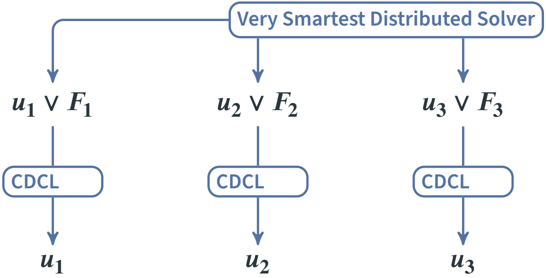
$u_1 \vee F_1$ $\quad$ $u_2 \vee F_2$ $\quad$ $u_3 \vee F_3$ $\quad$ $\overline{u_1} \vee \overline{u_2} \vee \overline{u_3}$

Very Smartest Distributed Solver

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

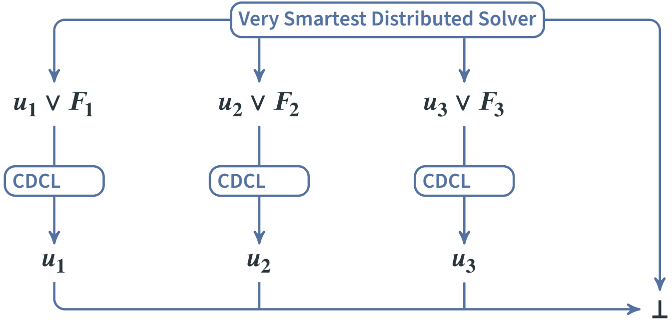
# Trying to merge proofs...

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

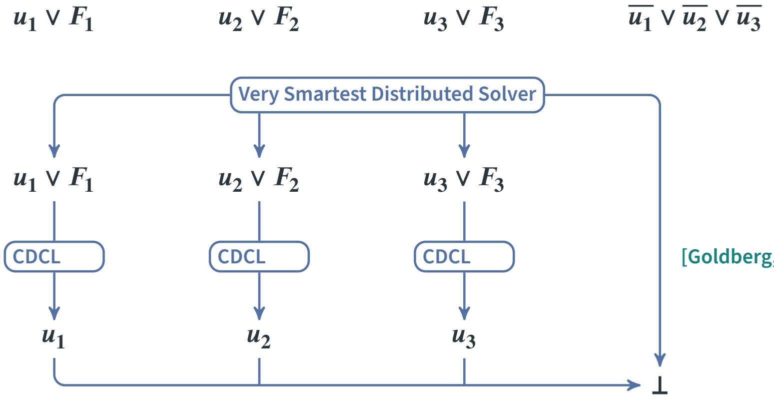


# Trying to merge proofs...

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



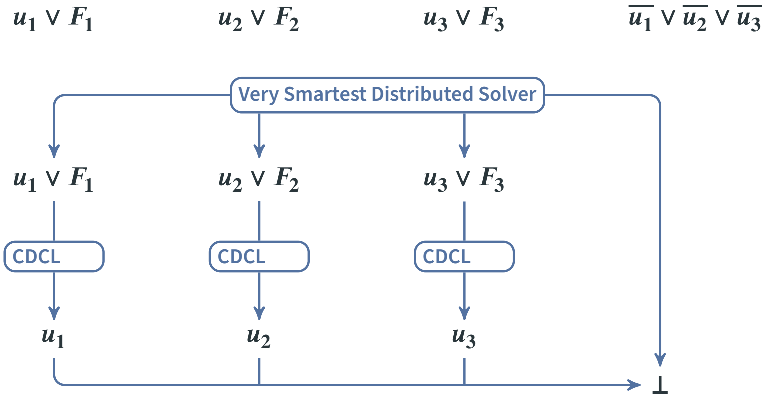
# Trying to merge proofs...




[Goldberg, Novikov '03]

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

# Trying to merge proofs...



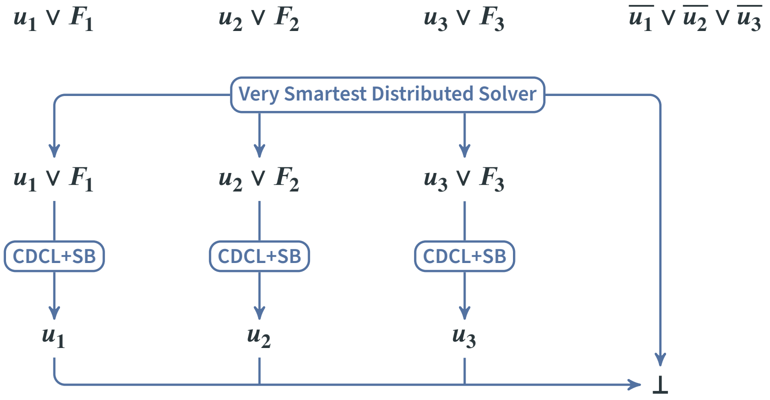

$\pi_1 : u_1 \lor F_1 \vdash u_1$

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

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

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

# Trying to merge proofs...



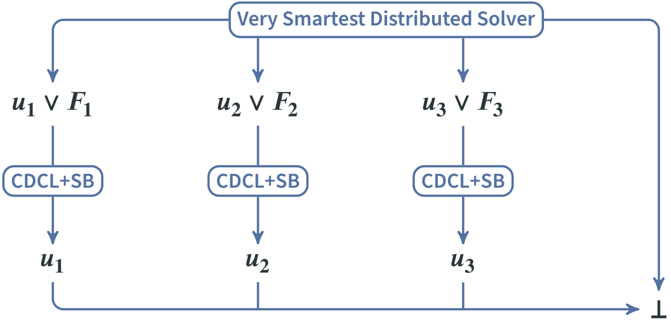

$$\pi_1 : u_1 \vee F_1 \vdash u_1$$
$$\pi_2 : u_2 \vee F_2 \vdash u_2$$
$$\pi_3 : u_3 \vee F_3 \vdash u_3$$

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

# Trying to merge proofs...

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

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




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

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

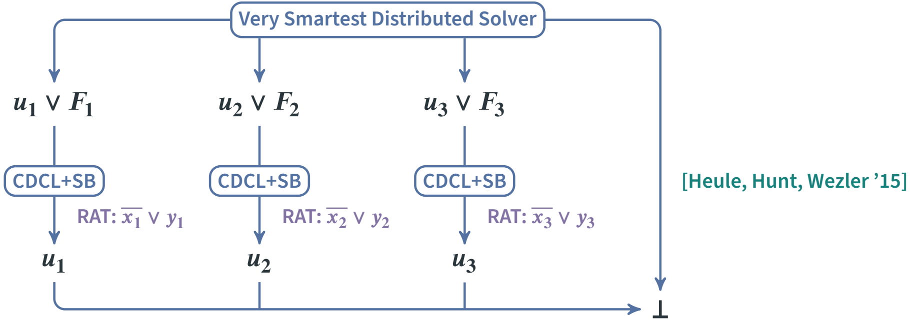
$\pi_3 : u_3 \vee F_3 \vdash u_3$

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

# Trying to merge proofs...

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

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



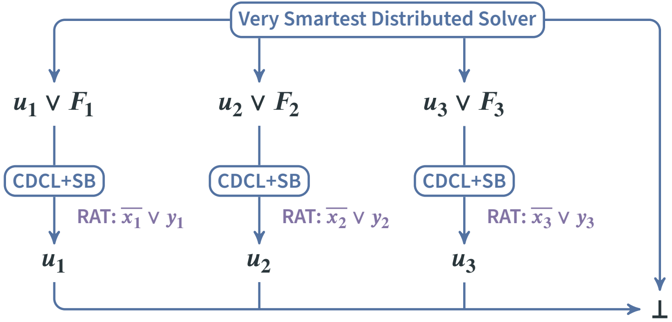

$$\pi_1 : u_1 \lor F_1 \vdash u_1$$
$$\pi_2 : u_2 \lor F_2 \vdash u_2$$
$$\pi_3 : u_3 \lor F_3 \vdash u_3$$

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

# Trying to merge proofs...

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

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




[Järvisalo, Heule, Biere ’12]

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

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

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

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

# … and failing because of interference

**What are DRAT proofs really doing?**

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

# … and failing because of interference

**What are DRAT proofs really doing?**

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

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

# … and failing because of interference

**What are DRAT proofs really doing?**

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

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

*each SR addition of $C$ upon $\sigma$ 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 $I \vDash F$ we have $\mathrm{mut}(I) \vDash G$

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

*each SR addition of $C$ upon $\sigma$ 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 $I \vDash F$ we have $\mathrm{mut}(I) \vDash G$

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

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

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

# … and failing because of interference

**What are DRAT proofs really doing?**

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

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

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

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

$\mathrm{mut}_1$ is "if $I \models x_1 \wedge \overline{y_1}$ 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 $\mathrm{mut}(I) \models G$

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

*each SR addition of $C$ upon $\sigma$ 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 $\mathrm{mut}(I) \models G$

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

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

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

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

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

what we need is this!

# … and failing because of interference

**What are DRAT proofs really doing?**

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

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

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

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

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

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

Interference-based proof systems force a single concurrent $\mathrm{mut}$ prefix…

# … and failing because of interference

What are DRAT proofs really doing?

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

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

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

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

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

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

Interference-based proof systems force a single concurrent $\mathrm{mut}$ prefix…

*… because of the accumulated formula*

# RUP-like inferences in dynamic logic

Unit propagation (BCP) a *blazingly* fast sound but incomplete algorithm for inconsistency in CNF formulas [Zhang, Madigan, Moskewicz, Malik ’01]

# RUP-like inferences in dynamic logic

Unit propagation (BCP) a *blazingly* fast sound but incomplete algorithm for inconsistency in CNF formulas [Zhang, Madigan, Moskewicz, Malik ’01]

Reverse unit propagation (RUP) a *blazingly* fast sound but incomplete algorithm for implication in CNF formulas [Goldberg, Novikov ’03]

# RUP-like inferences in dynamic logic

**Unit propagation (BCP)** a *blazingly* fast sound but incomplete algorithm for inconsistency in CNF formulas [Zhang, Madigan, Moskewicz, Malik '01]

**Reverse unit propagation (RUP)** a *blazingly* fast sound but incomplete algorithm for implication in CNF formulas [Goldberg, Novikov '03]

$$F \wedge \overline{C} \vDash \bot \quad \text{iff} \quad F \vDash C$$

# RUP-like inferences in dynamic logic

**Unit propagation (BCP)** a *blazingly* fast sound but incomplete algorithm for inconsistency in CNF formulas [Zhang, Madigan, Moskewicz, Malik ’01]

**Reverse unit propagation (RUP)** a *blazingly* fast sound but incomplete algorithm for implication in CNF formulas [Goldberg, Novikov ’03]

**Can we have this for dynamic formulas?** yes, but not built on consistency.

$$F \wedge \overline{C} \models \bot \quad \text{iff} \quad F \models C$$

# RUP-like inferences in dynamic logic

**Unit propagation (BCP)** a *blazingly* fast sound but incomplete algorithm for inconsistency in CNF formulas [Zhang, Madigan, Moskewicz, Malik ’01]

**Reverse unit propagation (RUP)** a *blazingly* fast sound but incomplete algorithm for implication in CNF formulas [Goldberg, Novikov ’03]

**Can we have this for dynamic formulas?** yes, but not built on consistency.

$F \wedge \overline{\varepsilon.C} \vDash \bot$ iff $F \vDash \varepsilon.C$ I don’t have diamonds!

# RUP-like inferences in dynamic logic

**Unit propagation (BCP)** a *blazingly* fast sound but incomplete algorithm for inconsistency in CNF formulas [Zhang, Madigan, Moskewicz, Malik '01]

**Reverse unit propagation (RUP)** a *blazingly* fast sound but incomplete algorithm for implication in CNF formulas [Goldberg, Novikov '03]

**Can we have this for dynamic formulas?** yes, but not built on consistency.

Instead we reduce a dynamic implication check to (several) RUP checks

$$F \wedge \overline{\varepsilon.C} \vDash \bot \quad \text{iff} \quad F \vDash \varepsilon.C$$

I don't have diamonds!

# Look Ma! No interference!

$$F = (u_1 \vee F_1) \wedge (u_2 \vee F_2) \wedge (u_3 \vee F_3) \wedge (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

symmetry: $\sigma_1 = \{x_1 \leftrightarrow y_1\}$

# Look Ma! No interference!

$$F \quad = \quad (u_1 \lor F_1) \quad \land \quad (u_2 \lor F_2) \quad \land \quad (u_3 \lor F_3) \quad \land \quad (\overline{u_1} \lor \overline{u_2} \lor \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \lor y_1$ in $u_1 \lor F_1$

symmetry: $\sigma_1 = \{x_1 \leftrightarrow y_1\}$ program: $\nabla(B_1 \,:\, 1 \parallel \langle\sigma_1\rangle)$

# Look Ma! No interference!

$$F = (u_1 \vee F_1) \wedge (u_2 \vee F_2) \wedge (u_3 \vee F_3) \wedge (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

symmetry: $\sigma_1 = \{x_1 \leftrightarrow y_1\}$ program: $\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle)$

$$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).\, B_1$$

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

symmetry: $\sigma_1 = \{x_1 \leftrightarrow y_1\}$ program: $\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle)$

$$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).\, B_1$$

$$F \wedge B_1 \vdash 1.\, B_1 \qquad\qquad F \wedge \overline{B_1} \vdash \langle \sigma_1 \rangle.\, B_1$$

# Look Ma! No interference!

$$F = (u_1 \vee F_1) \wedge (u_2 \vee F_2) \wedge (u_3 \vee F_3) \wedge (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

symmetry: $\sigma_1 = \{x_1 \leftrightarrow y_1\}$ program: $\nabla(B_1 : 1 \parallel \langle\sigma_1\rangle)$

$$F \vdash \nabla(B_1 : 1 \parallel \langle\sigma_1\rangle).\, B_1$$

$$F \wedge B_1 \vdash 1.\, B_1 \qquad F \wedge \overline{B_1} \vdash \langle\sigma_1\rangle.\, B_1$$

$$F \wedge B_1 \vdash B_1$$

trivial by inclusion

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

symmetry: $\sigma_1 = \{x_1 \leftrightarrow y_1\}$ program: $\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle)$

$$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle). B_1$$

$$F \wedge B_1 \vdash 1. B_1 \qquad\qquad F \wedge \overline{B_1} \vdash \langle \sigma_1 \rangle. B_1$$

$$F \wedge B_1 \vdash B_1 \qquad\qquad F \wedge \overline{B_1} \vdash B_1|_{\sigma_1}$$

trivial by inclusion holds by RUP

# Look Ma! No interference!

$$F = (u_1 \lor F_1) \land (u_2 \lor F_2) \land (u_3 \lor F_3) \land (\overline{u_1} \lor \overline{u_2} \lor \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \lor y_1$ in $u_1 \lor F_1$

symmetry: $\sigma_1 = \{x_1 \leftrightarrow y_1\}$ program: $\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle)$

$$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).\, B_1$$

$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).\, C$ for $C \in u_1 \lor F_1$

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

symmetry: $\sigma_1 = \{x_1 \leftrightarrow y_1\}$ program: $\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle)$

$$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).\, B_1$$

$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).\, C$ for $C \in u_1 \vee F_1$

$F \wedge B_1 \vdash 1.\, C$ $\qquad$ $F \wedge \overline{B_1} \vdash \langle \sigma_1 \rangle.\, C$

# Look Ma! No interference!

$$F = (u_1 \lor F_1) \land (u_2 \lor F_2) \land (u_3 \lor F_3) \land (\overline{u_1} \lor \overline{u_2} \lor \overline{u_3})$$

Step 1 introduce the symmetry breaker $B_1 = \overline{x_1} \lor y_1$ in $u_1 \lor F_1$

symmetry: $\sigma_1 = \{x_1 \leftrightarrow y_1\}$ program: $\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle)$

$$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).\, B_1$$

$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).\, C$ for $C \in u_1 \lor F_1$

$F \land B_1 \vdash 1.\, C$ $F \land \overline{B_1} \vdash \langle \sigma_1 \rangle.\, C$

$F \land B_1 \vdash C$

trivial by inclusion

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

symmetry: $\sigma_1 = \{x_1 \leftrightarrow y_1\}$ program: $\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle)$

$$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).\, B_1$$

$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).\, C$ for $C \in u_1 \vee F_1$

$F \wedge B_1 \vdash 1.\, C$ $F \wedge \overline{B_1} \vdash \langle \sigma_1 \rangle.\, C$

$F \wedge B_1 \vdash C$ $F \wedge \overline{B_1} \vdash C|_{\sigma_1}$

trivial by inclusion holds by RUP (because of symmetry)

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).\, (u_1 \vee F_1) \wedge B_1$

symmetry: $\sigma_1 = \{x_1 \leftrightarrow y_1\}$

program: $\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle)$

$$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).\, B_1$$

$$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).\, C \quad \text{for } C \in u_1 \vee F_1$$

$F \wedge B_1 \vdash 1.\, C$

$F \wedge B_1 \vdash C$

trivial by inclusion

$F \wedge \overline{B_1} \vdash \langle \sigma_1 \rangle .\, C$

$F \wedge \overline{B_1} \vdash C|_{\sigma_1}$

holds by RUP (because of symmetry)

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).\, (u_1 \vee F_1) \wedge B_1$

**Step 2** derive $u_1$ from $(u_1 \vee F_1) \wedge B_1$

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle). (u_1 \vee F_1) \wedge B_1$

**Step 2** derive $u_1$ from $(u_1 \vee F_1) \wedge B_1$

$(u_1 \vee F_1) \wedge B_1 \vdash u_1$ by a bunch of RUP steps

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

$F \vdash \nabla(B_1 \,:\, 1 \parallel \langle\sigma_1\rangle).\,(u_1 \vee F_1) \wedge B_1$

**Step 2** derive $u_1$ from $(u_1 \vee F_1) \wedge B_1$

$(u_1 \vee F_1) \wedge B_1 \vdash u_1$ by a bunch of RUP steps

we haven't derived $(u_1 \vee F_1) \wedge B_1$ though...

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).\, (u_1 \vee F_1) \wedge B_1$

**Step 2** derive $u_1$ from $(u_1 \vee F_1) \wedge B_1$

$(u_1 \vee F_1) \wedge B_1 \vdash u_1$ by a bunch of RUP steps

we haven't derived $(u_1 \vee F_1) \wedge B_1$ though...

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

$F \vdash \nabla(B_1 \,:\, 1 \parallel \langle \sigma_1 \rangle).\,(u_1 \vee F_1) \wedge B_1$

**Step 2** derive $u_1$ from $(u_1 \vee F_1) \wedge B_1$

$(u_1 \vee F_1) \wedge B_1 \vdash u_1$ by a bunch of RUP steps

we haven't derived $(u_1 \vee F_1) \wedge B_1$ though...

but we have necessitation!

$$\nabla(B_1 \,:\, 1 \parallel \langle \sigma_1 \rangle).\,(u_1 \vee F_1) \wedge B_1 \vdash \nabla(B_1 \,:\, 1 \parallel \langle \sigma_1 \rangle).u_1$$

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).(u_1 \vee F_1) \wedge B_1$

**Step 2** derive $u_1$ from $(u_1 \vee F_1) \wedge B_1$

$\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).(u_1 \vee F_1) \wedge B_1 \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).u_1$

**Step 3** derive $u_1$ from $u_1 \vee F_1$

# Look Ma! No interference!

$$F = (u_1 \lor F_1) \land (u_2 \lor F_2) \land (u_3 \lor F_3) \land (\overline{u_1} \lor \overline{u_2} \lor \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \lor y_1$ in $u_1 \lor F_1$

$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).(u_1 \lor F_1) \land B_1$

**Step 2** derive $u_1$ from $(u_1 \lor F_1) \land B_1$

$\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).(u_1 \lor F_1) \land B_1 \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).u_1$

**Step 3** derive $u_1$ from $u_1 \lor F_1$

$$\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).u_1 \vdash u_1$$

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

$$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).(u_1 \vee F_1) \wedge B_1$$

**Step 2** derive $u_1$ from $(u_1 \vee F_1) \wedge B_1$

$$\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).(u_1 \vee F_1) \wedge B_1 \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).u_1$$

**Step 3** derive $u_1$ from $u_1 \vee F_1$

$$\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).u_1 \vdash u_1$$

$$B_1 \wedge (1.u_1) \vdash u_1 \qquad\qquad \overline{B_1} \wedge (\langle \sigma_1 \rangle.u_1) \vdash u_1$$

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).(u_1 \vee F_1) \wedge B_1$

**Step 2** derive $u_1$ from $(u_1 \vee F_1) \wedge B_1$

$\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).(u_1 \vee F_1) \wedge B_1 \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).u_1$

**Step 3** derive $u_1$ from $u_1 \vee F_1$

$$\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).u_1 \vdash u_1$$

$B_1 \wedge (1.u_1) \vdash u_1$ $\qquad\qquad$ $\overline{B_1} \wedge (\langle \sigma_1 \rangle.u_1) \vdash u_1$

$B_1 \wedge u_1 \vdash u_1$

trivial by inclusion

# Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle). (u_1 \vee F_1) \wedge B_1$

**Step 2** derive $u_1$ from $(u_1 \vee F_1) \wedge B_1$

$\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle). (u_1 \vee F_1) \wedge B_1 \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle). u_1$

**Step 3** derive $u_1$ from $u_1 \vee F_1$

$$\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle). u_1 \vdash u_1$$

| | |
|---|---|
| $B_1 \wedge (1. u_1) \vdash u_1$ | $\overline{B_1} \wedge (\langle \sigma_1 \rangle. u_1) \vdash u_1$ |
| $B_1 \wedge u_1 \vdash u_1$ | $B_1 \wedge u_1 \vdash u_1\big\vert_{\sigma_1}$ |
| trivial by inclusion | holds by RUP (because of cleanliness) |

[Fazekas, Biere, Scholl '19]

## Look Ma! No interference!

$$F \quad = \quad (u_1 \vee F_1) \quad \wedge \quad (u_2 \vee F_2) \quad \wedge \quad (u_3 \vee F_3) \quad \wedge \quad (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$$

**Step 1** introduce the symmetry breaker $B_1 = \overline{x_1} \vee y_1$ in $u_1 \vee F_1$

$F \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).(u_1 \vee F_1) \wedge B_1$

**Step 2** derive $u_1$ from $(u_1 \vee F_1) \wedge B_1$

$\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).(u_1 \vee F_1) \wedge B_1 \vdash \nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).u_1$

**Step 3** derive $u_1$ from $u_1 \vee F_1$

$\nabla(B_1 : 1 \parallel \langle \sigma_1 \rangle).u_1 \vdash u_1$

**Step 4** repeat for $u_2$ and $u_3$, then resolve

# Conclusion

You don’t need interference for RAT (or PR, or SR, or WSR)

You don’t even need an accumulated formula

You can get new reasoning tools if you stop thinking in terms of redundancy notions

Proof logging and checking is not necessarily more complex