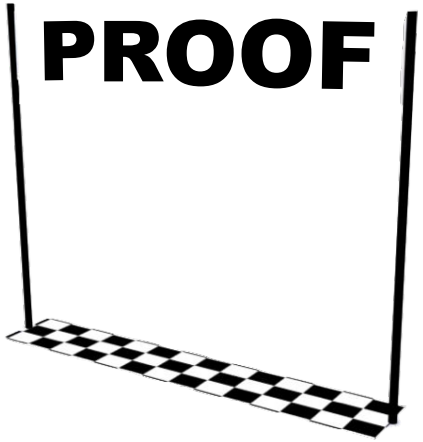
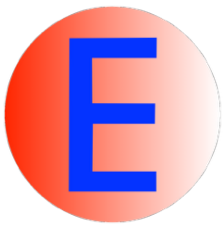
Performance of Clause Selection Heuristics for Saturation-Based Theorem Proving

Stephan Schulz

Martin Möhrmann





DHBW

Agenda

- Introduction
- Heuristics for saturating theorem proving
  - Saturation with the given-clause algorithm
  - Clause selection heuristics
- Experimental setup
- Results and analysis
  - Comparison of heuristics
  - Potential for improvement - how good are we?
- Conclusion

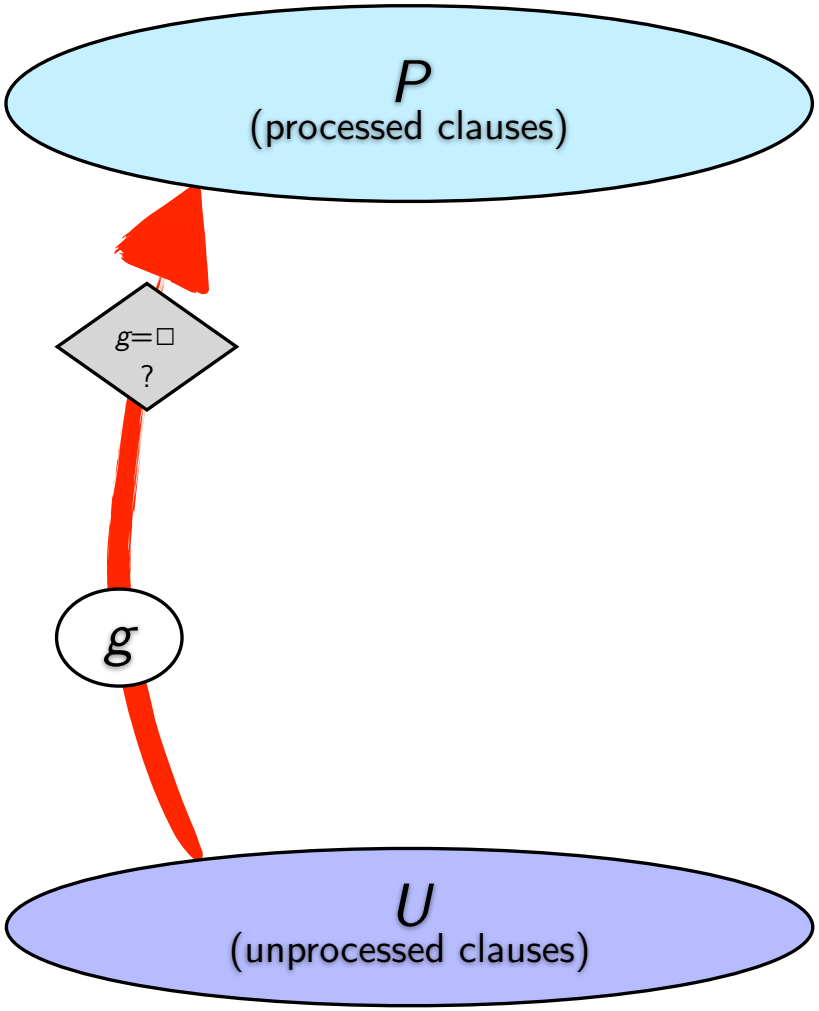
Introduction

- Heuristics are crucial for first-order theorem provers
  - Practical experience is clear
  - Proof search happens in an infinite search space
  - Proofs are rare
- A lot of collected developer experience (folklore)
  - . . . but no (published) systematic evaluation
  - . . . and no (published) recent evaluation at all

Saturating Theorem Proving

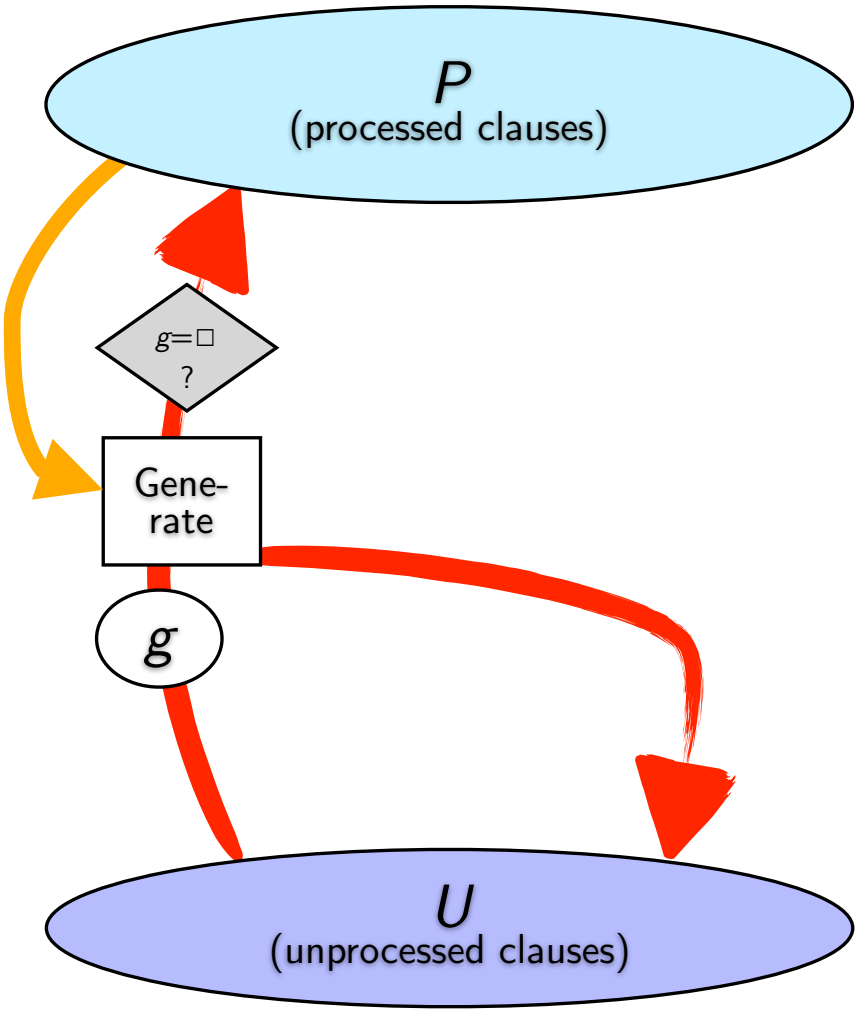
- Search state is a set of first-order clauses
- Inferences add new clauses
  - Existing clauses are premises
  - Inference generates new clause
  - If clause set is unsatisfiable then $\square$ can eventually be derived
  - Redundancy elimination (rewriting, subsumption ...) simplifies search state
- Inference rules try to minimize necessary consequences
  - Restricted by term orderings
  - Restricted by literal orderings
- Question: In which order do we compute potential consequences?
  - Given-clause algorithm
  - Controlled by clause selection heuristic

The Given-Clause Algorithm



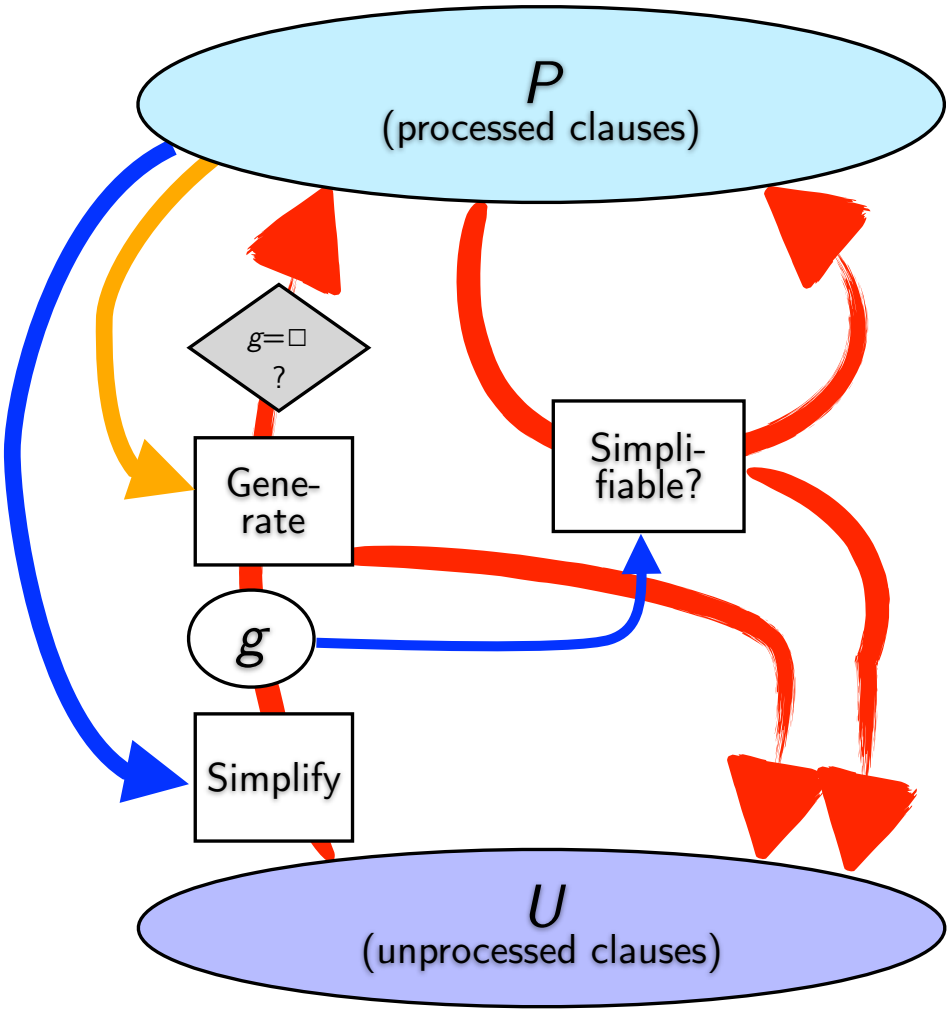
- Aim: Move everything from U to P

The Given-Clause Algorithm



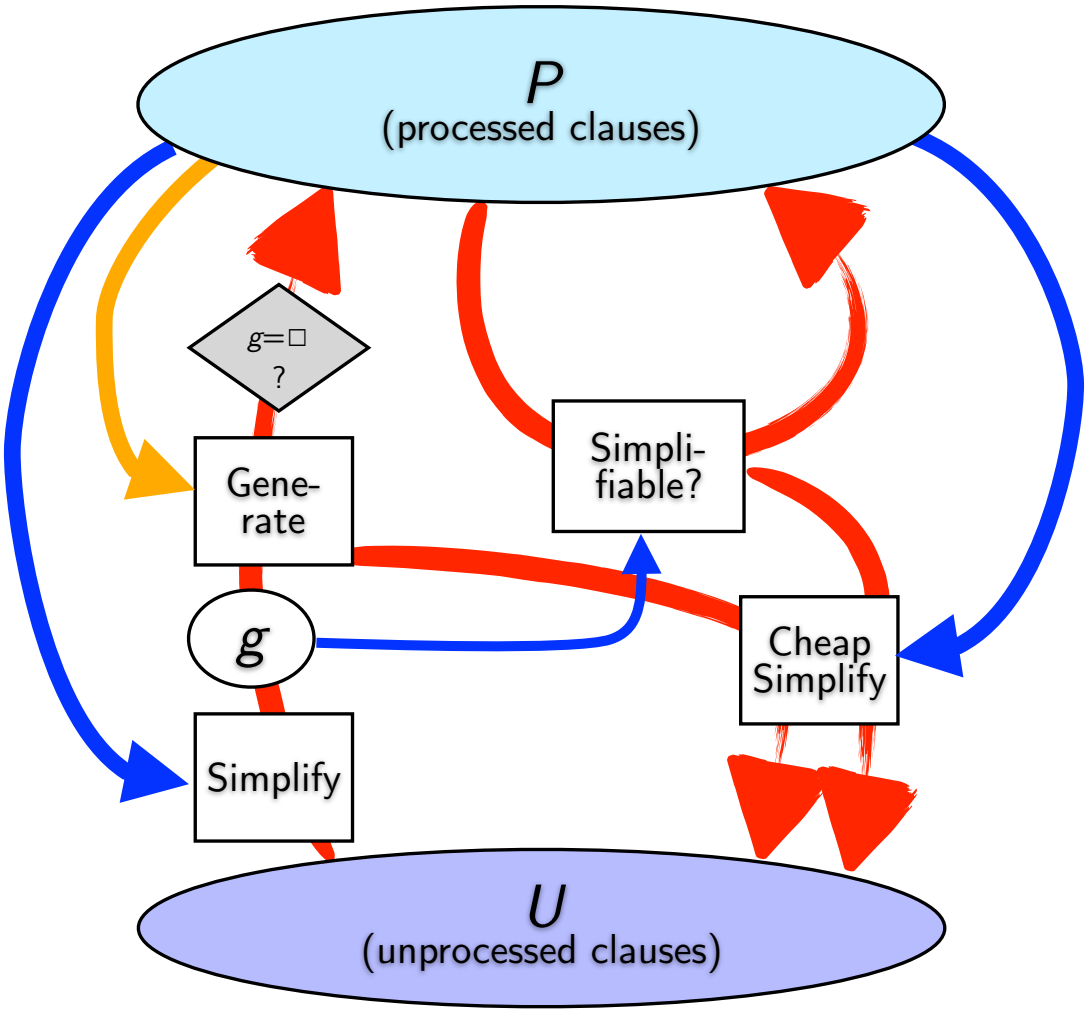
- Aim: Move everything from U to P
- Invariant: All generating inferences with premises from P have been performed

The Given-Clause Algorithm



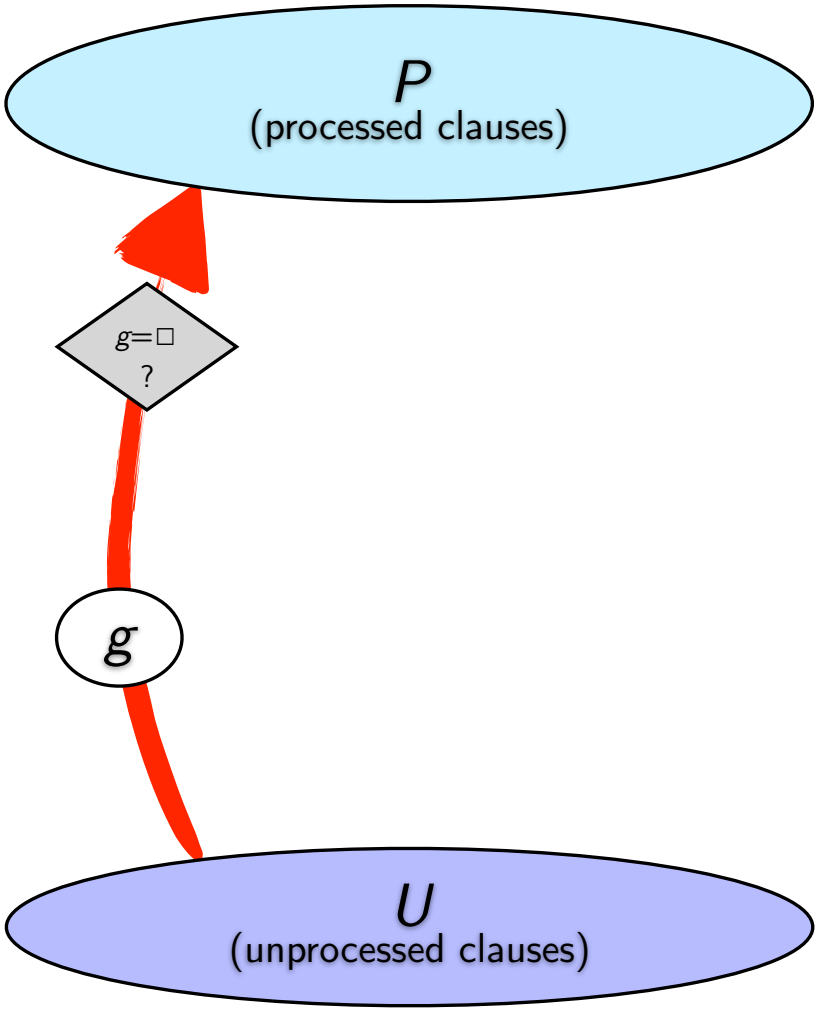
- Aim: Move everything from U to P
- Invariant: All generating inferences with premises from P have been performed
- Invariant: P is interreduced

The Given-Clause Algorithm



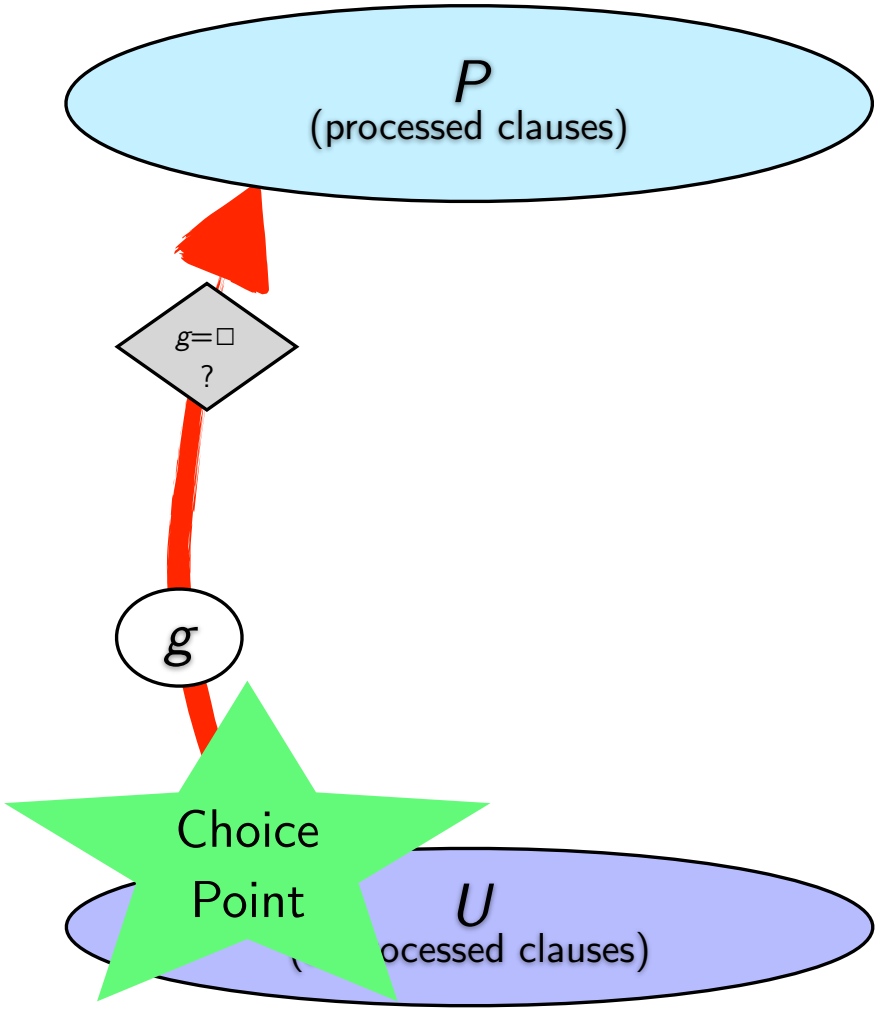
- Aim: Move everything from U to P
- Invariant: All generating inferences with premises from P have been performed
- Invariant: P is interreduced
- Clauses added to U are simplified with respect to P

Choice Point Clause Selection



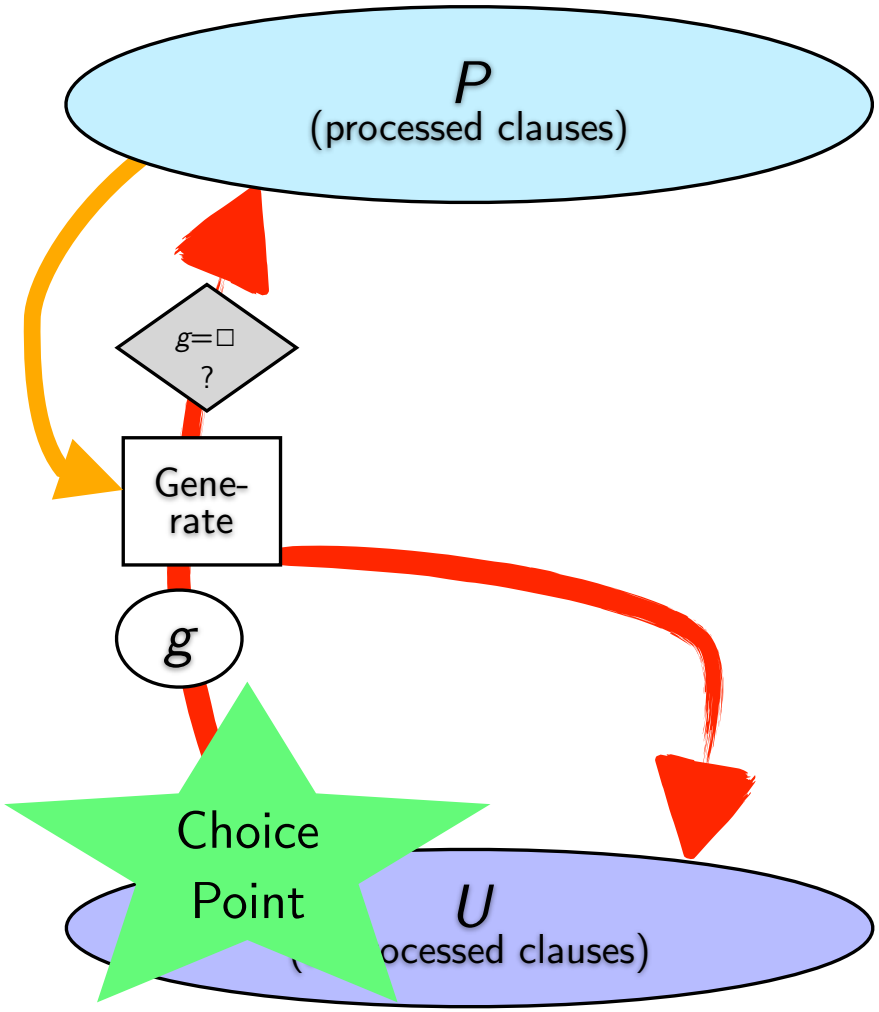
- Aim: Move everything from U to P

Choice Point Clause Selection



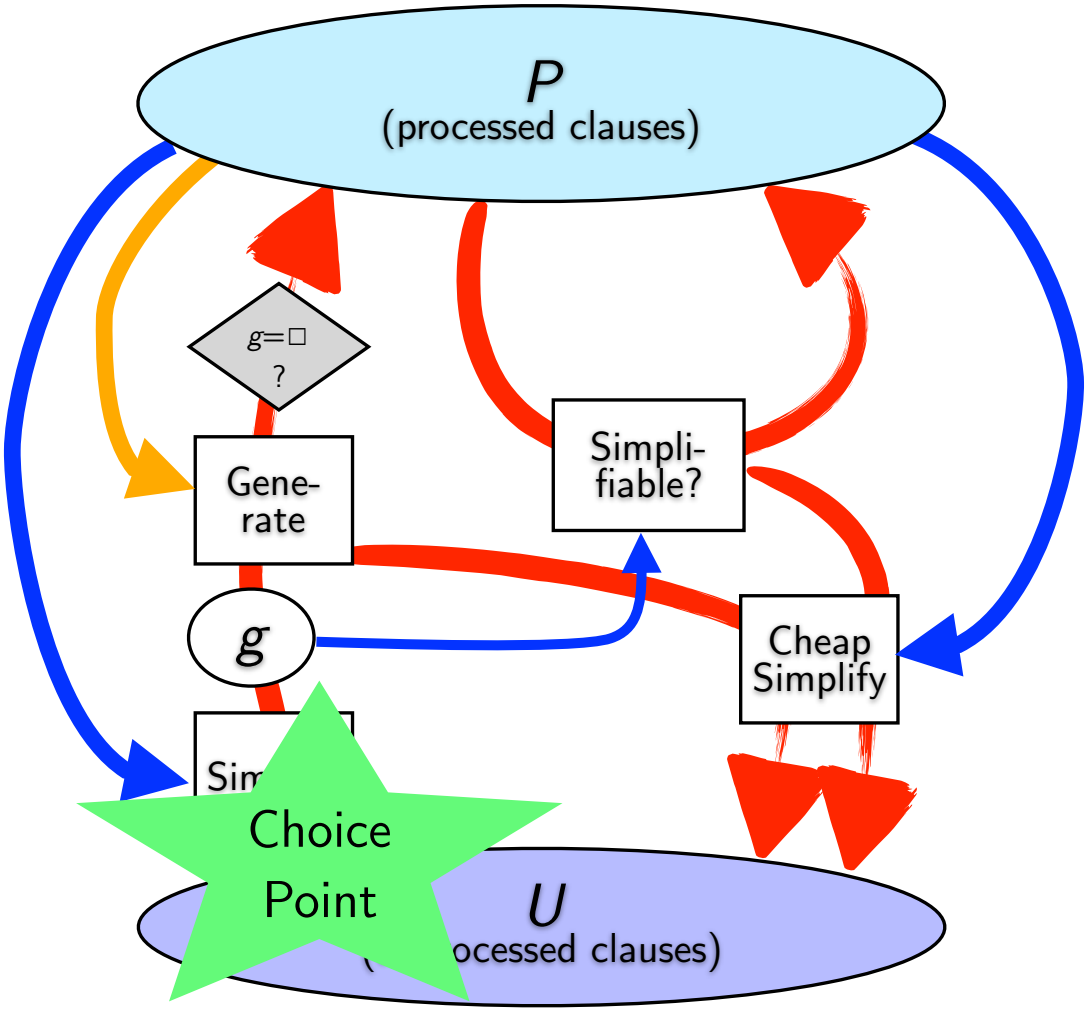
- Aim: Move everything from U to P
- Without generation: Only choice point!

Choice Point Clause Selection



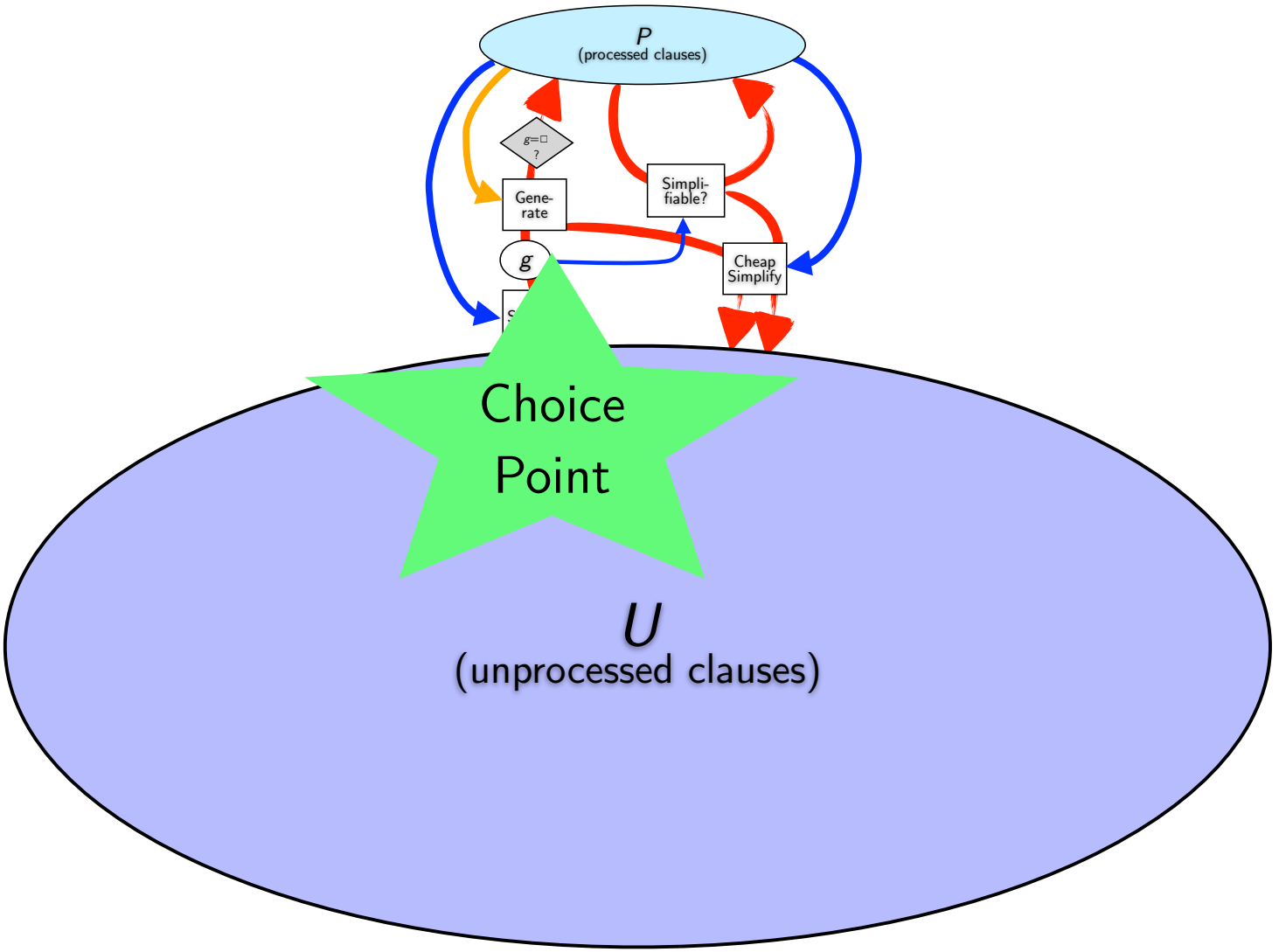
- Aim: Move everything from U to P
- Without generation: Only choice point!
- With generation: Still the major dynamic choice point!

Choice Point Clause Selection

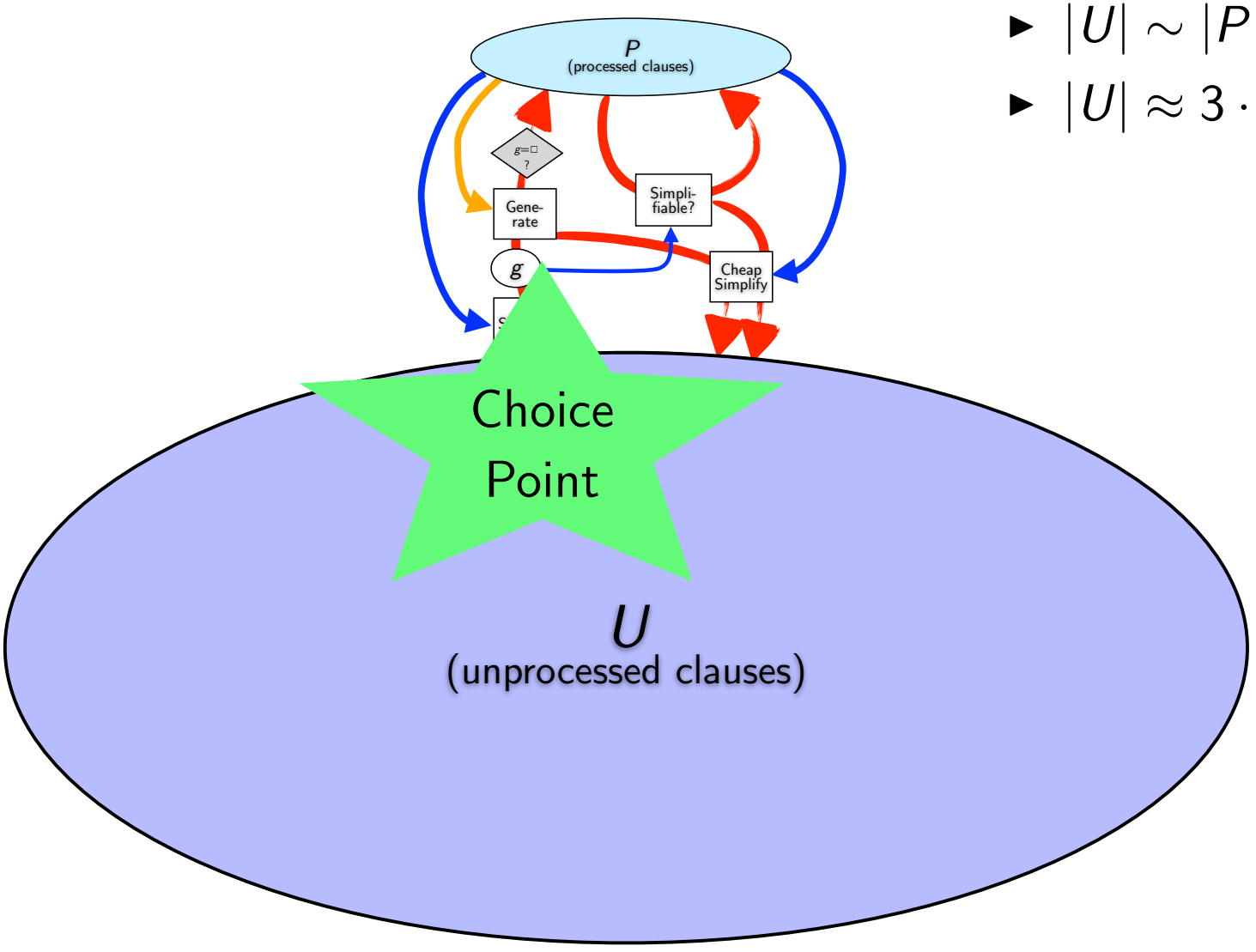


- Aim: Move everything from U to P
- Without generation: Only choice point!
- With generation: Still the major dynamic choice point!
- With simplification: Still the major dynamic choice point!

The Size of the Problem

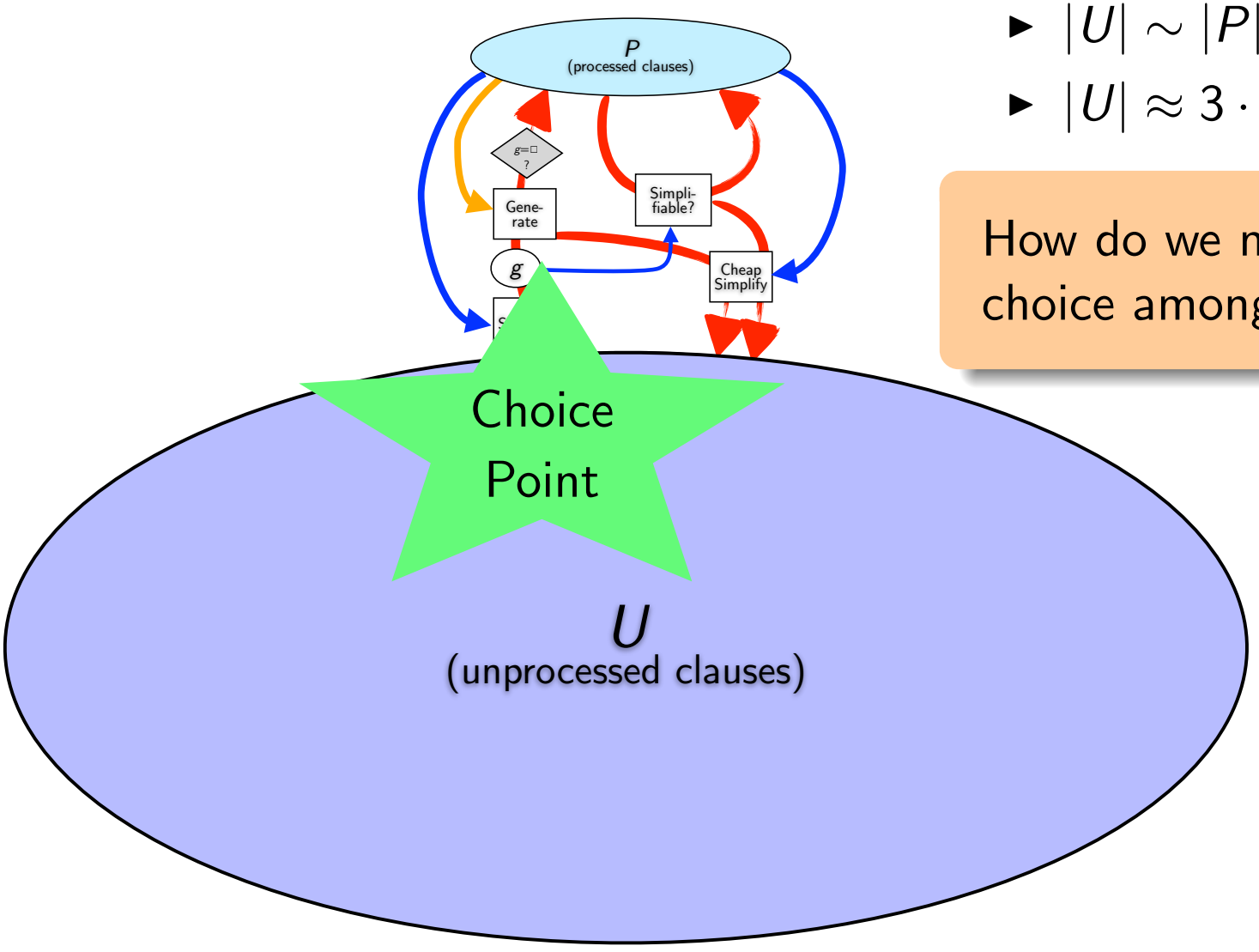


The Size of the Problem



- $|U| \sim |P|^2$
- $|U| \approx 3 \cdot 10^7$ after 300s

The Size of the Problem



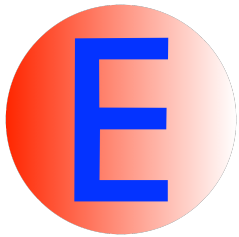
- $|U| \sim |P|^2$
- $|U| \approx 3 \cdot 10^7$ after 300s

How do we make the best choice among millions?

Basic Clause Selection Heuristics

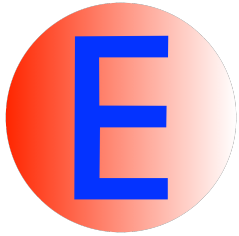
- Basic idea: Clauses ordered by heuristic evaluation
  - Heuristic assigns a numerical value to a clause
  - Clauses with smaller (better) evaluations are processed first
- Example: Evaluation by symbol counting
  - $|\{f(X) \neq a, P(a) \neq \$true, g(Y) = f(a)\}| = 10$
  - Motivation: Small clauses are general, $\square$ has 0 symbols
  - *Best-first* search
- Example: FIFO evaluation
  - Clause evaluation based on generation time (always prefer older clauses)
  - Motivation: Simulate *breadth-first* search, find shortest proofs
- Combine best-first/breadth-first seach
  - E.g. pick 4 out of every 5 clauses according to size, the last according to age

Clause Selection Heuristics in E



- Many symbol-counting variants
  - E.g. Assign different weights to symbol classes (predicates, functions, variables)
  - E.g. Goal directed: lower weight for symbols occuring in original conjecture
  - E.g. ordering-aware/calculus-aware: higher weight for symbols in inference terms
- Arbitrary combinations of base evaluation functions
  - E.g. 5 priority queues ordered by different evaluation functions, weighted round-robin selection

Clause Selection Heuristics in E



- Many symbol-counting variants
  - E.g. Assign different weights to symbol classes (predicates, functions, variables)
  - E.g. Goal directed: lower weight for symbols occuring in original conjecture
  - E.g. ordering-aware/calculus-aware: higher weight for symbols in inference terms
- Arbitrary combinations of base evaluation functions
  - E.g. 5 priority queues ordered by different evaluation functions, weighted round-robin selection

E can simulate nearly all other approaches to clause selection!

Folklore on Clause Selection/Evaluation

- FIFO is obviously fair, but awful – *Everybody*
- Prefering small clauses is good – *Everybody*
- Interleaving best-first (small) and breadth-first (FIFO) is better
  - *"The optimal pick-given ratio is 5"* – *Otter*
- Processing all initial clauses early is good – *Waldmeister*
- Preferring clauses with orientable equation is good – *DISCOUNT*
- Goal-direction is good – *E*

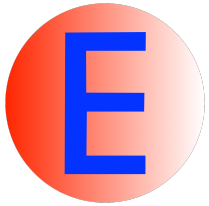
Folklore on Clause Selection/Evaluation

- FIFO is obviously fair, but awful – *Everybody*
- Prefering small clauses is good – *Everybody*
- Interleaving best-first (small) and breadth-first (FIFO) is better
  - *"The optimal pick-given ratio is 5"* – *Otter*
- Processing all initial clauses early is good – *Waldmeister*
- Preferring clauses with orientable equation is good – *DISCOUNT*
- Goal-direction is good – *E*

Can we confirm or refute these claims?

Experimental setup

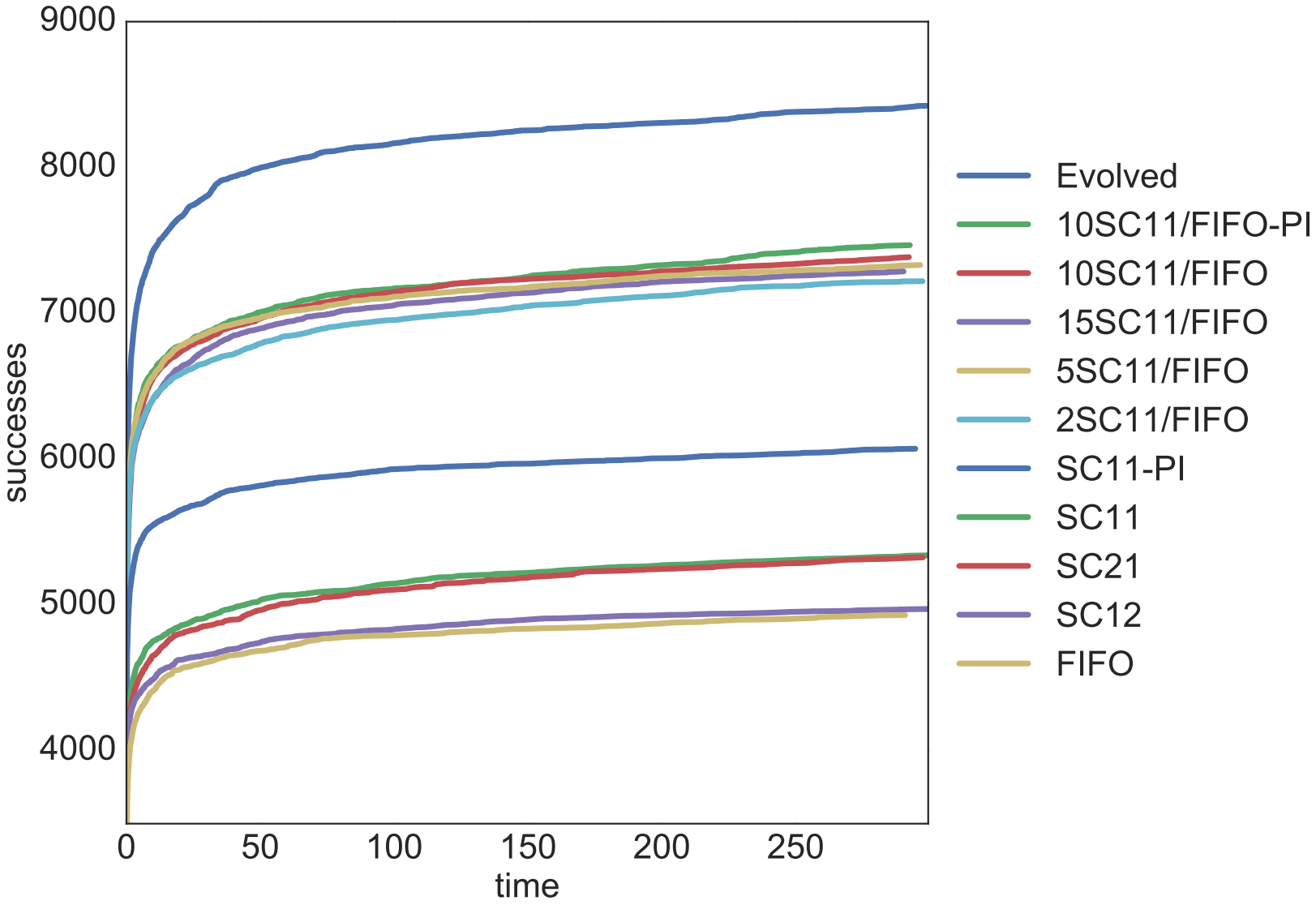
- Prover: E 1.9.1-pre
- 14 different heuristics
  - 13 selected to test folklore claims (interleave 1 or 2 evaluations)
  - Plus modern *evolved* heuristic (interleaves 5 evaluations)
- TPTP release 6.3.0
  - Only (assumed) provable first-order problems
  - 13774 problems: 7082 FOF and 6692 CNF
- Compute environment
  - StarExec cluster: single threaded run on Xeon E5-2609 (2.4 GHz)
  - 300 second time limit, no memory limit ($\geq$64 GB/core physical)



Meet the Heuristics

Heuristic	Rank	Successes		Successes within 1s	
		total	unique	absolute	of column 3
FIFO	14	4930 (35.8%)	17	3941	79.9%
SC12	13	4972 (36.1%)	5	4155	83.6%
SC11	9	5340 (38.8%)	0	4285	80.2%
SC21	10	5326 (38.7%)	17	4194	78.7%
RW212	11	5254 (38.1%)	13	5764	79.8%
2SC11/FIFO	7	7220 (52.4%)	24	5846	79.7%
5SC11/FIFO	5	7331 (53.2%)	3	5781	78.3%
10SC11/FIFO	3	7385 (53.6%)	1	5656	77.6%
15SC11/FIFO	6	7287 (52.9%)	6	5006	82.5%
GD	12	4998 (36.3%)	12	5856	78.4%
5GD/FIFO	4	7379 (53.6%)	62	4213	80.2%
SC11-PI	8	6071 (44.1%)	13	4313	86.3%
10SC11/FIFO-PI	2	7467 (54.2%)	31	5934	80.4%
Evolved	1	8423 (61.2%)	593	6406	76.1%

Successes Over Time



Folklore put to the Test

- FIFO is awful, prefering small clauses is good – mostly confirmed
  - In general, only modest advantage for symbol counting (36% FIFO vs. 39% for best SC)
  - Exception: UEQ (32% vs. 63%)

Folklore put to the Test

- FIFO is awful, prefering small clauses is good – mostly confirmed
  - In general, only modest advantage for symbol counting (36% FIFO vs. 39% for best SC)
  - Exception: UEQ (32% vs. 63%)
- Interleaving best-first/breadth-first is better – confirmed
  - 54% for interleaving vs. 39% for best SC
  - Influence of different pick-given ratios is surprisingly small
  - UEQ is again an outlier (60% for 2:1 vs. 70% for 15:1)
  - *The optimal pick-given ratio is 10* (for E)

Folklore put to the Test

- FIFO is awful, prefering small clauses is good – mostly confirmed
  - In general, only modest advantage for symbol counting (36% FIFO vs. 39% for best SC)
  - Exception: UEQ (32% vs. 63%)
- Interleaving best-first/breadth-first is better – confirmed
  - 54% for interleaving vs. 39% for best SC
  - Influence of different pick-given ratios is surprisingly small
  - UEQ is again an outlier (60% for 2:1 vs. 70% for 15:1)
  - *The optimal pick-given ratio is 10* (for E)
- Processing all initial clauses early is good – confirmed
  - Effect is less pronounced for interleaved heuristics

Folklore put to the Test

- FIFO is awful, prefering small clauses is good – mostly confirmed
  - In general, only modest advantage for symbol counting (36% FIFO vs. 39% for best SC)
  - Exception: UEQ (32% vs. 63%)
- Interleaving best-first/breadth-first is better – confirmed
  - 54% for interleaving vs. 39% for best SC
  - Influence of different pick-given ratios is surprisingly small
  - UEQ is again an outlier (60% for 2:1 vs. 70% for 15:1)
  - *The optimal pick-given ratio is 10* (for E)
- Processing all initial clauses early is good – confirmed
  - Effect is less pronounced for interleaved heuristics
- Preferring clauses with orientable equation is good – not confirmed
  - There is no evidence in our data, not even for UEQ

Folklore put to the Test

- FIFO is awful, prefering small clauses is good – mostly confirmed
  - In general, only modest advantage for symbol counting (36% FIFO vs. 39% for best SC)
  - Exception: UEQ (32% vs. 63%)
- Interleaving best-first/breadth-first is better – confirmed
  - 54% for interleaving vs. 39% for best SC
  - Influence of different pick-given ratios is surprisingly small
  - UEQ is again an outlier (60% for 2:1 vs. 70% for 15:1)
  - *The optimal pick-given ratio is 10* (for E)
- Processing all initial clauses early is good – confirmed
  - Effect is less pronounced for interleaved heuristics
- Preferring clauses with orientable equation is good – not confirmed
  - There is no evidence in our data, not even for UEQ
- Goal-direction is good – partially confirmed
  - GD on its own performs similar to SC
  - GD shines in combination with FIFO

Selected Results

- Good heuristics do make a difference
  - 71% more solutions with Evolved vs. FIFO
  - 58% more solutions with Evolved vs. best SC

Selected Results

- Good heuristics do make a difference
  - 71% more solutions with Evolved vs. FIFO
  - 58% more solutions with Evolved vs. best SC
- Success comes early
  - $\approx$ 80% of all proofs found in less than 1s
  - . . . with little variation between strategies (spread: 76%–84%)

Selected Results

- Good heuristics do make a difference
  - 71% more solutions with Evolved vs. FIFO
  - 58% more solutions with Evolved vs. best SC
- Success comes early
  - $\approx$ 80% of all proofs found in less than 1s
  - . . . with little variation between strategies (spread: 76%–84%)
- Cooperation beats portfolio/strategy scheduling
  - SC11 solves 5340 problems
  - FIFO solves 4930 problems
  - Union of the previous two contains 6329 problems
  - . . . but 10SC11/FIFO solves 7385

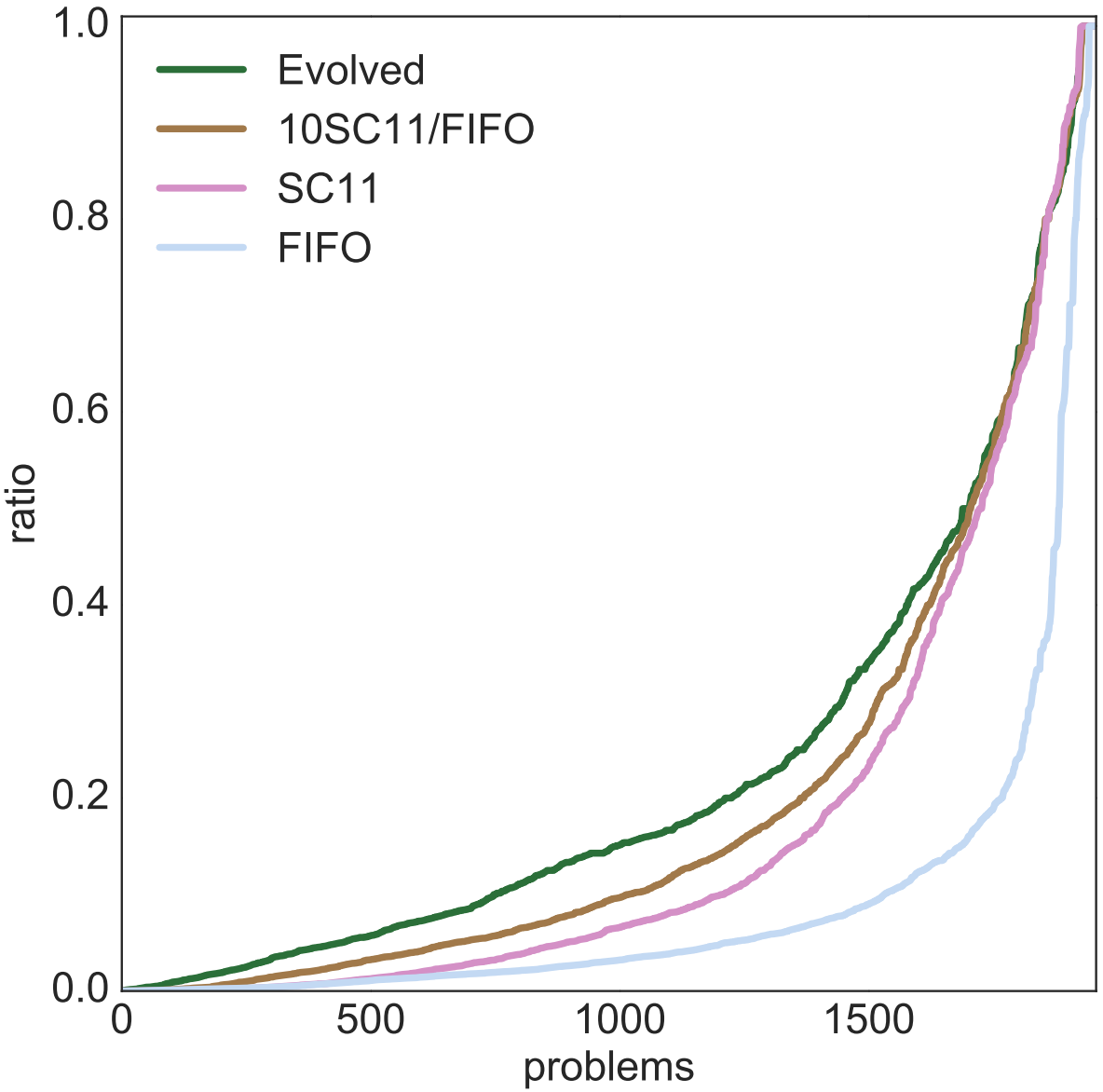
Selected Results

- Good heuristics do make a difference
  - 71% more solutions with Evolved vs. FIFO
  - 58% more solutions with Evolved vs. best SC
- Success comes early
  - $\approx$ 80% of all proofs found in less than 1s
  - . . . with little variation between strategies (spread: 76%–84%)
- Cooperation beats portfolio/strategy scheduling
  - SC11 solves 5340 problems
  - FIFO solves 4930 problems
  - Union of the previous two contains 6329 problems
  - . . . but 10SC11/FIFO solves 7385
- Evolving *Evolved* paid off
  - Significantly better than best "naive" heuristic
  - 10$\times$ more unique solutions than second-best

Measuring absolute Performance

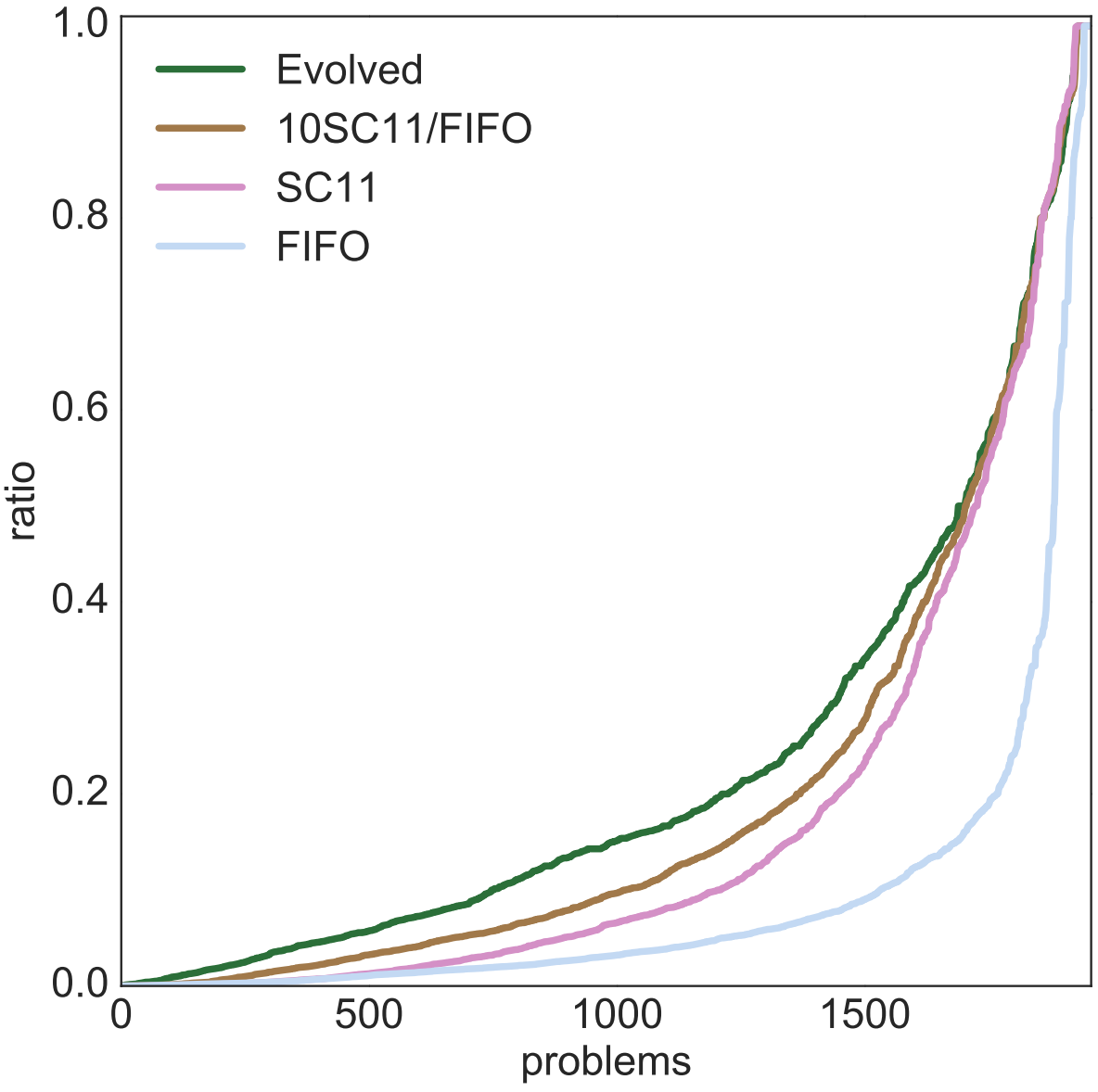
- Definition: Given-clause utilization
  - A useful given clause appears in the proof object
  - A useless given clause does not contribute to the proof
  - The *given-clause utilization ratio* for a proof search is the ratio of useful given clauses to all given clauses
- Given-clause utilization ratio measures heuristic quality
  - A perfect heuristic has a GCUR of 1
  - A failed heurisic has a GCUR of 0
  - A better heuristic will pick fewer useless clauses

Given Clause Utilization



- ≈ 2000 non-trivial problems solved by all 4 heuristics
- GCUR rank corresponds to global performance
- GCUR rates are low even for these easy problems

Given Clause Utilization



- $\approx$ 2000 non-trivial problems solved by all 4 heuristics
- GCUR rank corresponds to global performance
- GCUR rates are low even for these easy problems

Significant potential for improvement!

Future Work

- Evolve/develop better individual heuristics and collections of heuristics
  - (Even) more complex heuristics?
  - Evolve for diversity/swarm success
- Learn better heuristics from proofs/proof searches
  - Feature-based learning?
  - Pattern-based learning
  - Deep learning?
- Evaluate how results transfer to other situations
  - Otter-loop?
  - AVATAR?

Conclusion

- First-order proof search is a hard problem that critically depends on complex heuristics
- Developer folklore is useful
  - . . . but needs to be documented
  - . . . but needs to be re-verified
- Given-clause utilization is useful to gauge the quality of heuristics
- Given-clause utilization is low for current heuristics
  - . . . especially for harder problems
  - Huge potential for further improvements

Conclusion

- First-order proof search is a hard problem that critically depends on complex heuristics
- Developer folklore is useful
  - . . . but needs to be documented
  - . . . but needs to be re-verified
- Given-clause utilization is useful to gauge the quality of heuristics
- Given-clause utilization is low for current heuristics
  - . . . especially for harder problems
  - Huge potential for further improvements

Questions?

Conclusion

- First-order proof search is a hard problem that critically depends on complex heuristics
- Developer folklore is useful
  - . . . but needs to be documented
  - . . . but needs to be re-verified
- Given-clause utilization is useful to gauge the quality of heuristics
- Given-clause utilization is low for current heuristics
  - . . . especially for harder problems
  - Huge potential for further improvements

Thank you!