Herbrand's Revenge

SAT Solving for First-Order Theorem Proving

DHBW

Stephan Schulz

schulz@eprover.org

Herbrand's Revenge

SAT Solving for First-Order Theorem Proving

...and other news from E

Stephan Schulz

schulz@eprover.org

DHBW

Context: First-Order Theorem Proving

- Theorem proving in first-order logic (with equality)
  - Quantifiers ($\forall, \exists$)
  - Standard connectives ($\neg, \wedge, \vee, \rightarrow, \ldots$)
  - Predicate symbols and function symbols are free
  - Exception: Equality is a congruence relation
- Standard approach: proof by contradiction

$$Ax \models C$$

iff

$$Ax \cup \{\neg C\} \text{ is unsatisfiable}$$

- Clausification turns full FOF into equisatisfiable clause set

Context: First-Order Theorem Proving

- Theorem proving in first-order logic (with equality)
  - Quantifiers ($\forall, \exists$)
  - Standard connectives ($\neg, \wedge, \vee, \rightarrow, \ldots$)
  - Predicate symbols and function symbols are free
  - Exception: Equality is a congruence relation
- Standard approach: proof by contradiction

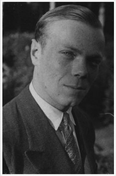
$$Ax \models C$$
$$\text{iff}$$
$$Ax \cup \{\neg C\} \text{ is unsatisfiable}$$

- Clausification turns full FOF into equisatisfiable clause set

Theorem proving is reduced to showing inconsistency of clause sets!

Herbrand's Theorem

Herbrand's Theorem (modern version)



"A set of first-order clauses is unsatisfiable, if and only if it has a finite set of ground instances that is propositionally unsatisfiable."

- If there is a model, there is a Herbrand model
  - Universe consists of ground terms
  - Function symbols are interpreted as constructors
  - Extended to equational logic (Herbrand equality model)
- Contraposition: If there is no ground term model, there is no model
  - Theoretical foundation of most first-order calculi
  - Practical application?

Example

Consider the following set C of clauses:

1. $p(a)$
2. $\neg p(X) \vee p(f(X))$
3. $\neg p(f(Y))$

Example

Consider the following set C of clauses:

1. $p(a)$
2. $\neg p(X) \vee p(f(X))$
3. $\neg p(f(Y))$

C' is a set of ground instances of clauses from C:

1. $p(a)$
2. $\neg p(a) \vee p(f(a))$
3. $\neg p(f(a))$

Example

Consider the following set C of clauses:

1. $p(a)$
2. $\neg p(X) \vee p(f(X))$
3. $\neg p(f(Y))$

C' is a set of ground instances of clauses from C:

1. $p(a)$
2. $\neg p(a) \vee p(f(a))$
3. $\neg p(f(a))$

C' is propositionally unsatisfiable, hence C is unsatisfiable

Enumerate and Check

- Davis&Putnam 1960: Direct application of Herbrand's theorem
  - Enumerate ground instances
  - Periodically check ground clause set via a specialised form of ground resolution
  - *A Computing Procedure for Quantification Theory*
- Theoretically sound and complete, but little practical success
  - Resolution is not very strong on propositional logic
  - Uncontrolled enumeration generates too many irrelevant instances

A Split in the Road

- Davis/Logemann/Loveland (1962): splitting and unit propagation
  - Search for propositional models
  - Propagate atom values forced by unit clauses
  - If no units, case distinction by splitting
  - Backtracking on fail
  - CDCL: DPLL+clause learning+non-chronological backtracking

A Split in the Road

- Davis/Logemann/Loveland (1962): splitting and unit propagation
  - Search for propositional models
  - Propagate atom values forced by unit clauses
  - If no units, case distinction by splitting
  - Backtracking on fail
  - CDCL: DPLL+clause learning+non-chronological backtracking

Modern CDCL solvers are unreasonably successful in practice

A Split in the Road

- Davis/Logemann/Loveland (1962): splitting and unit propagation
  - Search for propositional models
  - Propagate atom values forced by unit clauses
  - If no units, case distinction by splitting
  - Backtracking on fail
  - CDCL: DPLL+clause learning+non-chronological backtracking

Modern CDCL solvers are unreasonably successful in practice

- Robinson (1965): Generate instances via unification
  - Instantiation only to make conflicting constraints explicit (most general *unifier*)
  - Only instantiate as lightly as possible (*most general* unifier)
  - Integrated into *generating inferences*
  - Saturation/Proof completed by derivation of *empty clause*

A Split in the Road

- Davis/Logemann/Loveland (1962): splitting and unit propagation
  - Search for propositional models
  - Propagate atom values forced by unit clauses
  - If no units, case distinction by splitting
  - Backtracking on fail
  - CDCL: DPLL+clause learning+non-chronological backtracking

Modern CDCL solvers are unreasonably successful in practice

- Robinson (1965): Generate instances via unification
  - Instantiation only to make conflicting constraints explicit (most general *unifier*)
  - Only instantiate as lightly as possible (*most general* unifier)
  - Integrated into *generating inferences*
  - Saturation/Proof completed by derivation of *empty clause*

Unification/Saturation: Foundation of most state-of-the-art FO-provers

DPLL and Resolution

DPLL on C':

1. $p(a)$
2. $\neg p(a) \vee p(f(a))$
3. $\neg p(f(a))$

DPLL and Resolution

DPLL on C':

1. $p(a)$
2. $\neg p(a) \vee p(f(a))$
3. $\neg p(f(a))$
4. Propagate 1: $p(f(a))$ (from 2)
5. Propagate 4: $\square$ (from 3)

DPLL and Resolution

DPLL on C':

1. $p(a)$
2. $\neg p(a) \vee p(f(a))$
3. $\neg p(f(a))$
4. Propagate 1: $p(f(a))$ (from 2)
5. Propagate 4: $\square$ (from 3)

No decision/split, hence no backtracking: C' is unsatisfiable

But:
Instantiations provided externally!

DPLL and Resolution

DPLL on C':

1. $p(a)$
2. $\neg p(a) \vee p(f(a))$
3. $\neg p(f(a))$
4. Propagate 1: $p(f(a))$ (from 2)
5. Propagate 4: $\Box$ (from 3)

No decision/split, hence no backtracking: C' is unsatisfiable

But:
Instantiations provided externally!

DPLL and Resolution

DPLL on C':

1. $p(a)$
2. $\neg p(a) \vee p(f(a))$
3. $\neg p(f(a))$
4. Propagate 1: $p(f(a))$ (from 2)
5. Propagate 4: $\square$ (from 3)

No decision/split, hence no backtracking: C' is unsatisfiable

But:
Instantiations provided externally!

Resolution on C:

1. $p(a)$
2. $\neg p(X) \vee p(f(X))$
3. $\neg p(f(Y))$

DPLL and Resolution

DPLL on C':

1. $p(a)$
2. $\neg p(a) \vee p(f(a))$
3. $\neg p(f(a))$
4. Propagate 1: $p(f(a))$ (from 2)
5. Propagate 4: $\square$ (from 3)

No decision/split, hence no backtracking: C' is unsatisfiable

But:
Instantiations provided externally!

Resolution on C:

1. $p(a)$
2. $\neg p(X) \vee p(f(X))$
3. $\neg p(f(Y))$
4. $p(f(a))$ from 1,2 with $\sigma = \{X \mapsto a\}$
5. $\square$ from 4,3 with $\sigma = \{Y \mapsto a\}$

DPLL and Resolution

DPLL on C':

1. $p(a)$
2. $\neg p(a) \vee p(f(a))$
3. $\neg p(f(a))$
4. Propagate 1: $p(f(a))$ (from 2)
5. Propagate 4: $\Box$ (from 3)

No decision/split, hence no backtracking: C' is unsatisfiable

But:
Instantiations provided externally!

Resolution on C:

1. $p(a)$
2. $\neg p(X) \vee p(f(X))$
3. $\neg p(f(Y))$
4. $p(f(a))$ from 1,2 with $\sigma = \{X \mapsto a\}$
5. $\Box$ from 4,3 with $\sigma = \{Y \mapsto a\}$

Instantiations generated by unification!
What could possibly go wrong?

DPLL and Resolution

DPLL on C':

1. $p(a)$
2. $\neg p(a) \vee p(f(a))$
3. $\neg p(f(a))$
4. Propagate 1: $p(f(a))$ (from 2)
5. Propagate 4: $\square$ (from 3)

No decision/split, hence no backtracking: C' is unsatisfiable

But:
Instantiations provided externally!

Resolution on C:

1. $p(a)$
2. $\neg p(X) \vee p(f(X))$
3. $\neg p(f(Y))$
4. $p(f(a))$ from 1,2 with $\sigma = \{X \mapsto a\}$

DPLL and Resolution

DPLL on C':

1. $p(a)$
2. $\neg p(a) \vee p(f(a))$
3. $\neg p(f(a))$
4. Propagate 1: $p(f(a))$ (from 2)
5. Propagate 4: $\Box$ (from 3)

No decision/split, hence no backtracking: C' is unsatisfiable

But:
Instantiations provided externally!

Resolution on C:

1. $p(a)$
2. $\neg p(X) \vee p(f(X))$
3. $\neg p(f(Y))$
4. $p(f(a))$ from 1,2 with $\sigma = \{X \mapsto a\}$
5. $p(f(f(a))$ from 5,2 with $\sigma = \{X \mapsto a\}$
6. $p(f(f(f(a)))$ from 4,2 with $\sigma = \{X \mapsto a\}$
7. $p(f(f(f(f(a))))$ from 5,2 with $\sigma = \{X \mapsto a\}$
8. ...

DPLL and Resolution

DPLL on C':

1. $p(a)$
2. $\neg p(a) \vee p(f(a))$
3. $\neg p(f(a))$
4. Propagate 1: $p(f(a))$ (from 2)
5. Propagate 4: $\Box$ (from 3)

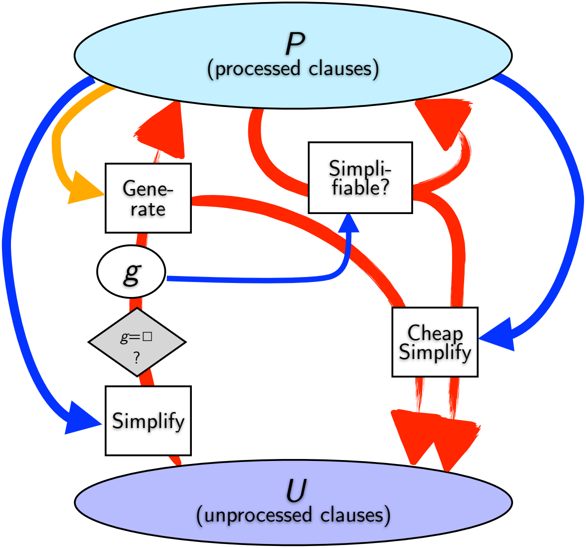
No decision/split, hence no backtracking: C' is unsatisfiable

But:
Instantiations provided externally!

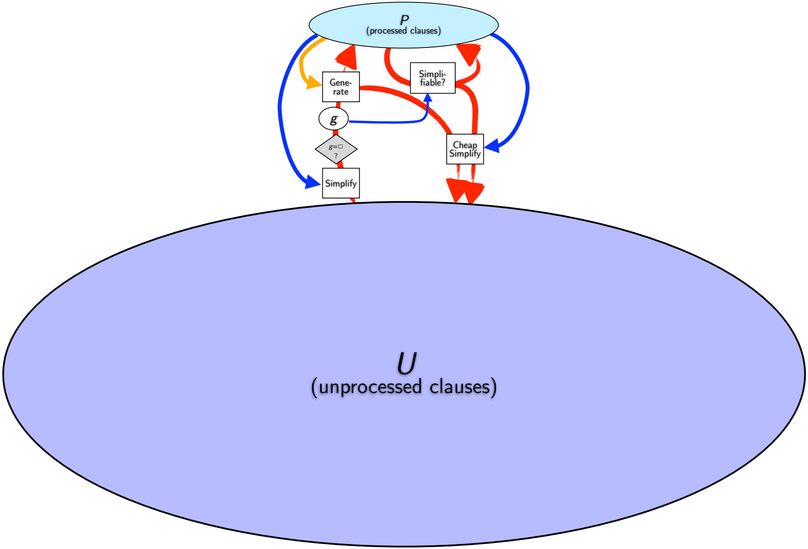
Unification-based saturation needs:

- Systematic inference control
- Fair inference strategy
- Good heuristic guidance

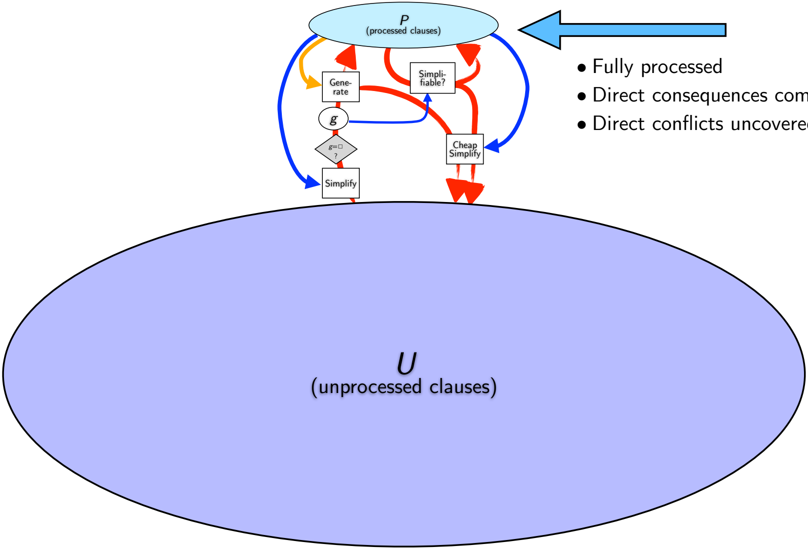
Saturation: Implementation and Observation



Saturation: Implementation and Observation



Saturation: Implementation and Observation

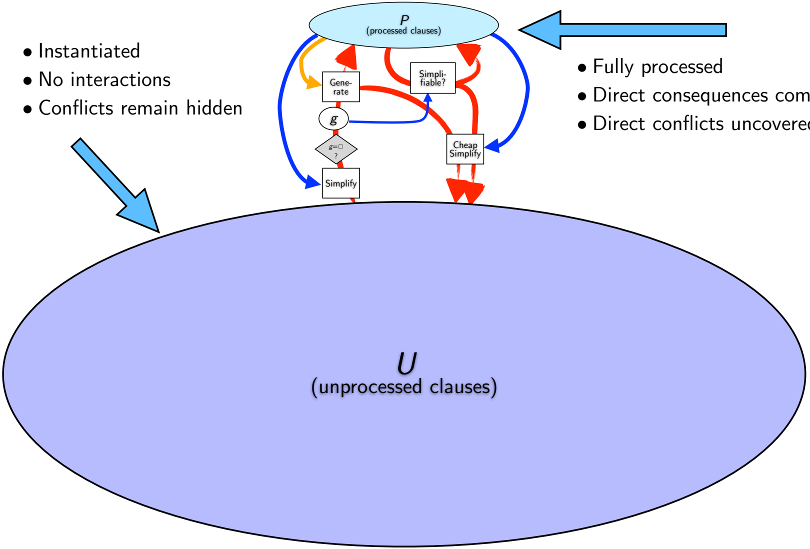


- Fully processed
- Direct consequences computed
- Direct conflicts uncovered

Saturation: Implementation and Observation

- Instantiated
- No interactions
- Conflicts remain hidden

- Fully processed
- Direct consequences computed
- Direct conflicts uncovered

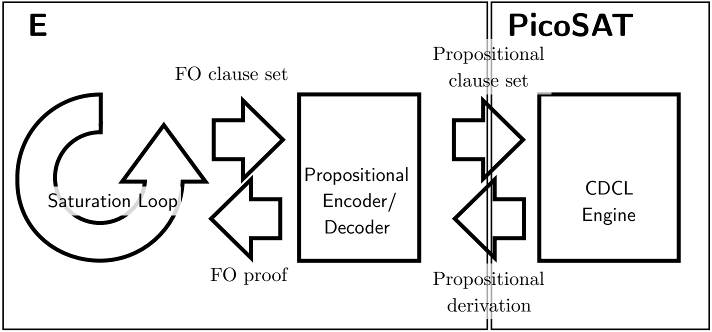


The Best of Both Worlds

- Combine saturation and CDCL
  - Saturation creates instances in controlled manner
  - CDCL uncovers hidden conflicts
- Implemention
  - Standard given-clause saturation algorithm (E)
  - Periodic grounding and SAT check (PicoSAT)

The Best of Both Worlds

- Combine saturation and CDCL
  - Saturation creates instances in controlled manner
  - CDCL uncovers hidden conflicts
- Implemention
  - Standard given-clause saturation algorithm (E)
  - Periodic grounding and SAT check (PicoSAT)



The Best of Both Worlds

while $U \neq \{\}$
    if prop_trigger(U,P)
        if prop_unsat_check(U,P)
            SUCCESS, Proof found
    $g = \text{extract\_best}(U)$
    $g = \text{simplify}(g, P)$
    if $g == \Box$
        SUCCESS, Proof found
    if g is not subsumed by any clause in P (or otherwise redundant w.r.t. P)
        $P = P \backslash \{c \in P \mid c \text{ subsumed by (or otherwise redundant w.r.t.) } g\}$
        $T = \{c \in P \mid c \text{ can be simplified with } g\}$
        $P = (P \backslash T) \cup \{g\}$
        $T = T \cup \text{generate}(g, P)$
        $T' = \{\}$
        foreach $c \in T$
            $c = \text{cheap\_simplify}(c, P)$
            if c is not trivial
                $T' = T' \cup \{c\}$
        $U = U \cup T'$
SUCCESS, original U is satisfiable

Experimental Setup

- E 2.1 with SAT extensions
- 16048 TPTP 7.0.0 CNF and FOF problems
- Different base strategies
- Different grounding constants
- 300 second overall time limit on StarExec cluster
- 3 seconds per attempt for PicoSAT

Core results

- Basic result: About 1% more proofs than plain saturation
- About 10% success on *hard* problems
  - Saturation alone solves ca. 90% of problems before first SAT check
  - PicoSAT contributes about 10% of proofs in cases where it is used

Core results

- Basic result: About 1% more proofs than plain saturation
- About 10% success on *hard* problems
  - Saturation alone solves ca. 90% of problems before first SAT check
  - PicoSAT contributes about 10% of proofs in cases where it is used
- SAT problem properties
  - Large (median 160 000 clauses)
  - Purity reduction removes ca. 90% of clauses
  - 95% easily satisfiable, 2.5% unsat, 2.5% timeout
  - Unsatisfiable core is small (median 4 clauses)
  - Successes for hard SAT and near-SAT problems

Core results

- Basic result: About 1% more proofs than plain saturation
- About 10% success on *hard* problems
  - Saturation alone solves ca. 90% of problems before first SAT check
  - PicoSAT contributes about 10% of proofs in cases where it is used
- SAT problem properties
  - Large (median 160 000 clauses)
  - Purity reduction removes ca. 90% of clauses
  - 95% easily satisfiable, 2.5% unsat, 2.5% timeout
  - Unsatisfiable core is small (median 4 clauses)
  - Successes for hard SAT and near-SAT problems

Success rate not overwhelming, but promising

SAT Proof Properties

Statistic	Min	1st q.	Median	3rd q.	Max
Clauses	3825	65972	160999	296951	2107682
Non-pure	2	1297	10478	36739	861260
Unsat core	2	3	4	10	1705

- "Unsatisfiable core is small (median 4 clauses)"
  - If only the saturation engine could magically pick the right clauses...
  - Further highlights the potential for good search heuristics for first-order reasoning!

SAT Proof Properties

Statistic	**Min**	**1st q.**	**Median**	**3rd q.**	**Max**
Clauses	3825	65972	160999	296951	2107682
Non-pure	2	1297	10478	36739	861260
Unsat core	2	3	4	10	1705

- "Unsatisfiable core is small (median 4 clauses)"
  - If only the saturation engine could magically pick the right clauses…
  - Further highlights the potential for good search heuristics for first-order reasoning!
- …but saturation will not beat CDCL on hard SAT problems
  - Orders of magnitude advantage in speed
  - Orders of magnitude davantage in memory

Heuristic choice points

- How often do we ground?/What is `prop_trigger()`?
  - Every n iterations of the main loop
  - Every n newly generated unprocessed clauses
  - Every time the number of terms inserted into the term bank for the first time exceeds $n * 2^k$ for $k \in \mathbb{N}$
- Which constants do we for instantiation?
  - Fresh constant
  - First constant
  - Most/least frequent constant in axioms/conjectures (various combinations)
- How long do we give the sat solver?
  - Limit on number of decision literals processed
  - Unlimited
  - (time limit - not implemented, I don't like the non-determinism)

Related Work (1)

- Clause Linking (Plaisted et al):
  - Simply create ("linking") instances via unification of clause pairs
  - Periodically ground and SAT-solve
  - Problem: How to pick which clauses to link?
- InstGen (Korovin/Ganzinger)
  - As clause linking, but guided by propositional model:
    - Find model for grounded clause set
    - If impossible: Problem is unsatisfiable
    - Otherwise: Lift propositional model to first-order
    - If that fails: Link conflicting clauses
  - Problem: No good equality handling

Related Work (2)

AVATAR (Voronkov's brood)

- Abstract propositional structure of clause set
  - Independent clause fragments are represented by propositional atoms
    - Independent: no variables shared with the rest of the clause
    - Equal fragments in different clauses represented by same atom
    - Ground and propositional literals are always independent
- While there are propositional models:
  - Saturate clause fragments forced true by model
  - Contradiction: Eliminate model
  - Satisfiable: Problem is satisfiable

  Out of propositional models: Unsatisfiable
- Problems:
  - (Good) implementation is expensive
  - There may not be abstractable propositional structure

And now for something completely different

The trouble with literal orderings

- Consider the following clause:

$$p(X,Y) \vee q(Y,Z) \vee r(Z,U) \vee s(U,X)$$

  - With Bachmair/Ganziger literal order: All incomparable
  - ... because (non-equational) literals are compared as terms
  - ... and different variables are uncomparable
- Four maximal literals!
  - Four *inference* literals
  - ... not good for search space!

Pseudo-transfinite literal orderings

- Term orderings for superpositions need four properties:
  - Termination
  - Extendable to ground-complete ordering
  - Compatibility with substitutions ($s > t \rightsquigarrow \sigma(s) > \sigma(t)$)
  - Compatibility with term structure ($s > t \rightsquigarrow f(\ldots s \ldots) > f(\ldots t \ldots)$)

Pseudo-transfinite literal orderings

- Term orderings for superpositions need four properties:
  - Termination
  - Extendable to ground-complete ordering
  - Compatibility with substitutions ($s > t \rightsquigarrow \sigma(s) > \sigma(t)$)
  - Compatibility with term structure ($s > t \rightsquigarrow f(\ldots s \ldots) > f(\ldots t \ldots)$)
- But: Literals cannot be nested!
  - We can drop the last condition for literal comparisons
- Alternative literal ordering: Compare predicate symbols first
  - Break ties conventionally
- Can (sometimes) reduce the number of maximal literals
  - Bachmair/Ganziner proof still goes through (I think ;-)

Pseudo-transfinite literal orderings

- Term orderings for superpositions need four properties:
  - Termination
  - Extendable to ground-complete ordering
  - Compatibility with substitutions ($s > t \leadsto \sigma(s) > \sigma(t)$)
  - Compatibility with term structure ($s > t \leadsto f(\ldots s \ldots) > f(\ldots t \ldots)$)
- But: Literals cannot be nested!
  - We can drop the last condition for literal comparisons
- Alternative literal ordering: Compare predicate symbols first
  - Break ties conventionally
- Can (sometimes) reduce the number of maximal literals
  - Bachmair/Ganziner proof still goes through (I think ;-)

Initial results: Not a killer, but adds useful variety!

And now for something completely different

Stronger rewriting

- Fact: Incompatable variabls make terms incomparable
- Standard implementation of rewriting with unorientable equations:
  - Match potential left hand side onto subterm
  - Check generated instance for orientability
- Standard implementation will never be able to use e.g. $f(X, a) = f(b, Y)$
  - Free variable Y makes right hand side potentially larger
  - Happens more often than one might think!
- Solution: Force intantiation of RHS variables
  - Pick smallest constant (of the right sort)
  - Bind all unbound variables of the RHS

Stronger rewriting

- Fact: Incompatable variabls make terms incomparable
- Standard implementation of rewriting with unorientable equations:
  - Match potential left hand side onto subterm
  - Check generated instance for orientability
- Standard implementation will never be able to use e.g. $f(X, a) = f(b, Y)$
  - Free variable Y makes right hand side potentially larger
  - Happens more often than one might think!
- Solution: Force intantiation of RHS variables
  - Pick smallest constant (of the right sort)
  - Bind all unbound variables of the RHS

Initial results: Not a killer, but adds useful variety!

Future Work

- Future work
  - Explore different grounding and preprocessing options
  - Explore interaction with other heuristics
  - Mine propositional models for interesting conflicts (a la InstGen)
  - Use EUF SMT solver to handle ground equality
  - (Maybe) use general SMT solver to handle theories (?)
- New literal ordering & Strong rewriting
  - Extend handling of equality-literal
  - Evaluate different strategies…
  - …in combination with strong rewriting

Conclusion

- SAT Integration
  - CDCL provers have become extremely powerful
  - First-order provers can leverage this power even with light-weight integration
  - Feature is part of the standard E distribution since E 2.2
- There are still significant calculus refinements
  - (Some) implementation neeeded
  - Evaluation needed

Conclusion

- SAT Integration
  - CDCL provers have become extremely powerful
  - First-order provers can leverage this power even with light-weight integration
  - Feature is part of the standard E distribution since E 2.2
- There are still significant calculus refinements
  - (Some) implementation neeeded
  - Evaluation needed

Thank you!

Conclusion

- SAT Integration
  - CDCL provers have become extremely powerful
  - First-order provers can leverage this power even with light-weight integration
  - Feature is part of the standard E distribution since E 2.2
- There are still significant calculus refinements
  - (Some) implementation neeeded
  - Evaluation needed

Questions?