TECHNISCHE UNIVERSITÄT WIEN

logics Logical Methods in Computer Science

Microsoft

# Frying the Egg, Roasting the Chicken
# Unit Deletions in DRAT Proofs


Johannes Altmanninger, Adrián Rebola-Pardo

TU Wien

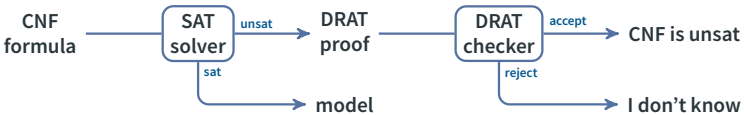


CPP 2020
New Orleans, USA
January 20th, 2020

Supported by FWF W1255-N23, WWTF VRG11-005 and Microsoft Research PhD Programme


Some pictures by Freepik from Flaticon www.flaticon.com

# SAT solving and DRAT proofs



# SAT solving and DRAT proofs


**DRAT proof** sequence of clause introductions and deletions

**i:** $C$ $C$ must be a RAT clause w.r.t. $F$
**d:** $C$ always allowed

# SAT solving and DRAT proofs


**DRAT proof** sequence of clause introductions and deletions

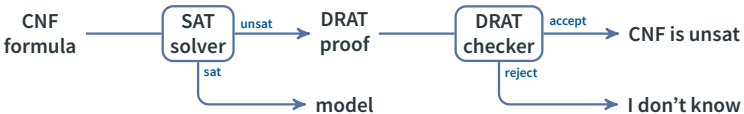
i: $C$ $C$ must be a RAT clause w.r.t. $F$
d: $C$ always allowed

# SAT solving and DRAT proofs


**DRAT proof** sequence of clause introductions and deletions

i: $C$ $C$ must be a RAT clause w.r.t. $F$
d: $C$ always allowed

# SAT solving and DRAT proofs

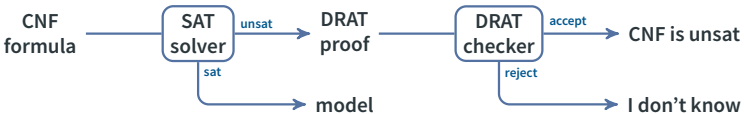



**DRAT proof** sequence of clause introductions and deletions

**i:** $C$ $C$ must be a RAT clause w.r.t. $F$
**d:** $C$ always allowed

- RAT introduction and clause deletion are satisfiability-preserving
  *Deriving* $\bot$ *proves unsatisfiability*

# SAT solving and DRAT proofs




**DRAT proof** sequence of clause introductions and deletions

**i:** $C$ $C$ must be a RAT clause w.r.t. $F$
**d:** $C$ always allowed

- RAT introduction and clause deletion are satisfiability-preserving
  *Deriving $\bot$ proves unsatisfiability*
- RAT introduction is non-monotonic
  *Clause deletion may enable new RAT inferences*

# Two flavors of DRAT

**Unit clauses** $C$ **is unit w.r.t.** $F$ **if all literals except one in** $C$ **are implied by unit propagation from** $F$

# Two flavors of DRAT

**Unit clauses** $C$ **is unit w.r.t.** $F$ **if all literals except one in** $C$ **are implied by unit propagation from** $F$

**Two different specifications of a correct proof exist**

# Two flavors of DRAT

**Unit clauses** $C$ **is unit w.r.t.** $F$ **if all literals except one in** $C$ **are implied by unit propagation from** $F$

**Two different specifications of a correct proof exist**

**specified DRAT**

[Wetzler, Heule, Hunt '14]

**operational DRAT**

`drat-trim` [Heule, Wetzler '16]
`gratgen` [Lammich '17]

# Two flavors of DRAT

**Unit clauses** $C$ **is unit w.r.t.** $F$ **if all literals except one in** $C$ **are implied by unit propagation from** $F$

**Two different specifications of a correct proof exist**

**specified DRAT**

[Wetzler, Heule, Hunt '14]

**all deletions are applied**

**operational DRAT**

`drat-trim` [Heule, Wetzler '16]
`gratgen` [Lammich '17]

**unit deletions are ignored**

# Two flavors of DRAT

**Unit clauses** $C$ **is unit w.r.t.** $F$ **if all literals except one in** $C$ **are implied by unit propagation from** $F$

**Two different specifications of a correct proof exist**

**specified DRAT**

[Wetzler, Heule, Hunt '14]

**all deletions are applied**

**sound proof system**

**operational DRAT**

`drat-trim` [Heule, Wetzler '16]

`gratgen` [Lammich '17]

**unit deletions are ignored**

**sound proof system**

# Two flavors of DRAT

**Unit clauses** $C$ is unit w.r.t. $F$ if all literals except one in $C$ are implied by unit propagation from $F$

Two different specifications of a correct proof exist

**specified DRAT**

[Wetzler, Heule, Hunt '14]
all deletions are applied
sound proof system

**operational DRAT**

`drat-trim` [Heule, Wetzler '16]
`gratgen` [Lammich '17]
unit deletions are ignored
sound proof system

**Discrepancies exist**

- Short correct proofs in one flavor, incorrect in the other [Rebola, Biere '18]
- About 60% of proofs generated in SAT competitions [Rebola, Cruz '18]

# The egg and the chicken

There are several practical issues with operational DRAT [Rebola, Biere '18]

# The egg and the chicken

There are several practical issues with operational DRAT [Rebola, Biere '18]

In previous episodes... [Rebola, Cruz '18]

# The egg and the chicken

There are several practical issues with operational DRAT [Rebola, Biere '18]

In previous episodes... [Rebola, Cruz '18]

A method for (relatively) efficient specified DRAT checking

*... but nowhere close to operational DRAT checkers*

# The egg and the chicken

There are several practical issues with operational DRAT [Rebola, Biere '18]

In previous episodes... [Rebola, Cruz '18]

- A method for (relatively) efficient specified DRAT checking
  - *... but nowhere close to operational DRAT checkers*
- About 60% of proofs generated in SAT competitions are discrepant
  - *... but was there a bug in our checker?*

# The egg and the chicken

There are several practical issues with operational DRAT [Rebola, Biere '18]

In previous episodes... [Rebola, Cruz '18]

A method for (relatively) efficient specified DRAT checking

*... but nowhere close to operational DRAT checkers*

About 60% of proofs generated in SAT competitions are discrepant

*... but was there a bug in our checker?*

Why do we keep using operational DRAT?

# The egg and the chicken

There are several practical issues with operational DRAT [Rebola, Biere '18]

In previous episodes... [Rebola, Cruz '18]

- A method for (relatively) efficient specified DRAT checking
  - *... but nowhere close to operational DRAT checkers*
- About 60% of proofs generated in SAT competitions are discrepant
  - *... but was there a bug in our checker?*

Why do we keep using operational DRAT?

most SAT solvers generate operational DRAT proofs

# The egg and the chicken

There are several practical issues with operational DRAT [Rebola, Biere '18]

In previous episodes... [Rebola, Cruz '18]

A method for (relatively) efficient specified DRAT checking

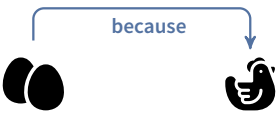
*... but nowhere close to operational DRAT checkers*

About 60% of proofs generated in SAT competitions are discrepant

*... but was there a bug in our checker?*

Why do we keep using operational DRAT?

most SAT solvers generate operational DRAT proofs




all efficient checkers check operational DRAT proofs

# The egg and the chicken

There are several practical issues with operational DRAT [Rebola, Biere '18]

In previous episodes... [Rebola, Cruz '18]

A method for (relatively) efficient specified DRAT checking

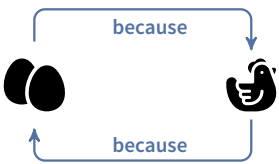
*... but nowhere close to operational DRAT checkers*

About 60% of proofs generated in SAT competitions are discrepant

*... but was there a bug in our checker?*

Why do we keep using operational DRAT?

most SAT solvers generate operational DRAT proofs




all efficient checkers check operational DRAT proofs

# The egg and the chicken

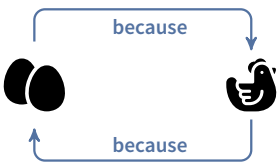
There are several practical issues with operational DRAT [Rebola, Biere '18]

In previous episodes... [Rebola, Cruz '18]

- A method for (relatively) efficient specified DRAT checking
  - *... but nowhere close to operational DRAT checkers*
- About 60% of proofs generated in SAT competitions are discrepant
  - *... but was there a bug in our checker?*

Why do we keep using operational DRAT?

most SAT solvers generate operational DRAT proofs




all efficient checkers check operational DRAT proofs

In this paper...

# The egg and the chicken

There are several practical issues with operational DRAT [Rebola, Biere '18]

In previous episodes... [Rebola, Cruz '18]

A method for (relatively) efficient specified DRAT checking

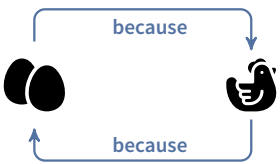
*... but nowhere close to operational DRAT checkers*

About 60% of proofs generated in SAT competitions are discrepant

*... but was there a bug in our checker?*

Why do we keep using operational DRAT?

most SAT solvers generate operational DRAT proofs




all efficient checkers check operational DRAT proofs

In this paper...

patching SAT solvers
so that spurious deletions
are avoided

# The egg and the chicken

There are several practical issues with operational DRAT [Rebola, Biere '18]

In previous episodes... [Rebola, Cruz '18]

A method for (relatively) efficient specified DRAT checking

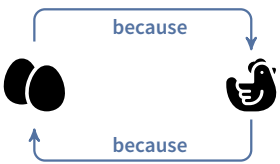
*... but nowhere close to operational DRAT checkers*

About 60% of proofs generated in SAT competitions are discrepant

*... but was there a bug in our checker?*

Why do we keep using operational DRAT?

most SAT solvers generate operational DRAT proofs




all efficient checkers check operational DRAT proofs

In this paper...

patching SAT solvers so that spurious deletions are avoided

developing a specified DRAT checker with competitive performance

# The egg and the chicken

There are several practical issues with operational DRAT [Rebola, Biere '18]

In previous episodes... [Rebola, Cruz '18]

A method for (relatively) efficient specified DRAT checking

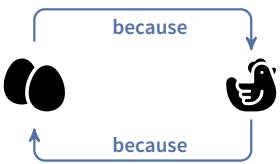
*... but nowhere close to operational DRAT checkers*

About 60% of proofs generated in SAT competitions are discrepant

*... but was there a bug in our checker?*

Why do we keep using operational DRAT?

most SAT solvers generate operational DRAT proofs




all efficient checkers check operational DRAT proofs

In this paper...

patching SAT solvers so that spurious deletions are avoided

developing a specified DRAT checker with competitive performance

certifying rejections for specified DRAT proofs

# The egg: spurious deletions in SAT solvers

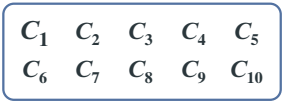
**Database** stores the clauses in the current CNF formula

**Trail** stores the propagated literals from the current CNF formula

# The egg: spurious deletions in SAT solvers

**Database** stores the clauses in the current CNF formula

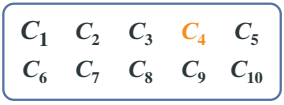
**Trail** stores the propagated literals from the current CNF formula




# The egg: spurious deletions in SAT solvers

**Database** stores the clauses in the current CNF formula

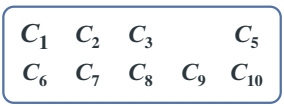
**Trail** stores the propagated literals from the current CNF formula




# The egg: spurious deletions in SAT solvers

**Database** stores the clauses in the current CNF formula

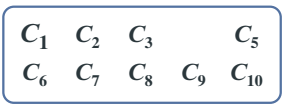
**Trail** stores the propagated literals from the current CNF formula




# The egg: spurious deletions in SAT solvers

**Database** stores the clauses in the current CNF formula

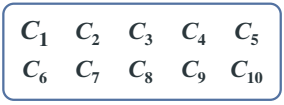
**Trail** stores the propagated literals from the current CNF formula




# The egg: spurious deletions in SAT solvers

**Database** stores the clauses in the current CNF formula

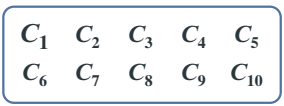
**Trail** stores the propagated literals from the current CNF formula




# The egg: spurious deletions in SAT solvers

**Database** stores the clauses in the current CNF formula

**Trail** stores the propagated literals from the current CNF formula





**Solver::removeSatisfied** `minisat` [Een, Sörensson '03]

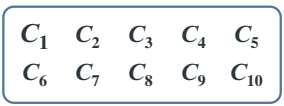
removes clauses in the database which are satisfied by the trail

*but propagated literals are not removed!*

# The egg: spurious deletions in SAT solvers

**Database** stores the clauses in the current CNF formula

**Trail** stores the propagated literals from the current CNF formula





**Solver::removeSatisfied** `minisat` [Een, Sörensson '03]

removes clauses in the database which are satisfied by the trail

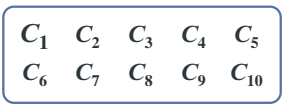
*but propagated literals are not removed!*

*the solver behaves as though the clause was not removed*

# The egg: spurious deletions in SAT solvers

**Database** stores the clauses in the current CNF formula

**Trail** stores the propagated literals from the current CNF formula





**Solver::removeSatisfied** `minisat` [Een, Sörensson '03]

removes clauses in the database which are satisfied by the trail

*but propagated literals are not removed!*

*the solver behaves as though the clause was not removed*

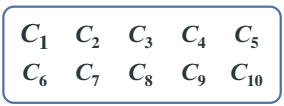
a **unit deletion** instruction d: $C$ is emitted

*... and checkers will ignore the emitted deletion*

# The egg: spurious deletions in SAT solvers

**Database** stores the clauses in the current CNF formula

**Trail** stores the propagated literals from the current CNF formula





**Solver::removeSatisfied** `minisat` [Een, Sörensson '03]

removes clauses in the database which are satisfied by the trail

*but propagated literals are not removed!*

*the solver behaves as though the clause was not removed*

a **unit deletion** instruction d: $C$ is emitted

*... and checkers will ignore the emitted deletion*

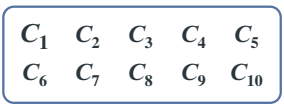
**Frying the egg** do not emit that deletion at all

*this solves the issue for purely CDCL solvers*

# The egg: spurious deletions in SAT solvers

**Database** stores the clauses in the current CNF formula

**Trail** stores the propagated literals from the current CNF formula





**Solver::removeSatisfied** `minisat` [Een, Sörensson '03]

removes clauses in the database which are satisfied by the trail

*but propagated literals are not removed!*

*the solver behaves as though the clause was not removed*

a **unit deletion** instruction d: $C$ is emitted

*... and checkers will ignore the emitted deletion*

**Frying the egg** do not emit that deletion at all

*this solves the issue for purely CDCL solvers*

*inprocessing techniques may still need to delete unit clauses* [Rebola, Biere '18]

# The chicken: competitive specified DRAT checking

**Unit deletions can shrink the set of literals implied by unit propagation**

*in DRAT checkers, this is also called the trail*

# The chicken: competitive specified DRAT checking

**Unit deletions can shrink the set of literals implied by unit propagation**

*in DRAT checkers, this is also called the trail*

**Operational DRAT checkers rely on monotonic trail growth**

`drat-trim` [Heule, Wetzler '16], `gratgen` [Lammich '16]

# The chicken: competitive specified DRAT checking

**Unit deletions can shrink the set of literals implied by unit propagation**

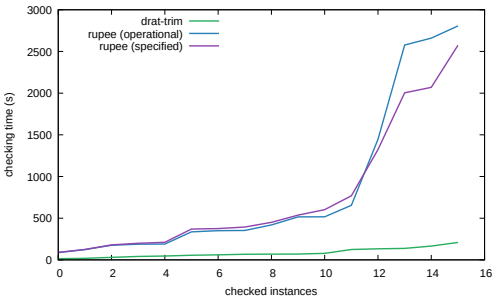
*in DRAT checkers, this is also called the trail*

**Operational DRAT checkers rely on monotonic trail growth**

`drat-trim` [Heule, Wetzler '16], `gratgen` [Lammich '16]

**Specified DRAT checkers perform worse than operational checkers**

`rupee` [Rebola, Cruz '18]



# The chicken: competitive specified DRAT checking

**Unit deletions can shrink the set of literals implied by unit propagation**

*in DRAT checkers, this is also called the trail*

**Operational DRAT checkers rely on monotonic trail growth**

`drat-trim` [Heule, Wetzler '16], `gratgen` [Lammich '16]

**Specified DRAT checkers perform worse than operational checkers**

`rupee` [Rebola, Cruz '18]

**Roasting the chicken implement a specified DRAT checker `rate` with all the state-of-the-art optimizations** `https://github.com/krobelus/rate`

*backwards checking, trail preprocessing, core-first propagation*

# The chicken: competitive specified DRAT checking

**Unit deletions can shrink the set of literals implied by unit propagation**

*in DRAT checkers, this is also called the trail*

**Operational DRAT checkers rely on monotonic trail growth**

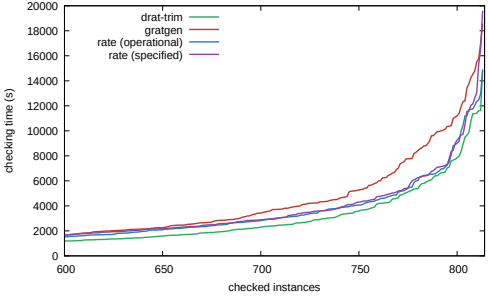
`drat-trim` [Heule, Wetzler '16], `gratgen` [Lammich '16]

**Specified DRAT checkers perform worse than operational checkers**

`rupee` [Rebola, Cruz '18]

**Roasting the chicken implement a specified DRAT checker `rate` with all the state-of-the-art optimizations** `https://github.com/krobelus/rate`

*backwards checking, trail preprocessing, core-first propagation*



# Does it taste good?: certifying rejections

**Incorrect (specified) proofs** **about 77% from the SAT competition 2019**

**How do we know this is not a bug in `rate`?**

# Does it taste good?: certifying rejections

**Incorrect (specified) proofs** **about 77% from the SAT competition 2019**

**How do we know this is not a bug in `rate`?**

*not that simple, because unit propagation is required*

# Does it taste good?: certifying rejections

**Incorrect (specified) proofs** about 77% from the SAT competition 2019

How do we know this is not a **bug** in `rate`?

*not that simple, because unit propagation is required*

**UP semantics** three-valued semantics where clauses are satisfied if either a literal is satisfied or two literals are unassigned [Rebola, Biere '18]

# Does it taste good?: certifying rejections

**Incorrect (specified) proofs** about 77% from the SAT competition 2019

**How do we know this is not a bug in `rate`?**

*not that simple, because unit propagation is required*

**UP semantics** three-valued semantics where clauses are satisfied if either a literal is satisfied or two literals are unassigned [Rebola, Biere '18]

*a clause $C$ is RAT$^*$ w.r.t. a CNF formula $F$ iff $F \wedge \neg C$ is UP-unsatisfiable*

# Does it taste good?: certifying rejections

**Incorrect (specified) proofs** about 77% from the SAT competition 2019

**How do we know this is not a bug in `rate`?**

*not that simple, because unit propagation is required*

**UP semantics** three-valued semantics where clauses are satisfied if either a literal is satisfied or two literals are unassigned [Rebola, Biere '18]

*a clause $C$ is RAT$^*$ w.r.t. a CNF formula $F$ iff $F \wedge \neg C$ is UP-unsatisfiable*

**SICK certificates** give an UP-model for the failed inference

*All rejected proofs are correctly so, in a certified way*

# Conclusions

**The egg** SAT solver introduce spurious, unnecessary deletions

**The chicken** Efficient DRAT checkers deviate from the specification when unit deletions occur in the proof

**The parsley** There is no easy way to check a single failed RAT inference

# Conclusions

**The egg** SAT solver introduce spurious, unnecessary deletions

*just do not do that*

**The chicken** Efficient DRAT checkers deviate from the specification when unit deletions occur in the proof

**The parsley** There is no easy way to check a single failed RAT inference

# Conclusions

**The egg** SAT solver introduce spurious, unnecessary deletions

*just do not do that*

**The chicken** Efficient DRAT checkers deviate from the specification when unit deletions occur in the proof

*implement the methods from* `rupee` *and the optimizations from* `drat-trim`

**The parsley** There is no easy way to check a single failed RAT inference

# Conclusions

**The egg** SAT solver introduce **spurious, unnecessary** deletions

*just do not do that*

**The chicken** Efficient DRAT checkers **deviate** from the specification when unit deletions occur in the proof

*implement the methods from* `rupee` *and the optimizations from* `drat-trim`

**The parsley** There is no easy way to check a single **failed** RAT inference

*certification through three-valued semantics*