Even shorter proofs without new variables

Adrián Rebola-Pardo

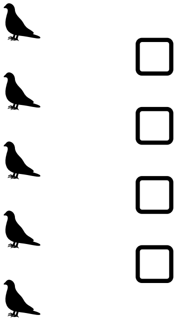
Vienna University of Technology
Johannes Kepler University

26th SAT Conference
Alghero, Italy
July 6th, 2023

Supported by LIT AI Lab (State of Upper Austria), FWF W1255-N23, WWTF VRG11-005, WWTF ICT15-103, and Microsoft Research PhD Programme

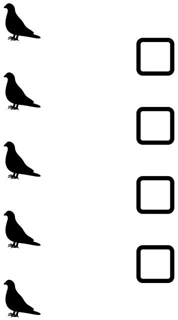
Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Problem 1: swapping pigeons

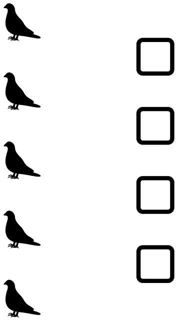
The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n - 1$ holes?

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$

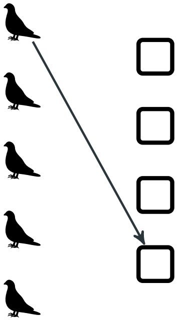


Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$

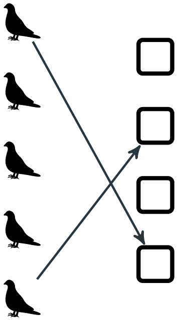


Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$

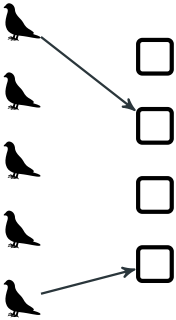


Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



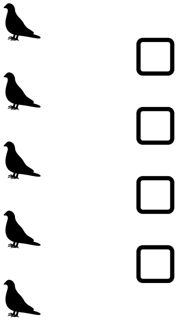
Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$

  otherwise swap pigeons 1 and n

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

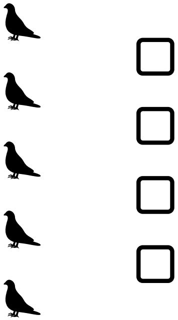
- w.l.o.g. pigeon 1 is not in hole $n-1$

  otherwise swap pigeons 1 *and* n

  ⋮
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

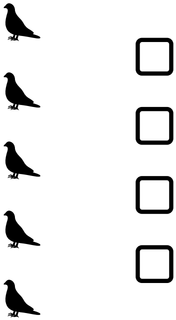
- w.l.o.g. pigeon 1 is not in hole $n-1$

  otherwise swap pigeons 1 and n

  $\vdots$
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1, \ldots, n-2$

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

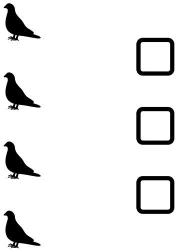
- w.l.o.g. pigeon 1 is not in hole $n-1$
  otherwise swap pigeons 1 and n

  $\vdots$
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1, \ldots, n-2$

  $\vdots$
- pigeon $n-1$ is in some hole $1, \ldots, n-2$

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

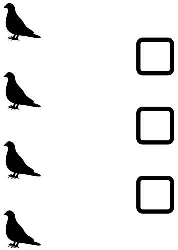
- w.l.o.g. pigeon 1 is not in hole $n-1$
  otherwise swap pigeons 1 and n

  $\vdots$
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1, \dots, n-2$

  $\vdots$
- pigeon $n-1$ is in some hole $1, \dots, n-2$

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

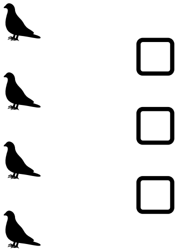
- w.l.o.g. pigeon 1 is not in hole $n-1$
  otherwise swap pigeons 1 and n

  $\vdots$
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1, \dots, n-2$

  $\vdots$
- pigeon $n-1$ is in some hole $1, \dots, n-2$
- solve $\mathrm{PHP}(n-1)$

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$

  otherwise swap pigeons 1 and n

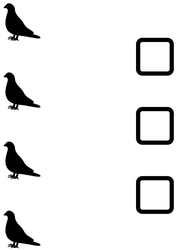
  $\vdots$
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1,\ldots,n-2$

  $\vdots$
- pigeon $n-1$ is in some hole $1,\ldots,n-2$
- solve $\mathrm{PHP}(n-1)$

How long are propositional proofs?

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$
  otherwise swap pigeons 1 and n

  ⋮
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1, \ldots, n-2$

  ⋮
- pigeon $n-1$ is in some hole $1, \ldots, n-2$
- solve $\mathrm{PHP}(n-1)$

How long are propositional proofs?

Theoretical Computer Science 39 (1985) 297–308
North-Holland 297

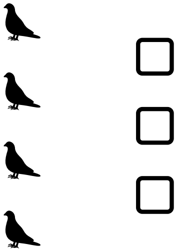
THE INTRACTABILITY OF RESOLUTION

Armin HAKEN
Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A4, Canada

resolution exponential lower bound
classical separation result

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



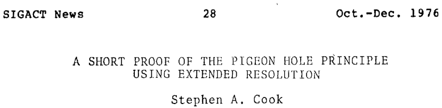
Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$
  otherwise swap pigeons 1 and n

  $\vdots$
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1, \ldots, n-2$

  $\vdots$
- pigeon $n-1$ is in some hole $1, \ldots, n-2$
- solve $\mathrm{PHP}(n-1)$

How long are propositional proofs?


SIGACT News 28 Oct.-Dec. 1976

A SHORT PROOF OF THE PIGEON HOLE PRINCIPLE
USING EXTENDED RESOLUTION

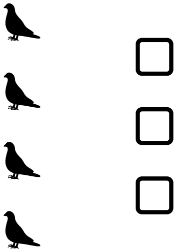
Stephen A. Cook

extended resolution $O(n^4)$

new variables as definitions

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$
  otherwise swap pigeons 1 and n

  $\vdots$
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1, \dots, n-2$

  $\vdots$
- pigeon $n-1$ is in some hole $1, \dots, n-2$
- solve $\mathrm{PHP}(n-1)$

How long are propositional proofs?

Short Proofs Without New Variables

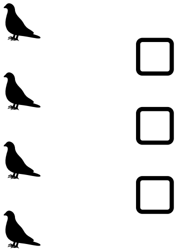
Marijn J.H. Heule[1(✉)], Benjamin Kiesl[2], and Armin Biere[3]

DPR $O(n^3)$

DPR conditionally assigns variables w.l.o.g to $\top/\bot$

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$
  otherwise swap pigeons 1 and n

  $\vdots$
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1, \dots, n-2$

  $\vdots$
- pigeon $n-1$ is in some hole $1, \dots, n-2$
- solve $\mathrm{PHP}(n-1)$

How long are propositional proofs?

Short Proofs Without New Variables

Marijn J.H. Heule[1(✉)], Benjamin Kiesl[2], and Armin Biere[3]

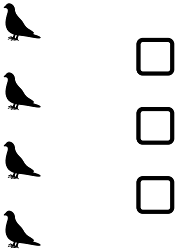
$O(n)$ conditional w.l.o.g.s/pigeon

DPR $O(n^3)$

DPR conditionally assigns variables w.l.o.g to $\top/\bot$

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$
  otherwise swap pigeons 1 and n

  ⋮
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1, \ldots, n-2$

  ⋮
- pigeon $n-1$ is in some hole $1, \ldots, n-2$
- solve $\mathrm{PHP}(n-1)$

How long are propositional proofs?

Short Proofs Without New Variables

Marijn J.H. Heule[1(✉)], Benjamin Kiesl[2], and Armin Biere[3]

$O(n)$ conditional w.l.o.g.s/pigeon

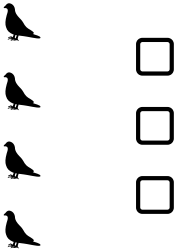
$O(n)$ pigeons/iteration

DPR $O(n^3)$

DPR conditionally assigns variables w.l.o.g to ⊤/⊥

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$

  otherwise swap pigeons 1 and n

  $\vdots$
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1, \dots, n-2$

  $\vdots$
- pigeon $n-1$ is in some hole $1, \dots, n-2$
- solve $\mathrm{PHP}(n-1)$

How long are propositional proofs?

Short Proofs Without New Variables

Marijn J.H. Heule[1(✉)], Benjamin Kiesl[2], and Armin Biere[3]

$O(n)$ conditional w.l.o.g.s/pigeon

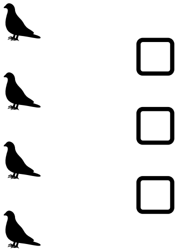
$O(n)$ pigeons/iteration

DPR $O(n^3)$ $O(n)$ iterations

DPR conditionally assigns variables w.l.o.g to $\top/\bot$

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$
  otherwise swap pigeons 1 and n

  $\vdots$
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1, \ldots, n-2$

  $\vdots$
- pigeon $n-1$ is in some hole $1, \ldots, n-2$
- solve $\mathrm{PHP}(n-1)$

How long are propositional proofs?

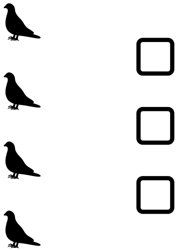
DRAT Proofs, Propagation Redundancy, and Extended Resolution

Sam Buss[1(✉)] and Neil Thapen[2]

DSR conditionally assigns variables w.l.o.g. to literals/⊤/⊥

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$

  otherwise swap pigeons 1 and n

  ⋮
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1, \dots, n-2$

  ⋮
- pigeon $n-1$ is in some hole $1, \dots, n-2$
- solve $\mathrm{PHP}(n-1)$

How long are propositional proofs?

DRAT Proofs, Propagation Redundancy, and Extended Resolution

Sam Buss[1(✉)] and Neil Thapen[2]

DSR conditionally assigns variables w.l.o.g. to literals/⊤/⊥

we can swap variables in $O(1)$!

Problem

The pi

es?

$- 1$

le $n - 1$

$- 2$

$. . , n - 2$

How lo

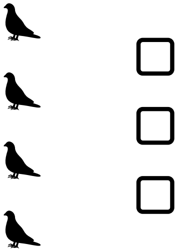
DRA

ables

!

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$
  otherwise swap pigeons 1 and n

  $\vdots$
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1, \dots, n-2$

  $\vdots$
- pigeon $n-1$ is in some hole $1, \dots, n-2$
- solve $\mathrm{PHP}(n-1)$

How long are propositional proofs?

DRAT Proofs, Propagation Redundancy, and Extended Resolution

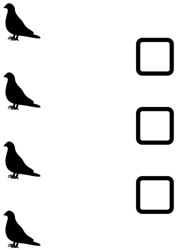
Sam Buss[1(✉)] and Neil Thapen[2]

DSR conditionally assigns variables w.l.o.g. to literals/$\top$/$\bot$

we can swap variables in $O(1)$!

Problem 1: swapping pigeons

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$

  otherwise swap pigeons 1 and n

  $\vdots$
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1, \ldots, n-2$

  $\vdots$
- pigeon $n-1$ is in some hole $1, \ldots, n-2$
- solve $\mathrm{PHP}(n-1)$

How long are propositional proofs?

DRAT Proofs, Propagation Redundancy, and Extended Resolution

Sam Buss[1(✉)] and Neil Thapen[2]

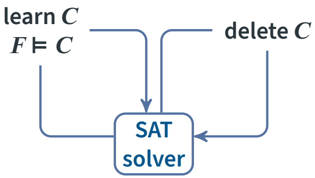
DSR conditionally assigns variables w.l.o.g. to literals/⊤/⊥

we can swap variables in $O(1)$!

why does this fail?

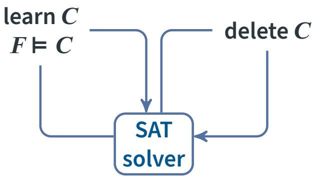
how do we make it work?

Problem 2: interference-free lemmas



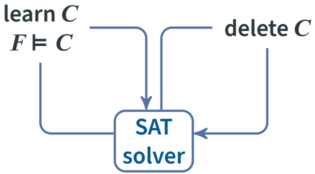
Problem 2: interference-free lemmas

derive as RUP



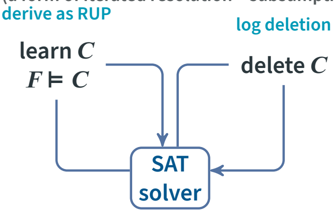
Problem 2: interference-free lemmas

(a form of iterated resolution + subsumption)
derive as RUP



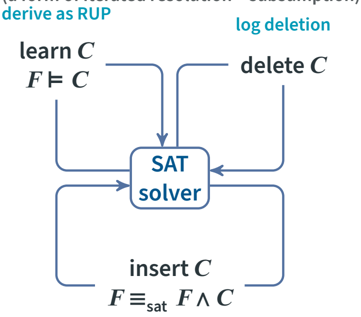
Problem 2: interference-free lemmas

(a form of iterated resolution + subsumption)



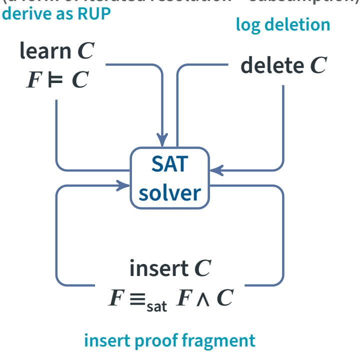
Problem 2: interference-free lemmas

(a form of iterated resolution + subsumption)

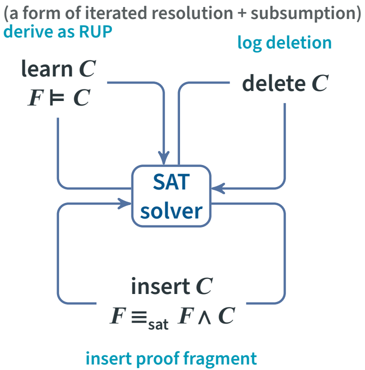


Problem 2: interference-free lemmas

(a form of iterated resolution + subsumption)



Problem 2: interference-free lemmas



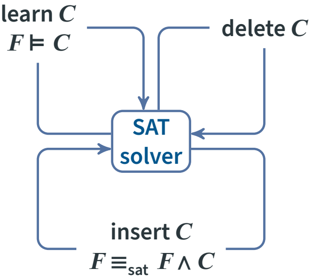
Proof generation for inprocessing

Problem 2: interference-free lemmas

(a form of iterated resolution + subsumption)

derive as RUP

log deletion

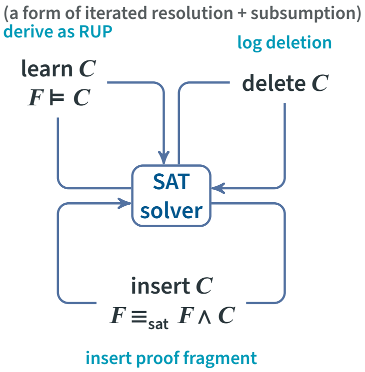


insert proof fragment

Proof generation for inprocessing

if C is a RUP over F

Problem 2: interference-free lemmas

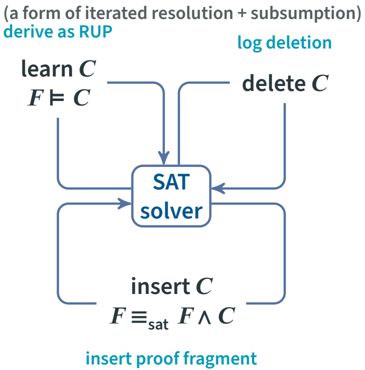


Proof generation for inprocessing

i: C

if C is a RUP over F

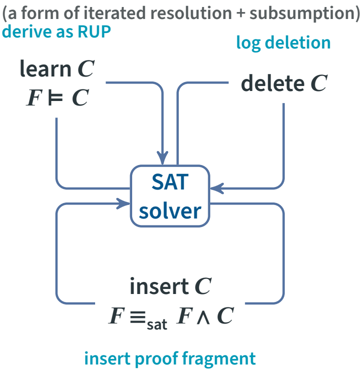
Problem 2: interference-free lemmas



Proof generation for inprocessing

i: C

Problem 2: interference-free lemmas

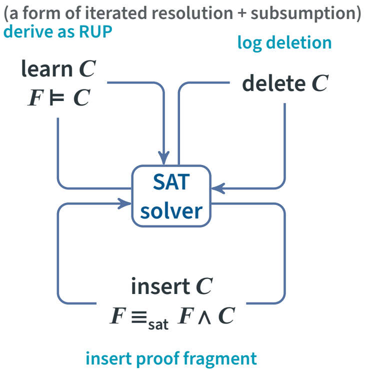


Proof generation for inprocessing

i: C

if C is implied by F

Problem 2: interference-free lemmas

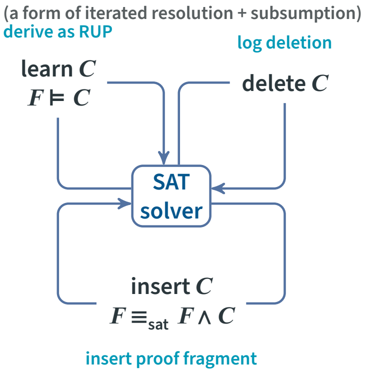


Proof generation for inprocessing

i: L_1
i: L_2
i: L_3
i: C
d: L_3
d: L_2
d: L_1

if C is implied by F

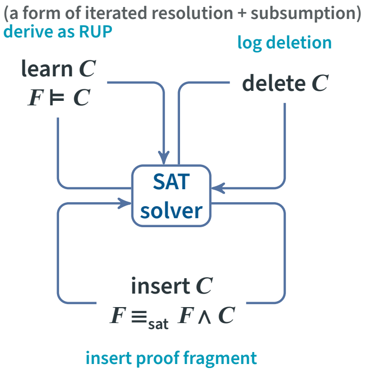
Problem 2: interference-free lemmas



Proof generation for inprocessing

i: C

Problem 2: interference-free lemmas



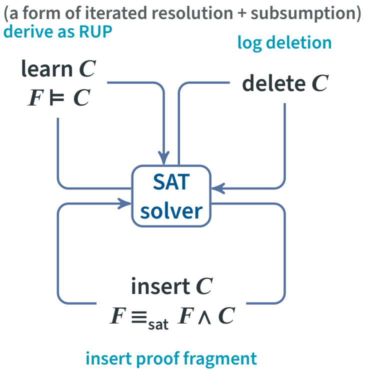
Proof generation for inprocessing

i: C

if C is an SR clause over F upon σ

C is an SR clause if $\sigma(C)$ is a tautology and all clauses in $\sigma(F)$ are RUP clauses over $F|_{\bar{C}}$

Problem 2: interference-free lemmas



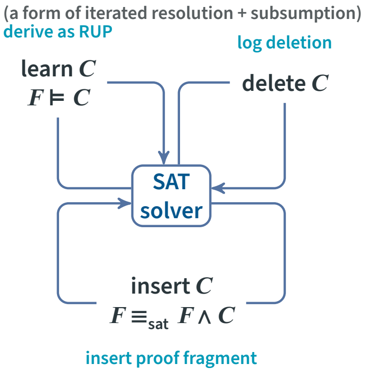
Proof generation for inprocessing

i: C $[\sigma]$

if C is an SR clause over F upon σ

C is an SR clause if $\sigma(C)$ is a tautology and all clauses in $\sigma(F)$ are RUP clauses over $F|_{\bar{C}}$

Problem 2: interference-free lemmas

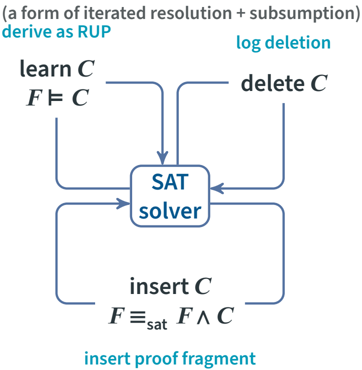


Proof generation for inprocessing

i: C $[\sigma]$

C is an SR clause if $\sigma(C)$ is a tautology and all clauses in $\sigma(F)$ are RUP clauses over $F|_{\bar{C}}$

Problem 2: interference-free lemmas



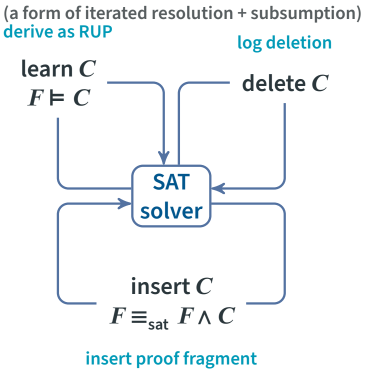
Proof generation for inprocessing

i: C [σ]

if $F|_{\bar{C}} \vDash \sigma(F)$

C is an SR clause if $\sigma(C)$ is a tautology and all clauses in $\sigma(F)$ are RUP clauses over $F|_{\bar{C}}$

Problem 2: interference-free lemmas



Proof generation for inprocessing

i: L_1
i: L_2
i: L_3
i: C $[\sigma]$
d: L_3
d: L_2
d: L_1

all clauses in $\sigma(F)$ are RUP clauses over $F \wedge L_1 \wedge L_2 \wedge L_3$

if $F|_{\bar{C}} \models \sigma(F)$

C is an SR clause if $\sigma(C)$ is a tautology and all clauses in $\sigma(F)$ are RUP clauses over $F|_{\bar{C}}$

Problem

(a form of

derive as

learn

$F \vDash ($

ing

(F)

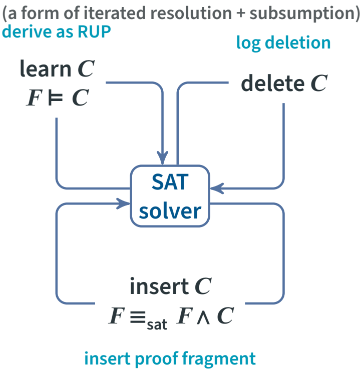
es

$\wedge L_3$

logy and

s over $F|_{\bar{C}}$

Problem 2: interference-free lemmas



Proof generation for inprocessing

i: L_1
i: L_2
i: L_3
i: C $[\sigma]$
d: L_3
d: L_2
d: L_1

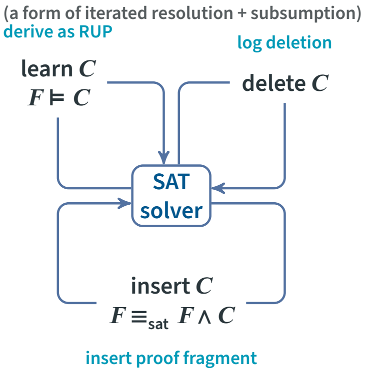
all clauses in $\sigma(F)$
are RUP clauses
over $F \wedge L_1 \wedge L_2 \wedge L_3$

if $F|_{\bar{C}} \vDash \sigma(F)$

C is an SR clause if $\sigma(C)$ is a tautology and all clauses in $\sigma(F)$ are RUP clauses over $F|_{\bar{C}}$

we also need $\sigma(L_1), \sigma(L_2), \sigma(L_3)$ to be RUPs over $F \wedge L_1 \wedge L_2 \wedge L_3$

Problem 2: interference-free lemmas



Proof generation for inprocessing

i: L_1
i: L_2
i: L_3
i: C $[\sigma]$
d: L_3
d: L_2
d: L_1

all clauses in $\sigma(F)$ are RUP clauses over $F \wedge L_1 \wedge L_2 \wedge L_3$

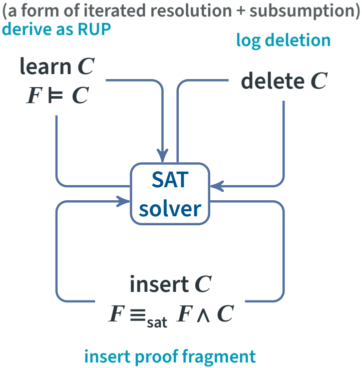
if $F|_{\bar{C}} \vDash \sigma(F)$

C is an SR clause if $\sigma(C)$ is a tautology and all clauses in $\sigma(F)$ are RUP clauses over $F|_{\bar{C}}$

we also need $\sigma(L_1), \sigma(L_2), \sigma(L_3)$ to be RUPs over $F \wedge L_1 \wedge L_2 \wedge L_3$

... which might need extra lemmas themselves...

Problem 2: interference-free lemmas



Proof generation for inprocessing

i: L_1
i: L_2
i: L_3
i: $C\ [\sigma]$
d: L_3
d: L_2
d: L_1

all clauses in $\sigma(F)$ are RUP clauses over $F \wedge L_1 \wedge L_2 \wedge L_3$

if $F|_{\bar{C}} \models \sigma(F)$

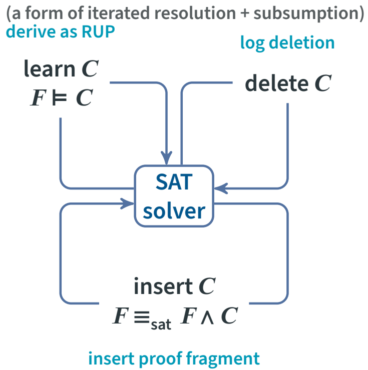
C is an SR clause if $\sigma(C)$ is a tautology and all clauses in $\sigma(F)$ are RUP clauses over $F|_{\bar{C}}$

we also need $\sigma(L_1), \sigma(L_2), \sigma(L_3)$ to be RUPs over $F \wedge L_1 \wedge L_2 \wedge L_3$

... which might need extra lemmas themselves...

... and so on...

Problem 2: interference-free lemmas



Proof generation for inprocessing

i: L_1
i: L_2
i: L_3
i: C $[\sigma]$
d: L_3
d: L_2
d: L_1

all clauses in $\sigma(F)$ are RUP clauses over $F \wedge L_1 \wedge L_2 \wedge L_3$

if $F|_{\bar{C}} \vDash \sigma(F)$

C is an SR clause if $\sigma(C)$ is a tautology and all clauses in $\sigma(F)$ are RUP clauses over $F|_{\bar{C}}$

we also need $\sigma(L_1), \sigma(L_2), \sigma(L_3)$ to be RUPs over $F \wedge L_1 \wedge L_2 \wedge L_3$

... which might need extra lemmas themselves...

... and so on...

how can we temporarily introduce interference-free lemmas?

Problem 3: cores and trimming under interference

Generating an unsatisfiable core from a proof

Problem 3: cores and trimming under interference

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

Problem 3: cores and trimming under interference

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- **if C is not marked, skip it**

Problem 3: cores and trimming under interference

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clause, it is in the core

Problem 3: cores and trimming under interference

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clause, it is in the core
- if C is a RUP clause, mark its antecedents

Problem 3: cores and trimming under interference

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clause, it is in the core
- if C is a RUP clause, mark its antecedents

$$\text{sub}\ \frac{E_0}{A_0} \qquad \text{res}\ \frac{A_0 \qquad E_1}{A_1} \qquad \text{res}\ \frac{A_1 \qquad E_2}{A_2} \qquad \ddots \qquad \text{res}\ \frac{A_{n-1} \qquad E_n}{C}$$

Problem 3: cores and trimming under interference

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clause, it is in the core
- if C is a RUP clause, mark its antecedents

$$\text{res}\ \dfrac{\begin{matrix}\text{res}\ \dfrac{\text{res}\ \dfrac{\text{sub}\ \dfrac{E_0\ \text{mark}}{A_0} \quad E_1\ \text{mark}}{A_1} \quad E_2\ \text{mark}}{A_2} \\ \ddots \\ A_{n-1}\end{matrix} \quad E_n\ \text{mark}}{C}$$

Problem 3: cores and trimming under interference

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

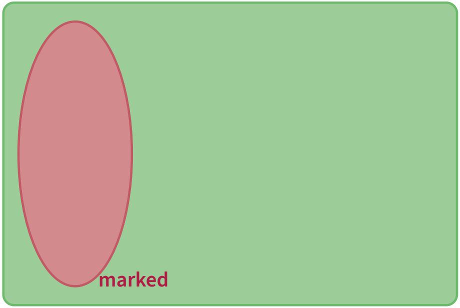
- if C is not marked, skip it
- if C is an input clause, it is in the core
- if C is a RUP clause, mark its antecedents
- if C is an SR clause upon σ...?

Problem 3: cores and trimming under interference

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clause, it is in the core
- if C is a RUP clause, mark its antecedents
- if C is an SR clause upon σ...?

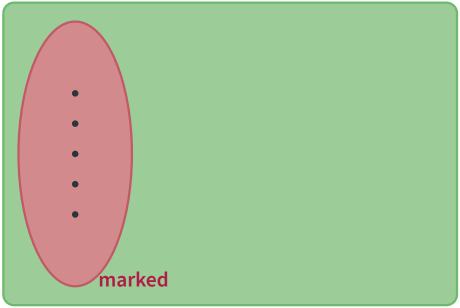


Problem 3: cores and trimming under interference

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clause, it is in the core
- if C is a RUP clause, mark its antecedents
- if C is an SR clause upon σ...?



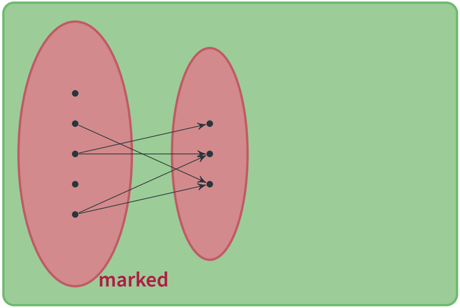
for each marked D, $\sigma(D)$ is a RUP clause over $F|_{\bar{C}}$ $\Rightarrow$ mark their antecedents

Problem 3: cores and trimming under interference

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clause, it is in the core
- if C is a RUP clause, mark its antecedents
- if C is an SR clause upon σ...?



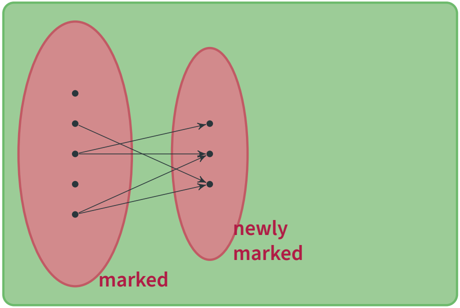
for each marked D, $\sigma(D)$ is a RUP clause over $F|_{\bar{C}}$ $\Rightarrow$ mark their antecedents

Problem 3: cores and trimming under interference

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clause, it is in the core
- if C is a RUP clause, mark its antecedents
- if C is an SR clause upon σ...?



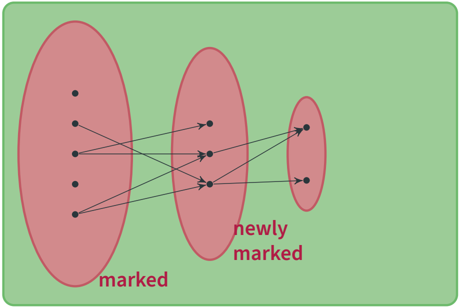
for each marked D, $\sigma(D)$ is a RUP clause over $F|_{\bar{C}}$ $\Rightarrow$ mark their antecedents

Problem 3: cores and trimming under interference

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clause, it is in the core
- if C is a RUP clause, mark its antecedents
- if C is an SR clause upon σ...?



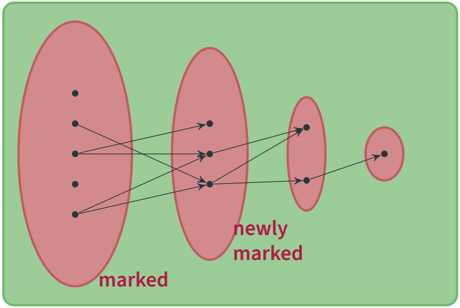
for each marked D, $\sigma(D)$ is a RUP clause over $F|_{\bar{C}}$ $\Rightarrow$ mark their antecedents

Problem 3: cores and trimming under interference

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clause, it is in the core
- if C is a RUP clause, mark its antecedents
- if C is an SR clause upon σ...?



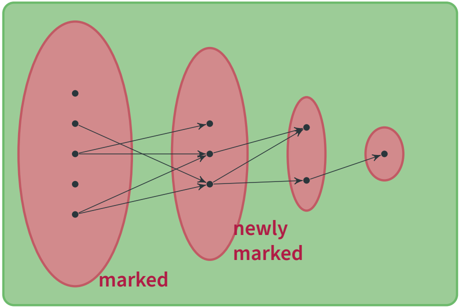
for each marked D, $\sigma(D)$ is a RUP clause over $F|_{\bar{C}}$ $\Rightarrow$ mark their antecedents

Problem 3: cores and trimming under interference

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clause, it is in the core
- if C is a RUP clause, mark its antecedents
- if C is an SR clause upon σ...?



can we do better than this fixpoint computation?

for each marked D, $\sigma(D)$ is a RUP clause over $F|_{\bar{C}}$ $\Rightarrow$ mark their antecedents

In this talk...

DRAT/DPR/DSR proofs

interference-free lemmas

pigeonhole refutation

non-fixpoint cores and trimming

In this talk...

DRAT/DPR/DSR proofs

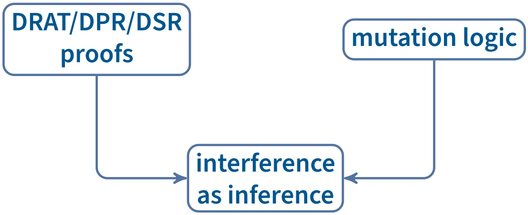
mutation logic

interference-free lemmas

pigeonhole refutation

non-fixpoint cores and trimming

In this talk...

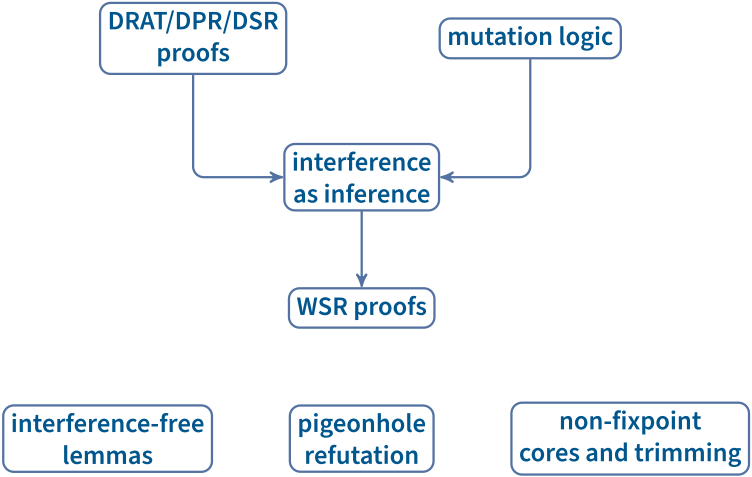


interference-free lemmas

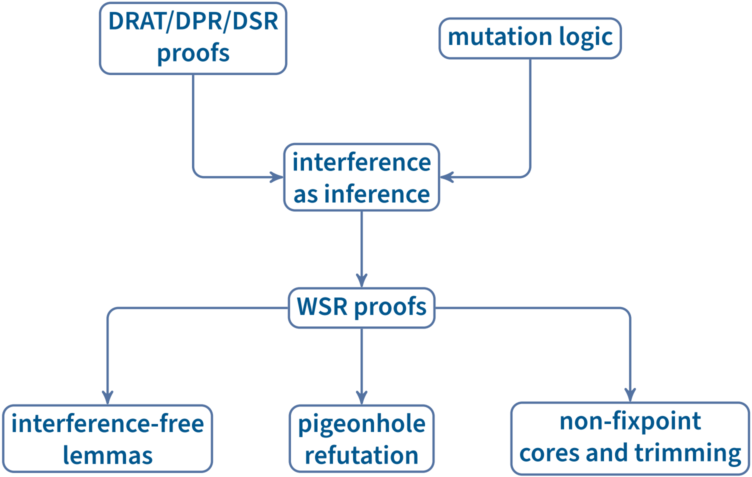
pigeonhole refutation

non-fixpoint cores and trimming

In this talk...


DRAT/DPR/DSR proofs
mutation logic
interference as inference
WSR proofs
interference-free lemmas
pigeonhole refutation
non-fixpoint cores and trimming

In this talk...



DRAT/DPR/DSR proofs
mutation logic
interference as inference
WSR proofs
interference-free lemmas
pigeonhole refutation
non-fixpoint cores and trimming

The interference zoo

Propagation-based redundancy notions

The interference zoo

Propagation-based redundancy notions

RUP [Goldberg, Novikov ’03]

C is a reverse unit propagation (RUP) over F whenever unit propagation over $F \wedge \neg C$ reaches a conflict

The interference zoo

Propagation-based redundancy notions

C is a reverse unit propagation (RUP) over F whenever unit propagation over $F \wedge \neg C$ reaches a conflict

Properties

The interference zoo

Propagation-based redundancy notions

C is a reverse unit propagation (RUP) over F whenever unit propagation over $F \wedge \neg C$ reaches a conflict

Properties

RUP is monotonic if $F \subseteq G$, then C is a RUP over G

Propagation-based redundancy notions

C is a reverse unit propagation (RUP) over F whenever unit propagation over $F \wedge \neg C$ reaches a conflict

Properties

- RUP is monotonic — if $F \subseteq G$, then C is a RUP over G
- RUP preserves models — every model of F is a model of $F \wedge C$

The interference zoo

Propagation-based redundancy notions

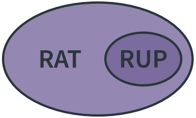
C is a reverse unit propagation (RUP) over F whenever unit propagation over $F \wedge \neg C$ reaches a conflict

Properties

- RUP is monotonic: if $F \subseteq G$, then C is a RUP over G
- RUP preserves models: every model of F is a model of $F \wedge C$
- RUP has a dependency structure: C is an iterated resolvent over F

The interference zoo

Propagation-based redundancy notions



[Järvisalo, Heule, Biere ’12]

C is a reverse unit propagation (RUP) over F whenever unit propagation over $F \wedge \neg C$ reaches a conflict

Properties

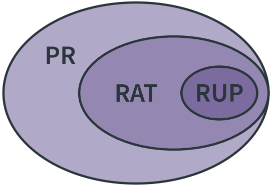
RUP is monotonic — if $F \subseteq G$, then C is a RUP over G

RUP preserves models — every model of F is a model of $F \wedge C$

RUP has a dependency structure — C is an iterated resolvent over F

The interference zoo

Propagation-based redundancy notions



[Heule, Kiesl, Biere ’17]

C is a reverse unit propagation (RUP) over F whenever unit propagation over $F \wedge \neg C$ reaches a conflict

Properties

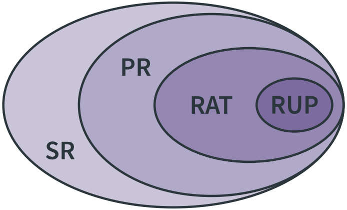
RUP is monotonic if $F \subseteq G$, then C is a RUP over G

RUP preserves models every model of F is a model of $F \wedge C$

RUP has a dependency structure C is an iterated resolvent over F

The interference zoo

Propagation-based redundancy notions



[Buss, Thapen ’19]

C is a reverse unit propagation (RUP) over F whenever unit propagation over $F \land \neg C$ reaches a conflict

Properties

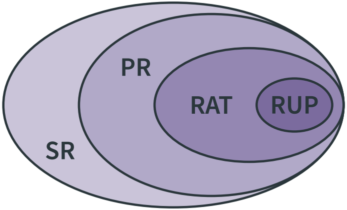
RUP is monotonic if $F \subseteq G$, then C is a RUP over G

RUP preserves models every model of F is a model of $F \land C$

RUP has a dependency structure C is an iterated resolvent over F

The interference zoo

Propagation-based redundancy notions



C is a reverse unit propagation (RUP) over F whenever unit propagation over $F \wedge \neg C$ reaches a conflict

Properties

RUP is monotonic if $F \subseteq G$, then C is a RUP over G

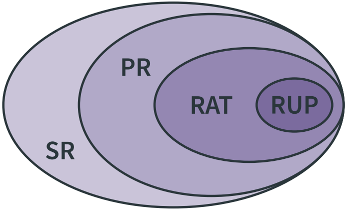
RAT/PR/SR are non-monotonic!

RUP preserves models every model of F is a model of $F \wedge C$

RUP has a dependency structure C is an iterated resolvent over F

The interference zoo

Propagation-based redundancy notions



C is a reverse unit propagation (RUP) over F whenever unit propagation over $F \wedge \neg C$ reaches a conflict

Properties

RUP is monotonic — if $F \subseteq G$, then C is a RUP over G

RAT/PR/SR are non-monotonic!

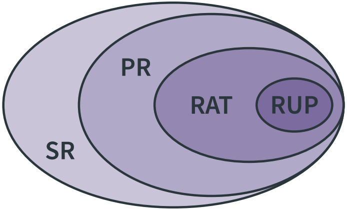
RUP preserves models — every model of F is a model of $F \wedge C$

RAT/PR/SR preserve satisfiability: if F is satisfiable, then so is $F \wedge C$

RUP has a dependency structure — C is an iterated resolvent over F

The interference zoo

Propagation-based redundancy notions



C is a reverse unit propagation (RUP) over F whenever unit propagation over $F \land \neg C$ reaches a conflict

Properties

RUP is monotonic — if $F \subseteq G$, then C is a RUP over G

RAT/PR/SR are non-monotonic!

RUP preserves models — every model of F is a model of $F \land C$

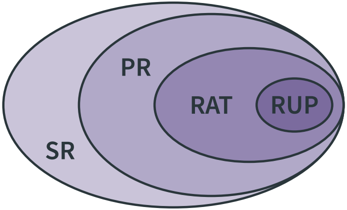
RAT/PR/SR preserve satisfiability: if F is satisfiable, then so is $F \land C$

RUP has a dependency structure — C is an iterated resolvent over F

RAT/PR/SR depend on the absence of clauses

The interference zoo

Propagation-based redundancy notions



C is a reverse unit propagation (RUP) over F whenever unit propagation over $F \wedge \neg C$ reaches a conflict

Properties

RUP is monotonic if $F \subseteq G$, then C is a RUP over G

RAT/PR/SR are non-monotonic!

RUP preserves models every model of F is a model of $F \wedge C$

RAT/PR/SR preserve satisfiability: if F is satisfiable, then so is $F \wedge C$

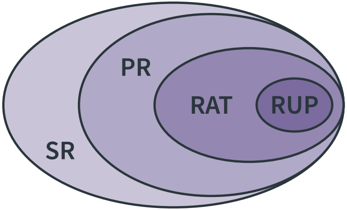
RUP has a dependency structure C is an iterated resolvent over F

RAT/PR/SR depend on the absence of clauses

lack of these properties is called *interference*

The interference zoo

Propagation-based redundancy notions



DRUP	deletion + RUP
DRAT	deletion + RAT
DPR	deletion + PR
DSR	deletion + SR

C is a reverse unit propagation (RUP) over F whenever unit propagation over $F \wedge \neg C$ reaches a conflict

Properties

RUP is monotonic   if $F \subseteq G$, then C is a RUP over G

RAT/PR/SR are non-monotonic!

RUP preserves models   every model of F is a model of $F \wedge C$

RAT/PR/SR preserve satisfiability: if F is satisfiable, then so is $F \wedge C$

RUP has a dependency structure C is an iterated resolvent over F

RAT/PR/SR depend on the absence of clauses

lack of these properties is called *interference*

Substitution redundancy

F	CNF formula
C	clause
σ	atomic substitution

Substitution redundancy

F	CNF formula
C	clause
σ	atomic substitution

C is a substitution redundant (SR) clause over F upon σ whenever:

- the clause $\sigma(C)$ is a tautology
- for each clause $D \in F$, the clause $C \vee \sigma(D)$ is a RUP clause over F

Substitution redundancy

F	CNF formula
C	clause
σ	atomic substitution

C is a substitution redundant (SR) clause over F upon σ whenever:

- the clause $\sigma(C)$ is a tautology
- for each clause $D \in F$, the clause $C \vee \sigma(D)$ is a RUP clause over F

Theorem   if C is an SR clause over F upon σ, then $F \equiv_{\text{sat}} F \wedge C$

Substitution redundancy

F	CNF formula
C	clause
σ	atomic substitution

C is a substitution redundant (SR) clause over F upon σ whenever:

- the clause $\sigma(C)$ is a tautology
- for each clause $D \in F$, the clause $C \vee \sigma(D)$ is a RUP clause over F

Theorem if C is an SR clause over F upon σ, then $F \equiv_{\text{sat}} F \wedge C$

Intuition if a model of F falsifies C, then σ transforms that model into a model of $F \wedge C$.

this is reasoning without loss of generality!

Substitution redundancy

F CNF formula
C clause
σ atomic substitution

C is a substitution redundant (SR) clause over F upon σ whenever:

- the clause $\sigma(C)$ is a tautology
- for each clause $D \in F$, the clause $C \vee \sigma(D)$ is a RUP clause over F

Theorem if C is an SR clause over F upon σ, then $F \equiv_{\text{sat}} F \wedge C$

Intuition if a model of F falsifies C, then σ transforms that model into a model of $F \wedge C$.

this is reasoning without loss of generality!

can we relax the conditions for SR?

this extends work from [Rebola-Pardo, Suda ’18]

Substitutions can be seen as transformations on interpretations

this extends work from [Rebola-Pardo, Suda '18]

Substitutions can be seen as transformations on interpretations

interpretations $I:$ variables $\to \{0,1\}$
formulas $\to \{0,1\}$

this extends work from [Rebola-Pardo, Suda '18]

Substitutions can be seen as transformations on interpretations

interpretations $I:$ variables $\to \{0,1\}$
formulas $\to \{0,1\}$

substitutions $\sigma:$ variables $\to$ formulas

Mutation logic, or how I learned to stop worrying and love interference

this extends work from [Rebola-Pardo, Suda ’18]

Substitutions can be seen as transformations on interpretations

interpretations	$I :$	variables $\rightarrow \{0, 1\}$
		formulas $\rightarrow \{0, 1\}$
substitutions	$\sigma :$	variables $\rightarrow$ formulas
mutations	$I \circ \sigma :$	variables $\rightarrow \{0, 1\}$

Mutation logic, or how I learned to stop worrying and love interference

this extends work from [Rebola-Pardo, Suda ’18]

Substitutions can be seen as transformations on interpretations

interpretations	$I :$ variables $\to \{0, 1\}$ formulas $\to \{0, 1\}$
substitutions	$\sigma :$ variables $\to$ formulas
mutations	$I \circ \sigma :$ variables $\to \{0, 1\}$
conditional mutations	$I \circ (\sigma :\!- Q) = \begin{cases} I \circ \sigma & \text{if } I \models Q \\ I & \text{if } I \not\models Q \end{cases}$

Mutation logic, or how I learned to stop worrying and love interference

this extends work from [Rebola-Pardo, Suda '18]

Substitutions can be seen as transformations on interpretations

interpretations $I :$ variables $\to \{0, 1\}$
formulas $\to \{0, 1\}$

substitutions $\sigma :$ variables $\to$ formulas

mutations $I \circ \sigma :$ variables $\to \{0, 1\}$

conditional mutations

$$\underbrace{I \circ (\sigma}_{\text{effect}} \;{:}{-}\; \underbrace{Q)}_{\text{trigger}} = \begin{cases} I \circ \sigma & \text{if } I \models Q \\ I & \text{if } I \not\models Q \end{cases}$$

this extends work from [Rebola-Pardo, Suda ’18]

Substitutions can be seen as transformations on interpretations

interpretations	$I :$ variables $\to \{0, 1\}$ formulas $\to \{0, 1\}$
substitutions	$\sigma :$ variables $\to$ formulas
mutations	$I \circ \sigma :$ variables $\to \{0, 1\}$
conditional mutations	$I \circ (\sigma :\!- Q) = \begin{cases} I \circ \sigma & \text{if } I \models Q \\ I & \text{if } I \not\models Q \end{cases}$

Mutation logic propositional logic + mutation operator

$$\nabla(\sigma :\!- Q).\, F$$

Mutation logic, or how I learned to stop worrying and love interference

this extends work from [Rebola-Pardo, Suda ’18]

Substitutions can be seen as transformations on interpretations

interpretations	$I:$ variables $\to \{0,1\}$ formulas $\to \{0,1\}$
substitutions	$\sigma:$ variables $\to$ formulas
mutations	$I \circ \sigma:$ variables $\to \{0,1\}$
conditional mutations	$I \circ (\sigma :\!- Q) = \begin{cases} I \circ \sigma & \text{if } I \vDash Q \\ I & \text{if } I \nvDash Q \end{cases}$

Mutation logic   propositional logic + mutation operator

$$I \vDash \nabla(\sigma :\!- Q).\, F \quad \text{iff} \quad I \circ (\sigma :\!- Q) \vDash F$$

Mutation logic, or how I learned to stop worrying and love interference

this extends work from [Rebola-Pardo, Suda '18]

Substitutions can be seen as transformations on interpretations

interpretations $I\colon$ variables $\to \{0,1\}$
formulas $\to \{0,1\}$

substitutions $\sigma\colon$ variables $\to$ formulas

mutations $I \circ \sigma\colon$ variables $\to \{0,1\}$

conditional mutations
$$I \circ (\sigma :- Q) = \begin{cases} I \circ \sigma & \text{if } I \vDash Q \\ I & \text{if } I \not\vDash Q \end{cases}$$

Mutation logic propositional logic + mutation operator

$$I \vDash \nabla(\sigma :- Q).\, F \quad \text{iff} \quad I \circ (\sigma :- Q) \vDash F$$

Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :- \bar{C}).F \wedge C$

Interference is inference (in mutation logic)

$$I \models \nabla(\sigma :- Q).\ F \quad \text{iff} \quad I \circ (\sigma :- Q) \models F$$

Theorem if C is an SR clause over F upon σ, then $F \models \nabla(\sigma :- \bar{C}).F \wedge C$

Interference is inference (in mutation logic)

$$I \models \nabla(\sigma :- Q).\, F \quad \text{iff} \quad I \circ (\sigma :- Q) \models F$$

Theorem if C is an SR clause over F upon σ, then $F \models \nabla(\sigma :- \bar{C}).F \wedge C$

$\forall D \in F$,
$C \vee \sigma(D)$ is RUP over F

Interference is inference (in mutation logic)

$$I \vDash \nabla(\sigma :\!- Q).\, F \quad \text{iff} \quad I \circ (\sigma :\!- Q) \vDash F$$

Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :\!- \bar{C}).F \wedge C$

$$\boxed{\begin{array}{l}\forall D \in F, \\ C \vee \sigma(D) \text{ is RUP over } F\end{array}}$$

$$\downarrow$$

$$\boxed{\begin{array}{l}\forall D \in F, \\ F \vDash C \vee \sigma(D)\end{array}}$$

Interference is inference (in mutation logic)

$$I \vDash \nabla(\sigma :- Q).\, F \quad \text{iff} \quad I \circ (\sigma :- Q) \vDash F$$

Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :- \bar{C}).F \wedge C$

$\forall D \in F,$
$C \vee \sigma(D)$ is RUP over F

$\downarrow$

$\forall D \in F,$
$F \vDash C \vee \sigma(D)$

$\downarrow$

$\forall D \in F,$
$F \wedge \bar{C} \vDash \sigma(D)$

Interference is inference (in mutation logic)

$$I \vDash \nabla(\sigma :- Q).\, F \quad \text{iff} \quad I \circ (\sigma :- Q) \vDash F$$

Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :- \bar{C}).F \wedge C$

$\forall D \in F,$
$C \vee \sigma(D)$ is RUP over F

$\downarrow$

$\forall D \in F,$
$F \vDash C \vee \sigma(D)$

$\downarrow$

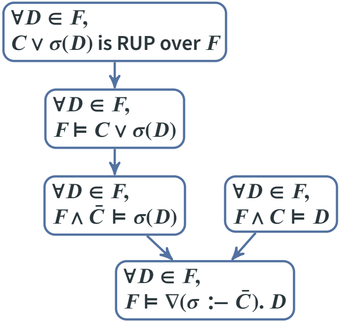
$\forall D \in F,$
$F \wedge \bar{C} \vDash \sigma(D)$

$\forall D \in F,$
$F \wedge C \vDash D$

Interference is inference (in mutation logic)

$$I \vDash \nabla(\sigma :\!- Q).\, F \quad \text{iff} \quad I \circ (\sigma :\!- Q) \vDash F$$

Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :\!- \bar{C}).F \wedge C$



Interference is inference (in mutation logic)

$$I \vDash \nabla(\sigma \coloneq Q).\, F \quad \text{iff} \quad I \circ (\sigma \coloneq Q) \vDash F$$

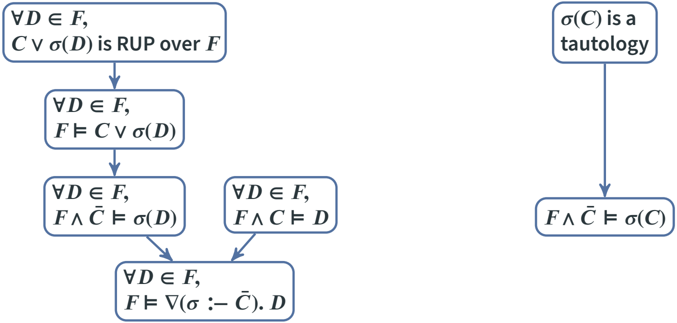
Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma \coloneq \bar{C}).F \wedge C$

$$\begin{tikzcd}
\boxed{\begin{array}{l}\forall D \in F, \\ C \vee \sigma(D) \text{ is RUP over } F\end{array}} \arrow[d] & & \boxed{\begin{array}{l}\sigma(C) \text{ is a} \\ \text{tautology}\end{array}} \\
\boxed{\begin{array}{l}\forall D \in F, \\ F \vDash C \vee \sigma(D)\end{array}} \arrow[d] & & \\
\boxed{\begin{array}{l}\forall D \in F, \\ F \wedge \bar{C} \vDash \sigma(D)\end{array}} \arrow[dr] & \boxed{\begin{array}{l}\forall D \in F, \\ F \wedge C \vDash D\end{array}} \arrow[d] & \\
& \boxed{\begin{array}{l}\forall D \in F, \\ F \vDash \nabla(\sigma \coloneq \bar{C}).\, D\end{array}} &
\end{tikzcd}$$

Interference is inference (in mutation logic)

$$I \vDash \nabla(\sigma :- Q).\, F \quad \text{iff} \quad I \circ (\sigma :- Q) \vDash F$$

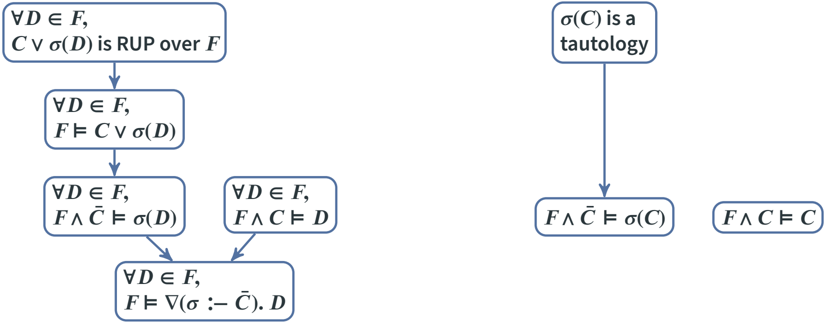
Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :- \bar{C}).F \wedge C$



Interference is inference (in mutation logic)

$$I \vDash \nabla(\sigma :\!- Q).\, F \quad \text{iff} \quad I \circ (\sigma :\!- Q) \vDash F$$

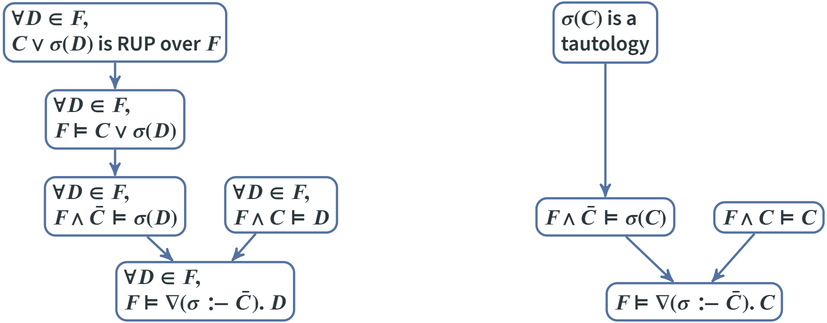
Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :\!- \bar{C}).F \wedge C$



Interference is inference (in mutation logic)

$$I \vDash \nabla(\sigma :\!\!- Q).\, F \quad \text{iff} \quad I \circ (\sigma :\!\!- Q) \vDash F$$

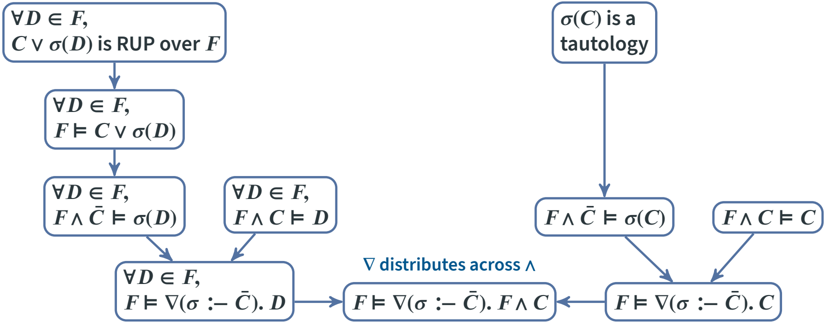
Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :\!\!- \bar{C}).F \wedge C$



Interference is inference (in mutation logic)

$$I \vDash \nabla(\sigma :\!- Q).\, F \quad \text{iff} \quad I \circ (\sigma :\!- Q) \vDash F$$

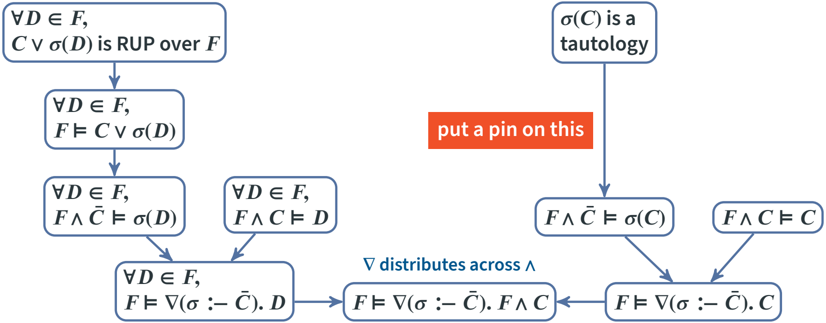
Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :\!- \bar{C}).F \land C$



Interference is inference (in mutation logic)

$$I \vDash \nabla(\sigma :\!-\, Q).\, F \quad \text{iff} \quad I \circ (\sigma :\!-\, Q) \vDash F$$

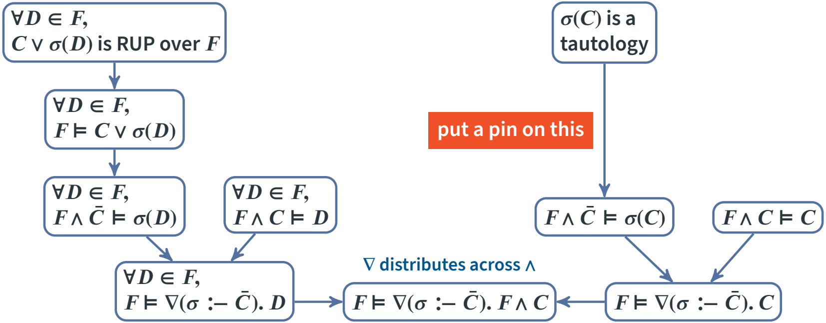
Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :\!-\, \bar{C}).F \wedge C$



Interference is inference (in mutation logic)

$$I \vDash \nabla(\sigma :\!- Q).\, F \quad \text{iff} \quad I \circ (\sigma :\!- Q) \vDash F$$

Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :\!- \bar{C}).F \wedge C$

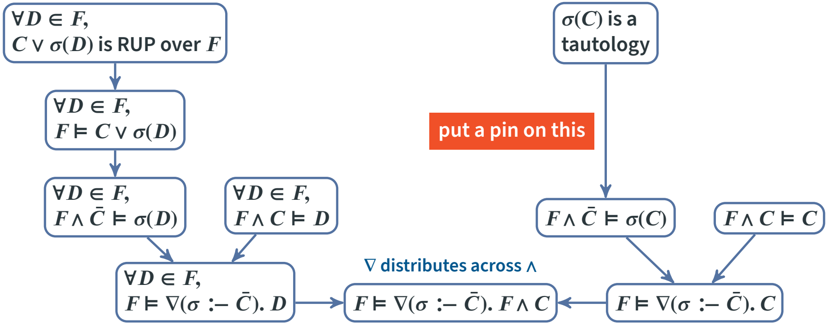


Interference is the same as deriving $\nabla(\sigma :\!- \bar{C}).D$ for each $D \in F \wedge C$

Interference is inference (in mutation logic)

$$I \vDash \nabla(\sigma :\!- Q).\, F \quad \text{iff} \quad I \circ (\sigma :\!- Q) \vDash F$$

Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :\!- \bar{C}).F \wedge C$

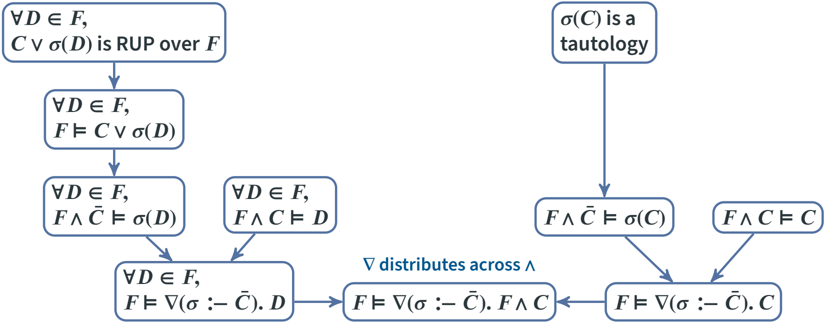


Interference is the same as deriving $\nabla(\sigma :\!- \bar{C}).D$ for each $D \in F \wedge C$

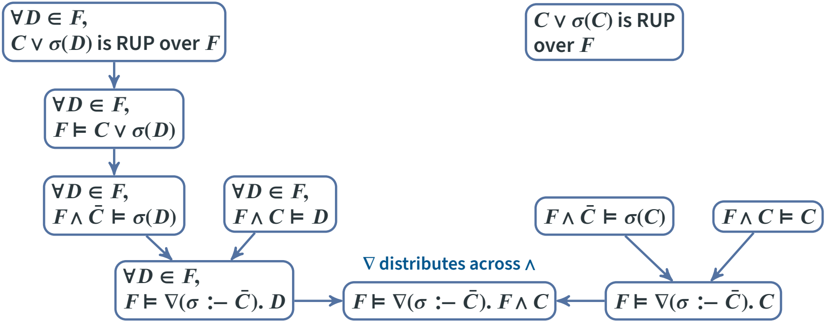
In the paper lots of details about mutation logic
+ a DAG-shaped proof system for interference!

I brought you some souvenirs from mutation world

Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :- \bar{C}).F \wedge C$

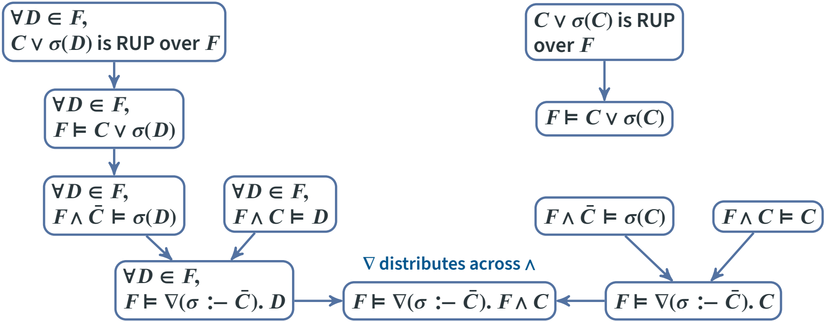


Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :- \bar{C}).F \wedge C$



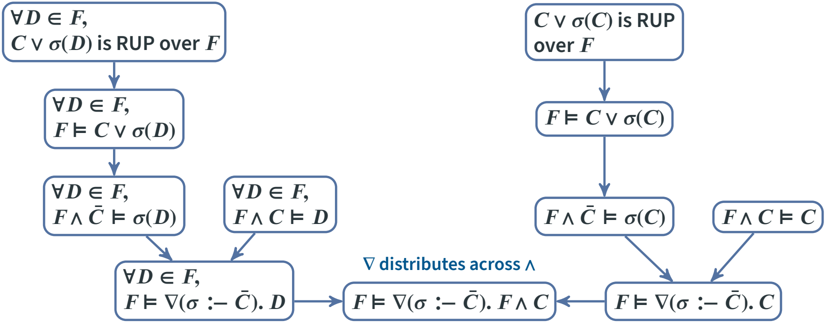
I brought you some souvenirs from mutation world

Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :- \bar{C}).F \wedge C$



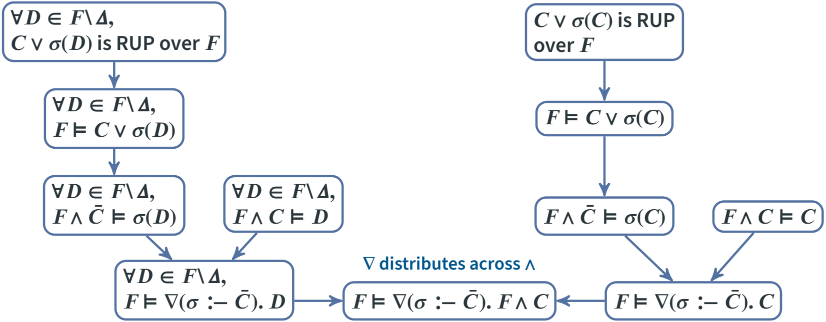
I brought you some souvenirs from mutation world

Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :- \bar{C}).F \wedge C$



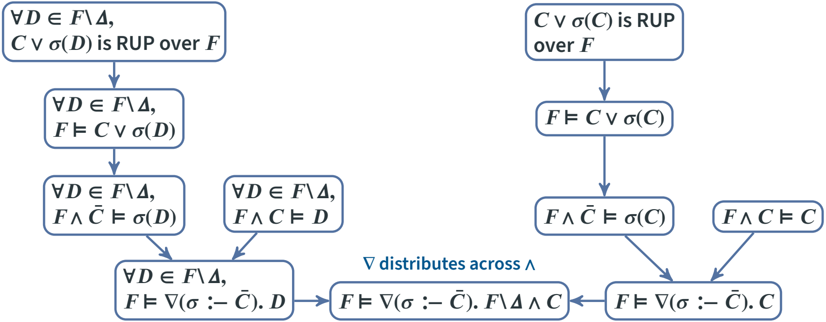
I brought you some souvenirs from mutation world

Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :- \bar{C}).F \wedge C$

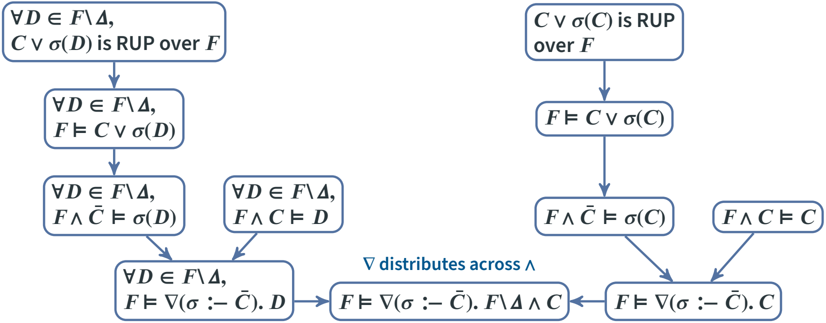


I brought you some souvenirs from mutation world

Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :- \bar{C}).F \wedge C$



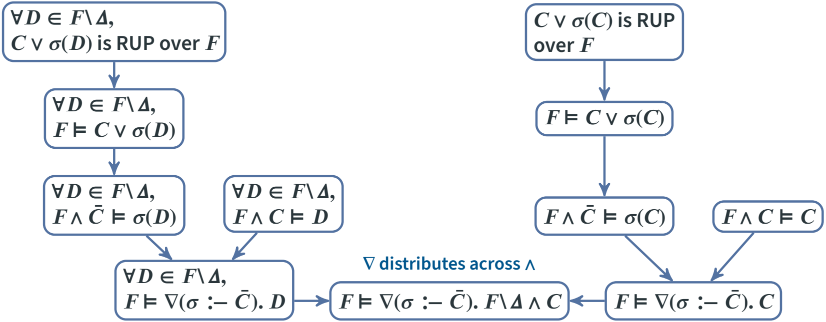
Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :\!-\, \bar{C}).F \wedge C$



A clause C is a **weak substitution redundancy (WSR)** clause upon σ over F modulo Δ if, for all clauses $D \in F \setminus \Delta \wedge C$, the clause $C \vee \sigma(D)$ is a RUP clause over F.

I brought you some souvenirs from mation world

Theorem if C is an SR clause over F upon σ, then $F \vDash \nabla(\sigma :- \bar{C}).F \wedge C$

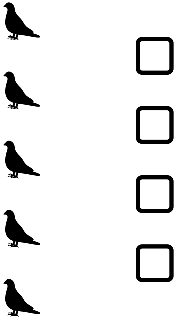


A clause C is a **weak substitution redundancy (WSR)** clause upon σ over F modulo Δ if, for all clauses $D \in F \setminus \Delta \wedge C$, the clause $C \vee \sigma(D)$ is a RUP clause over F.

Theorem if C is a WSR clause over F modulo Δ upon σ, then
$F \vDash \nabla(\sigma :- \bar{C}).F \setminus \Delta \wedge C$

Swapping pigeons, finally

The pigeonhole problem $\mathrm{PHP}(n)$

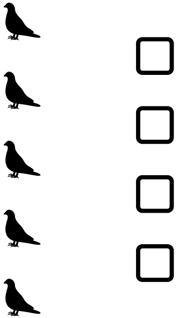


Can we fit n pigeons into $n-1$ holes?

Swapping pigeons, finally

The pigeonhole problem $\mathrm{PHP}(n)$

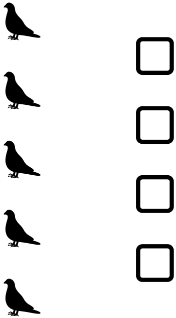
Can we fit n pigeons into $n-1$ holes?



p_{ir} pigeon i is in hole r

Swapping pigeons, finally

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

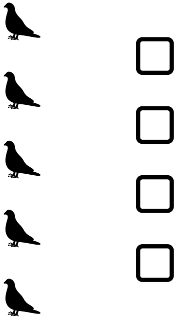
p_{ir} pigeon i is in hole r

$p_{i1} \vee \cdots \vee p_{i(n-1)}$ for $1 \leq i \leq n$

$\overline{p_{ir}} \vee \overline{p_{jr}}$ for $1 \leq i < j \leq n$ and $1 \leq r < n$

Swapping pigeons, finally

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$

  otherwise swap pigeons 1 and n

  $\vdots$
- w.l.o.g. pigeon $n-1$ is not in hole $n-1$
- pigeon 1 is in some hole $1, \ldots, n-2$

  $\vdots$
- pigeon $n-1$ is in some hole $1, \ldots, n-2$
- solve $\mathrm{PHP}(n-1)$

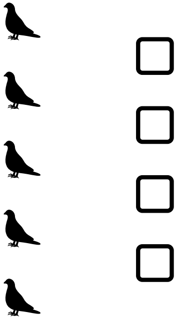
p_{ir} pigeon i is in hole r

$p_{i1} \vee \cdots \vee p_{i(n-1)}$ for $1 \leq i \leq n$

$\overline{p_{ir}} \vee \overline{p_{jr}}$ for $1 \leq i < j \leq n$ and $1 \leq r < n$

Swapping pigeons, finally

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$
  otherwise swap pigeons 1 and n

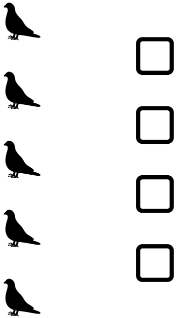
p_{ir} pigeon i is in hole r

$p_{i1} \vee \cdots \vee p_{i(n-1)}$ for $1 \le i \le n$

$\overline{p_{ir}} \vee \overline{p_{jr}}$ for $1 \le i < j \le n$ and $1 \le r < n$

Swapping pigeons, finally

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$

  otherwise swap pigeons 1 and n

  introduce $C = \overline{p_{1(n-1)}}$ as SR clause upon σ

  $\sigma = \{p_{1r} \mapsto p_{nr},\ p_{nr} \mapsto p_{1r} \ :\ 1 \leq r < n\}$

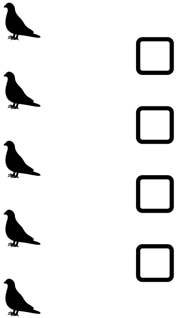
p_{ir} pigeon i is in hole r

$p_{i1} \vee \cdots \vee p_{i(n-1)}$ for $1 \leq i \leq n$

$\overline{p_{ir}} \vee \overline{p_{jr}}$ for $1 \leq i < j \leq n$ and $1 \leq r < n$

Swapping pigeons, finally

The pigeonhole problem $\mathrm{PHP}(n)$



p_{ir} pigeon i is in hole r

$p_{i1} \vee \cdots \vee p_{i(n-1)}$ for $1 \leq i \leq n$

$\overline{p_{ir}} \vee \overline{p_{jr}}$ for $1 \leq i < j \leq n$ and $1 \leq r < n$

Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$

  otherwise swap pigeons 1 and n

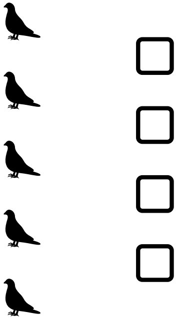
  introduce $C = \overline{p_{1(n-1)}}$ as SR clause upon σ

  $\sigma = \{p_{1r} \mapsto p_{nr}, p_{nr} \mapsto p_{1r} : 1 \leq r < n\}$

  $C \vee \sigma(D)$ is RUP for each clause D!

Swapping pigeons, finally

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$

  otherwise swap pigeons 1 and n

  introduce $C = \overline{p_{1(n-1)}}$ as SR clause upon σ

  $\sigma = \{p_{1r} \mapsto p_{nr}, p_{nr} \mapsto p_{1r} : 1 \leq r < n\}$

  $C \vee \sigma(D)$ is RUP for each clause D!

  but $\sigma(C)$ is not a tautology...

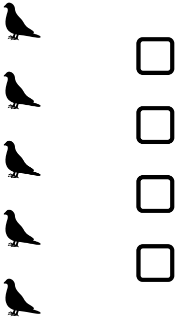
p_{ir} pigeon i is in hole r

$p_{i1} \vee \cdots \vee p_{i(n-1)}$ for $1 \leq i \leq n$

$\overline{p_{ir}} \vee \overline{p_{jr}}$ for $1 \leq i < j \leq n$ and $1 \leq r < n$

Swapping pigeons, finally

The pigeonhole problem $\mathrm{PHP}(n)$



p_{ir} pigeon i is in hole r

$p_{i1} \lor \cdots \lor p_{i(n-1)}$ for $1 \le i \le n$

$\overline{p_{ir}} \lor \overline{p_{jr}}$ for $1 \le i < j \le n$ and $1 \le r < n$

Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$

  otherwise swap pigeons 1 and n

  introduce $C = \overline{p_{1(n-1)}}$ as SR clause upon σ

  $\sigma = \{p_{1r} \mapsto p_{nr}, p_{nr} \mapsto p_{1r} : 1 \le r < n\}$

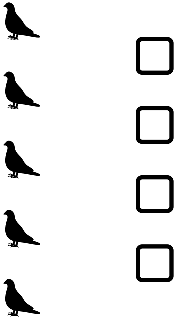
  $C \lor \sigma(D)$ is RUP for each clause D!

  but $\sigma(C)$ is not a tautology...

  ... but it suffices that $C \lor \sigma(C)$ is a RUP

Swapping pigeons, finally

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$
  otherwise swap pigeons 1 and n

  introduce $C = \overline{p_{1(n-1)}}$ as WSR clause upon σ

  $\sigma = \{p_{1r} \mapsto p_{nr},\ p_{nr} \mapsto p_{1r}\ :\ 1 \leq r < n\}$

  $C \vee \sigma(D)$ is RUP for each clause D!

  but $\sigma(C)$ is not a tautology...

  ... but it suffices that $C \vee \sigma(C)$ is a RUP

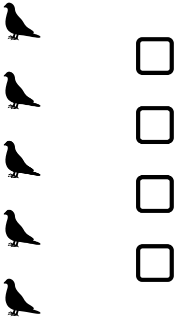
p_{ir} pigeon i is in hole r

$p_{i1} \vee \cdots \vee p_{i(n-1)}$ for $1 \leq i \leq n$

$\overline{p_{ir}} \vee \overline{p_{jr}}$ for $1 \leq i < j \leq n$ and $1 \leq r < n$

Swapping pigeons, finally

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$

  otherwise swap pigeons 1 and n

  introduce $C = \overline{p_{1(n-1)}}$ as WSR clause upon σ

  $\sigma = \{p_{1r} \mapsto p_{nr},\ p_{nr} \mapsto p_{1r} : 1 \le r < n\}$

  $C \vee \sigma(D)$ is RUP for each clause D!

  $C \vee \sigma(C) = \overline{p_{1(n-1)}} \vee \overline{p_{n(n-1)}}$ is RUP!

  … but it suffices that $C \vee \sigma(C)$ is a RUP

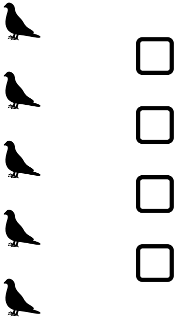
p_{ir} pigeon i is in hole r

$p_{i1} \vee \cdots \vee p_{i(n-1)}$ for $1 \le i \le n$

$\overline{p_{ir}} \vee \overline{p_{jr}}$ for $1 \le i < j \le n$ and $1 \le r < n$

Swapping pigeons, finally

The pigeonhole problem $\mathrm{PHP}(n)$



Can we fit n pigeons into $n-1$ holes?

- w.l.o.g. pigeon 1 is not in hole $n-1$

  otherwise swap pigeons 1 and n

  introduce $C = \overline{p_{1(n-1)}}$ as WSR clause upon σ

  $\sigma = \{p_{1r} \mapsto p_{nr}, p_{nr} \mapsto p_{1r} : 1 \leq r < n\}$

  $\vee\ \sigma(D)$ is RUP for each clause D!

  $(C) = \overline{p_{1(n-1)}} \vee \overline{p_{n(n-1)}}$ is RUP!

  … b uffices that $C \vee \sigma(C)$ is a RUP

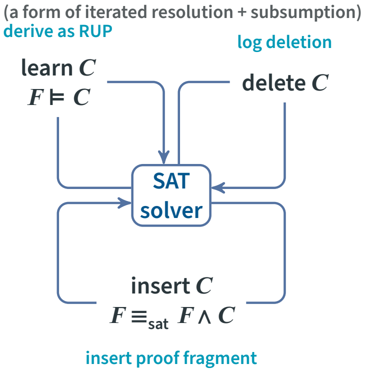
solved! (details in the paper)

p_{ir} pigeon i is in hole r

$p_{i1} \vee \cdots \vee p_{i(n-1)}$ for $1 \leq i \leq n$

$\overline{p_{ir}} \vee \overline{p_{jr}}$ for $1 \leq i < j \leq n$ and $1 \leq r < n$

Interference-free lemmas, finally



Proof generation for inprocessing

i: L_1
i: L_2
i: L_3
i: $C\ [\sigma]$
d: L_3
d: L_2
d: L_1

all clauses in $\sigma(F)$ are RUP clauses over $F \wedge L_1 \wedge L_2 \wedge L_3$

if $F|_{\bar{C}} \vDash \sigma(F)$

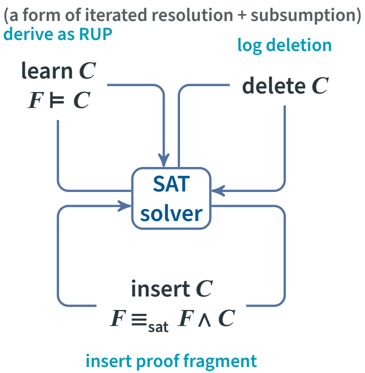
C is an SR clause if $\sigma(C)$ is a tautology and all clauses in $\sigma(F)$ are RUP clauses over $F|_{\bar{C}}$

we also need $\sigma(L_1), \sigma(L_2), \sigma(L_3)$ to be RUPs over $F \wedge L_1 \wedge L_2 \wedge L_3$

... which might need extra lemmas themselves...

... and so on...

Interference-free lemmas, finally



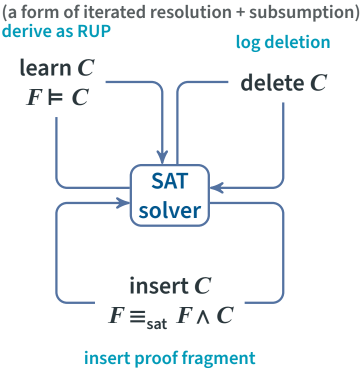
Proof generation for inprocessing

i: L_1
i: L_2
i: L_3
i: C $[\sigma]$
d: L_3
d: L_2
d: L_1

if $F|_{\bar{C}} \vDash \sigma(F)$

C is a WSR clause modulo Δ if
$C \vee \sigma(D)$ is a RUP for each $D \in F \setminus \Delta \wedge C$

Interference-free lemmas, finally



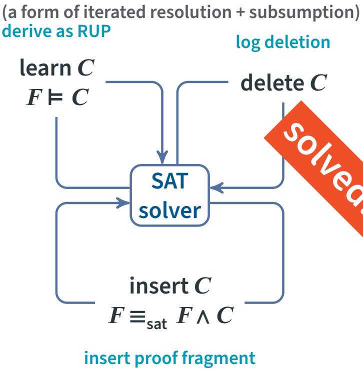
Proof generation for inprocessing

i: L_1
i: L_2
i: L_3
i: C $[\sigma]$ mod L_1, L_2, L_3

if $F|_{\bar{C}} \models \sigma(F)$

C is a WSR clause modulo Δ if
$C \vee \sigma(D)$ is a RUP for each $D \in F \setminus \Delta \wedge C$

Interference-free lemmas, finally



Proof generation for inprocessing

i: L_1
i: L_2
i: L_3
i: $C\ [\sigma]$ mod L_1, L_2, L_3

solved! (details in the paper)

$\ldots F|_{\bar{C}} \models \sigma(F)$

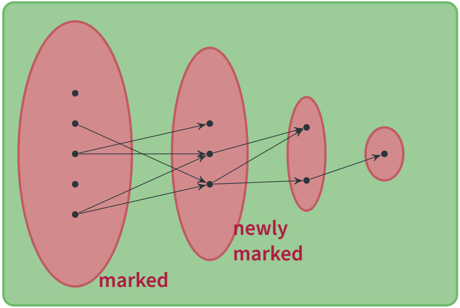
C is a W… use modulo Δ if
$C \vee \sigma(D)$ i… P for each $D \in F \setminus \Delta \wedge C$

Cores and trimming under interference, finally

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clause, it is in the core
- if C is a RUP clause, mark its antecedents
- if C is an SR clause upon σ...?



can we do better than this fixpoint computation?

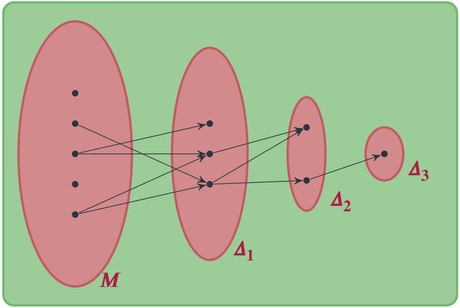
for each marked D, $\sigma(D)$ is a RUP clause over $F|_{\bar{C}}$ $\Rightarrow$ mark their antecedents

Cores and trimming under interference, finally

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clause, it is in the core
- if C is a RUP clause, mark its antecedents
- if C is an SR clause upon σ...?



can we do better than this fixpoint computation?

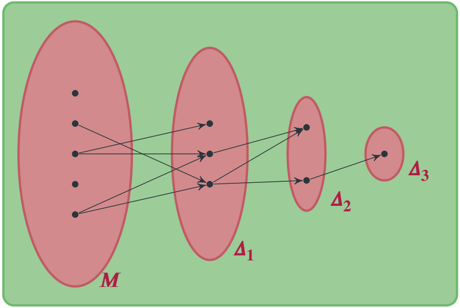
for each marked D, $\sigma(D)$ is a RUP clause over $F|_{\bar{C}}$ $\Rightarrow$ mark their antecedents

Cores and trimming under interference, finally

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clause, it is in the core
- if C is a RUP clause, mark its antecedents
- if C is an SR clause upon σ...?



can we do better than this fixpoint computation?

C is a WSR over $F \wedge \Delta_1$ modulo Δ_1 upon σ!

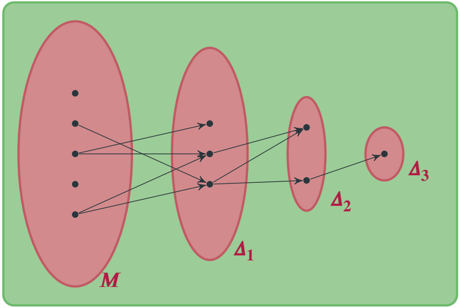
for each marked D, $\sigma(D)$ is a RUP clause over $F|_{\bar{C}}$ $\Rightarrow$ mark their antecedents

Cores and trimming under interference, finally

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clause, it is in the core
- if C is a RUP clause, mark its antecedents
- if C is an SR clause upon σ...?



can we do better than this fixpoint computation?

C is a WSR over $F \land \Delta_1$ modulo Δ_1 upon σ!

TL;DR just mark Δ_1

for each marked D, $\sigma(D)$ is a RUP clause over $F|_{\bar{C}}$ $\Rightarrow$ mark their antecedents

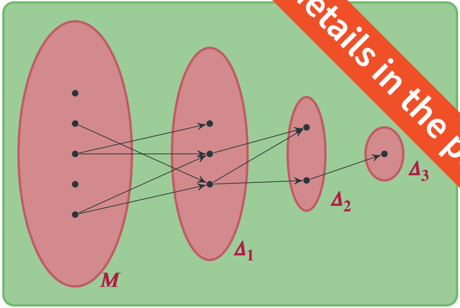
Cores and trimming under interference, finally

Generating an unsatisfiable core from a proof

mark the empty clause and proceed backwards

- if C is not marked, skip it
- if C is an input clau… it is in the core
- if C is a RUP clause, … its antecedents
- if C is an SR clause upo…

solved! (details in the paper)



can we do better than this fixpoint computation?

C is a WSR over $F \wedge \Delta_1$ modulo Δ_1 upon σ!

TL;DR just mark Δ_1

for each marked D, $\sigma(D)$ is a RUP clause over $F|_{\bar{C}}$ $\Rightarrow$ mark their antecedents

Takeaways and future directions

Existing interference rules are unnecessarily restrictive, and those restrictions can be removed at negligible cost.

Takeaways and future directions

Existing interference rules are unnecessarily restrictive, and those restrictions can be removed at negligible cost.

Doing so enables shorter, intuitive proofs.

Takeaways and future directions

Existing interference rules are unnecessarily restrictive, and those restrictions can be removed at negligible cost.

Doing so enables shorter, intuitive proofs.

Interference-free lemmas are possible for free.

Takeaways and future directions

Existing interference rules are unnecessarily restrictive, and those restrictions can be removed at negligible cost.

Doing so enables shorter, intuitive proofs.

Interference-free lemmas are possible for free.

There is no need to perform fixpoint core generation, even if not using WSR.

Takeaways and future directions

Existing interference rules are unnecessarily restrictive, and those restrictions can be removed at negligible cost.

Doing so enables shorter, intuitive proofs.

Interference-free lemmas are possible for free.

There is no need to perform fixpoint core generation, even if not using WSR.

There is a clear, mutation-based semantics for interference.

Takeaways and future directions

Existing interference rules are unnecessarily restrictive, and those restrictions can be removed at negligible cost.

Doing so enables shorter, intuitive proofs.

Interference-free lemmas are possible for free.

There is no need to perform fixpoint core generation, even if not using WSR.

There is a clear, mutation-based semantics for interference.

Future directions

Developing a proof format and proof checker with state-of-the-art efficiency.

Takeaways and future directions

Existing interference rules are unnecessarily restrictive, and those restrictions can be removed at negligible cost.

Doing so enables shorter, intuitive proofs.

Interference-free lemmas are possible for free.

There is no need to perform fixpoint core generation, even if not using WSR.

There is a clear, mutation-based semantics for interference.

Future directions

Developing a proof format and proof checker with state-of-the-art efficiency.

Can we improve on SDCL-like methods with WSR?

Takeaways and future directions

Existing interference rules are unnecessarily restrictive, and those restrictions can be removed at negligible cost.

Doing so enables shorter, intuitive proofs.

Interference-free lemmas are possible for free.

There is no need to perform fixpoint core generation, even if not using WSR.

There is a clear, mutation-based semantics for interference.

Future directions

- Developing a proof format and proof checker with state-of-the-art efficiency.
- Can we improve on SDCL-like methods with WSR?
- Can we use the mutation framework for QBF interference?

Takeaways and future directions

Existing interference rules are unnecessarily restrictive, and those restrictions can be removed at negligible cost.

Doing so enables shorter, intuitive proofs.

Interference-free lemmas are possible for free.

There is no need to perform fixpoint core generation, even if not using WSR.

There is a clear, mutation-based semantics for interference.

Future directions

Developing a proof format and proof checker with state-of-the-art efficiency.

Can we improve on SDCL-like methods with WSR?

Can we use the mutation framework for QBF interference?

... and maybe, potentially, perhaps, possibly, SMT/FOL?