MUS Extraction using Clausal Proofs

Marijn J.H. Heule

THE UNIVERSITY OF TEXAS AT AUSTIN

Joint work with
Anton Belov and Joao Marques-Silva (University College Dublin)

SAT Conference, July 14, 2014

Context: MUS Extraction Requires a lot of Memory

Given an unsatisfiable formula F, a minimal unsatisfiable subset (MUS) of F is a unsatisfiable formula $F' \subseteq F$ such that all $F'' \subset F'$ are satisfiable.

State-of-the-art MUS extraction tools use a lot of memory

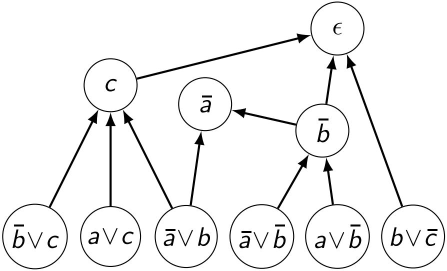
- The resolution graph is stored, either directly or indirectly via selector variables, to reuse learned clauses;
- Storing the graph heavily increases memory consumption;
- These tools run out of memory on hard instances or are significantly slowed down by high memory consumption.

We propose an alternative approach using clausal proofs that requires less memory and produces smaller resolution graphs

Resolution Proofs versus Clausal Proofs

Consider the formula $F := (\bar{b} \vee c) \wedge (a \vee c) \wedge (\bar{a} \vee b) \wedge (\bar{a} \vee \bar{b}) \wedge (a \vee \bar{b}) \wedge (b \vee \bar{c})$

A resolution graph of F is:



A resolution proof consists of all nodes and edges of the resolution graph

- graphs from CDCL solvers have ∼400 incoming edges per node
- resolution proof logging can heavily increase memory usage (×100)

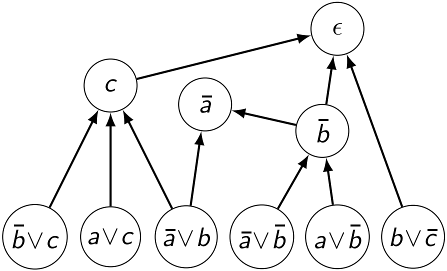
A clausal proof is a list of all nodes sorted by topological order

- clausal proofs are easy to emit and relatively small
- clausal proof checking requires reconstructing the edges (costly)

Reconstructing a Resolution Graph from a Clausal Proof

Consider the resolution graph on the left. The clausal proof is $\{(\bar{b}), (\bar{a}), (c), \epsilon\}$.

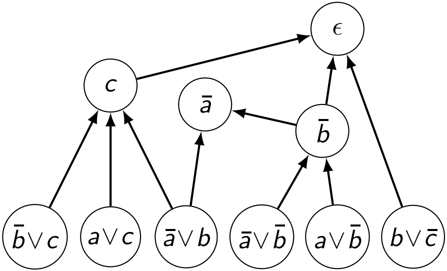
Recent work [FMCAD13] showed that one can obtain smaller cores using reconstruction heuristics.

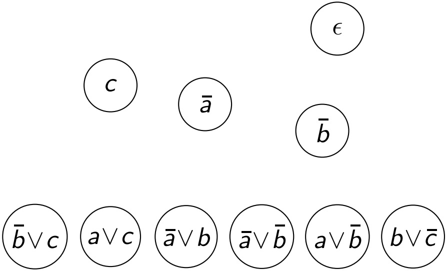


Reconstructing a Resolution Graph from a Clausal Proof

Consider the resolution graph on the left. The clausal proof is $\{(\bar{b}), (\bar{a}), (c), \epsilon\}$.

Recent work [FMCAD13] showed that one can obtain smaller cores using reconstruction heuristics.



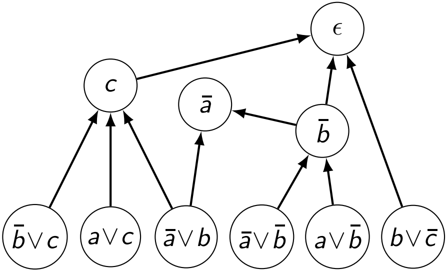


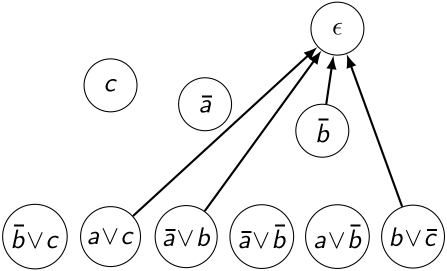
Reconstruction starts without incoming edges and traverses the proof in reverse order using unit propagation and conflict analysis

Reconstructing a Resolution Graph from a Clausal Proof

Consider the resolution graph on the left. The clausal proof is $\{(\bar{b}), (\bar{a}), (c), \epsilon\}$.

Recent work [FMCAD13] showed that one can obtain smaller cores using reconstruction heuristics.

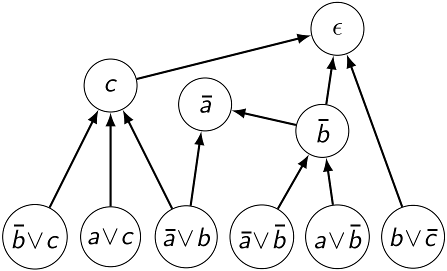




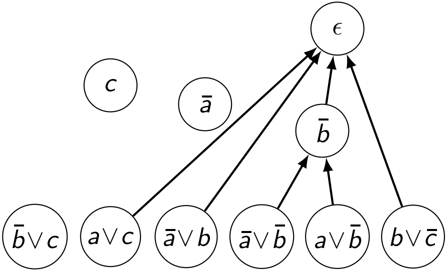
Reconstruction starts without incoming edges and traverses the proof in reverse order using unit propagation and conflict analysis

Reconstructing a Resolution Graph from a Clausal Proof

Consider the resolution graph on the left. The clausal proof is $\{(\bar{b}), (\bar{a}), (c), \epsilon\}$.



Recent work [FMCAD13] showed that one can obtain smaller cores using reconstruction heuristics.



Reconstruction starts without incoming edges and traverses the proof in reverse order using unit propagation and conflict analysis

State-of-the-Art MUS Extraction Algorithms

Two state-of-the-art approaches to extract a MUS:

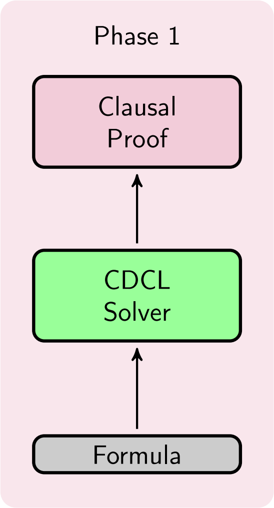
- Resolution-based using the resolution graph (HaifaMUC)
- Assumption-based using selector variables (MUSer2)

These approaches work as follows:

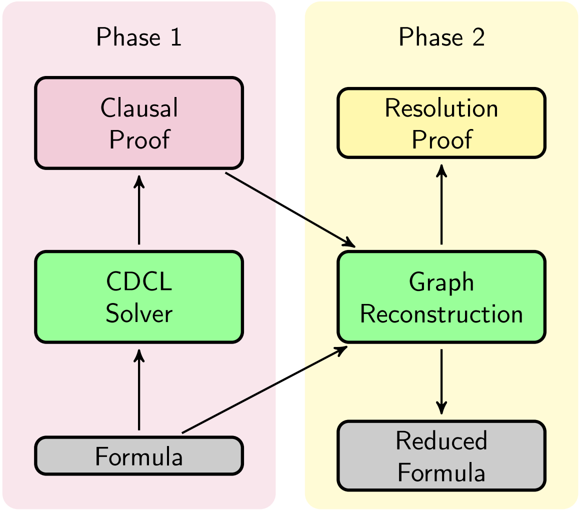
1. Solve the input formula and compute a resolution graph.
2. Remove all redundant clauses, i.e., those not in the core.
3. Pick a unmarked clause C, mark C, and solve the formula without C and all learned clauses that depend on C.
4. If that formula is satisfiable, then return to 3.
5. Otherwise update the resolution graph and return to 2.
6. Terminate when all clauses are marked.

Combining MUS Extraction and Clausal Proofs

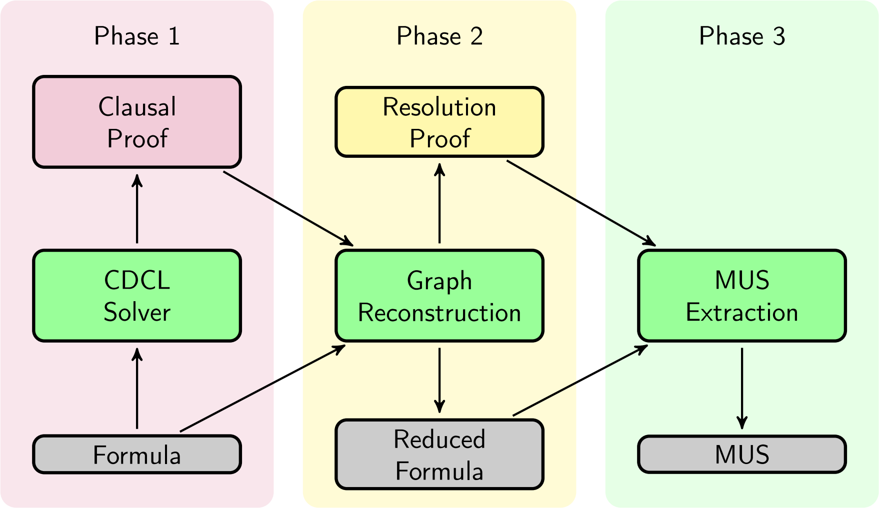
Trimming and MUS Extraction



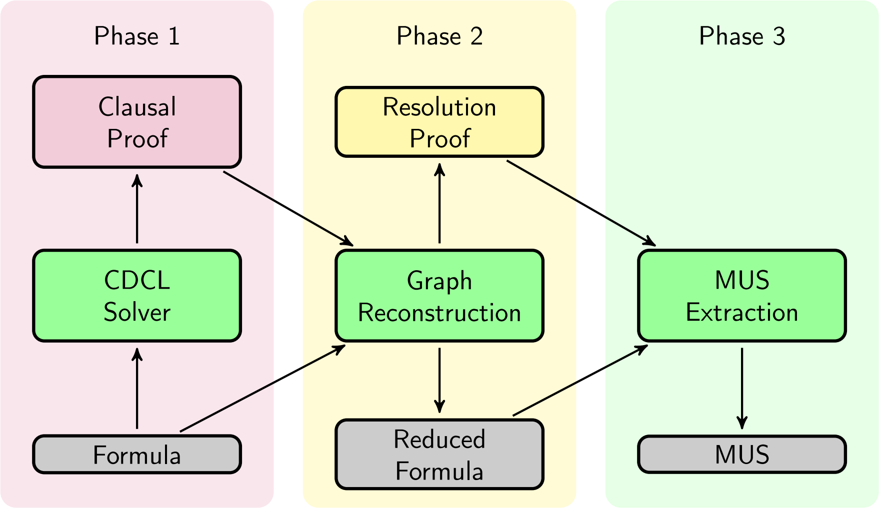
Trimming and MUS Extraction



Trimming and MUS Extraction

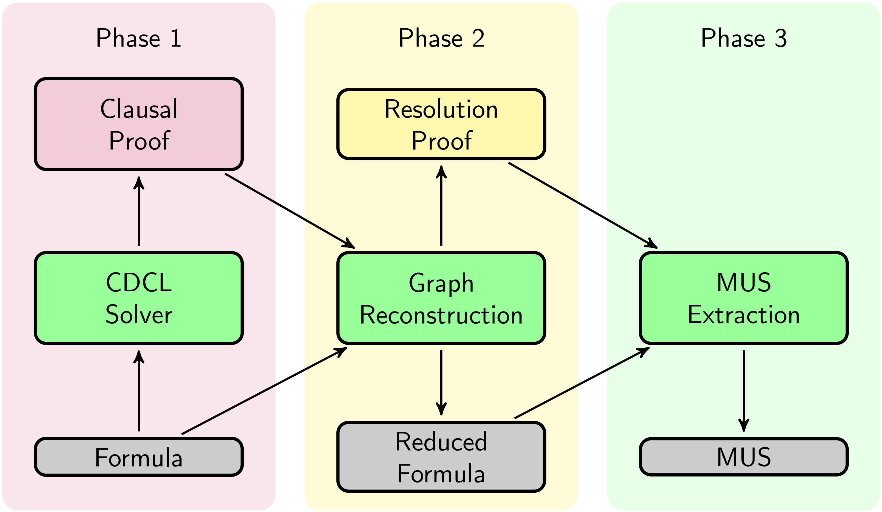


Trimming and MUS Extraction

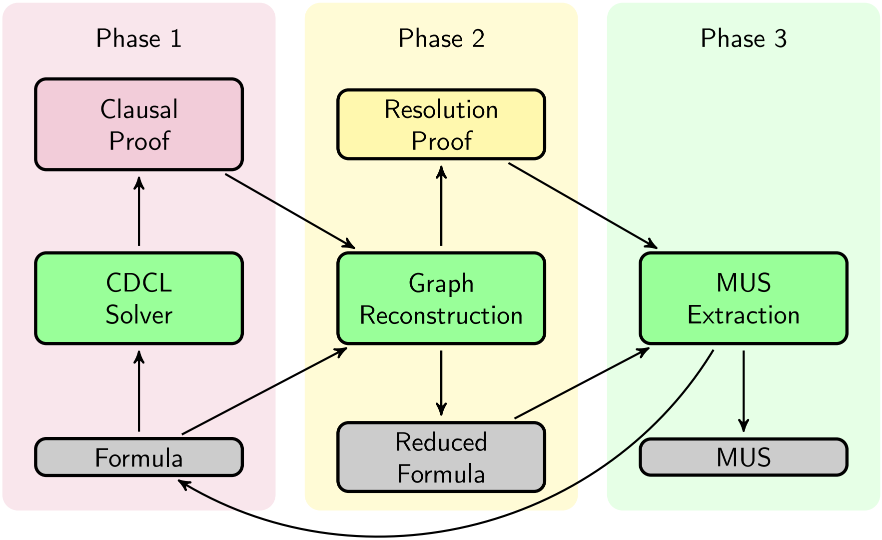


The clausal (Phase 1) and resolution proof (Phase 2) are stored on disk

Iterative Proof Trimming and MUS Extraction

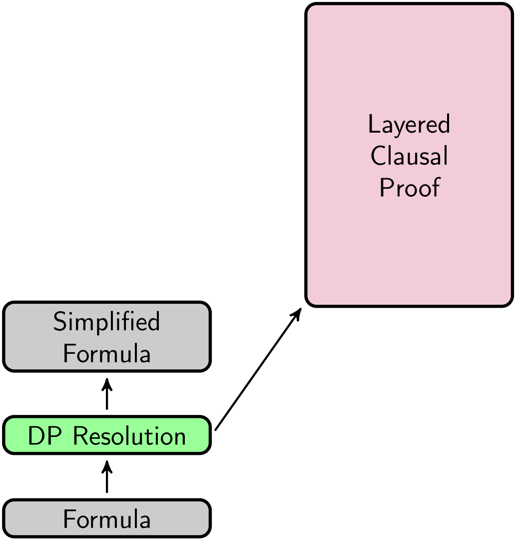


Iterative Proof Trimming and MUS Extraction



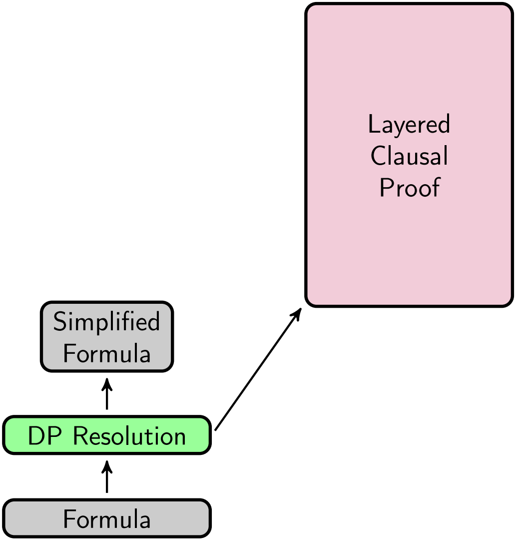
If a SAT call takes too much time, restart with the current core

Layered Trimming: Reduce Resolution Graph Connectivity



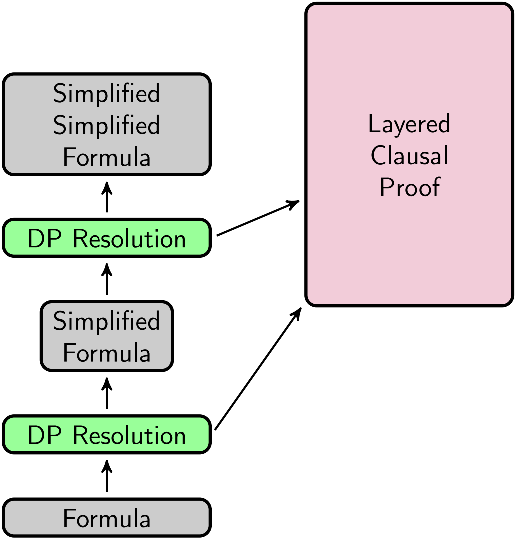
Key observation:
DP resolution creates nodes with two edges, while nodes created by CDCL solving have on average 400 edges.

Layered Trimming: Reduce Resolution Graph Connectivity



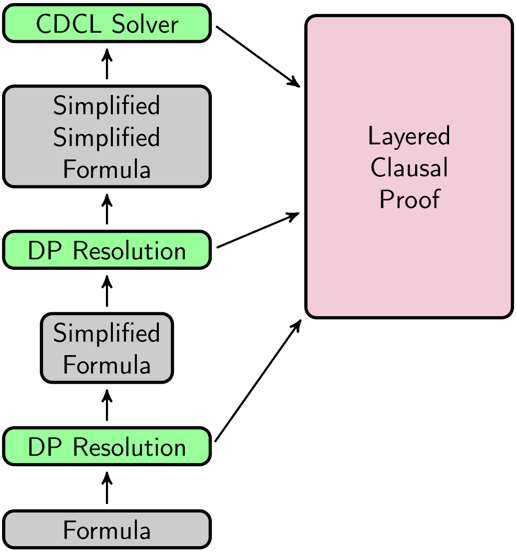
Key observation:
DP resolution creates nodes with two edges, while nodes created by CDCL solving have on average 400 edges.

Layered Trimming: Reduce Resolution Graph Connectivity



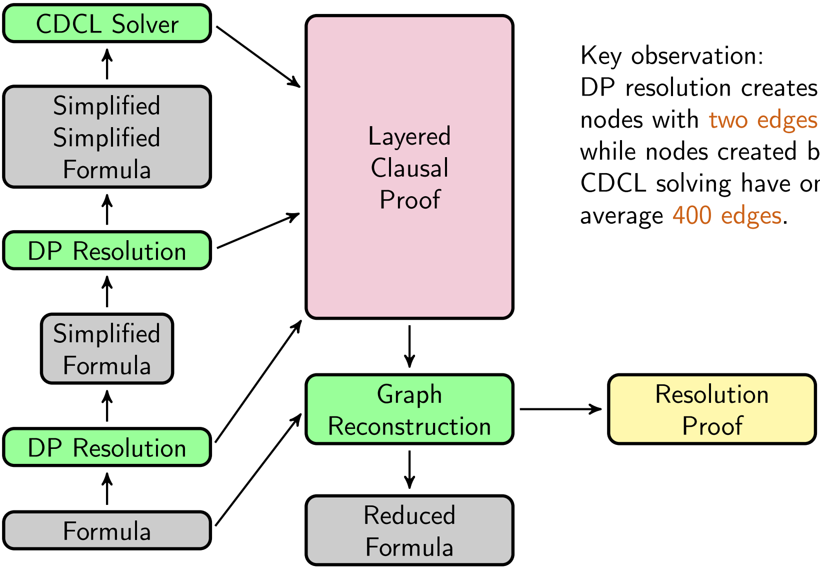
Key observation:
DP resolution creates nodes with two edges, while nodes created by CDCL solving have on average 400 edges.

Layered Trimming: Reduce Resolution Graph Connectivity



Key observation:
DP resolution creates nodes with two edges, while nodes created by CDCL solving have on average 400 edges.

Layered Trimming: Reduce Resolution Graph Connectivity



Key observation:
DP resolution creates nodes with two edges, while nodes created by CDCL solving have on average 400 edges.

Experimental Results

Experimental Results: Setup and Table

Hardware and Limits:

- 2 x Intel E5-2620 (2GHz) cluster nodes
- timeout 1800 seconds CPU time, 4 GB memory limit

Benchmarks

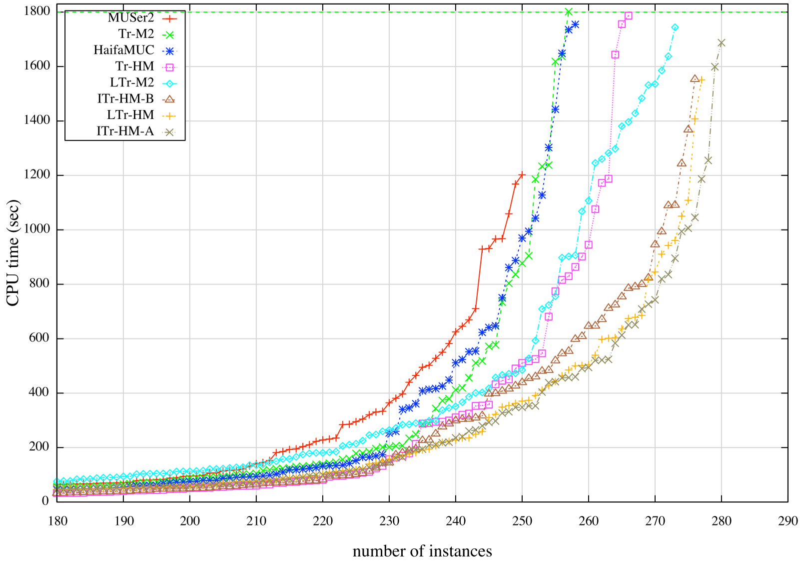
- 295 instances from the MUS Competition 2011
- 60 instances from SAT-COMP-09 which glucose solves in one minute

Abbreviations

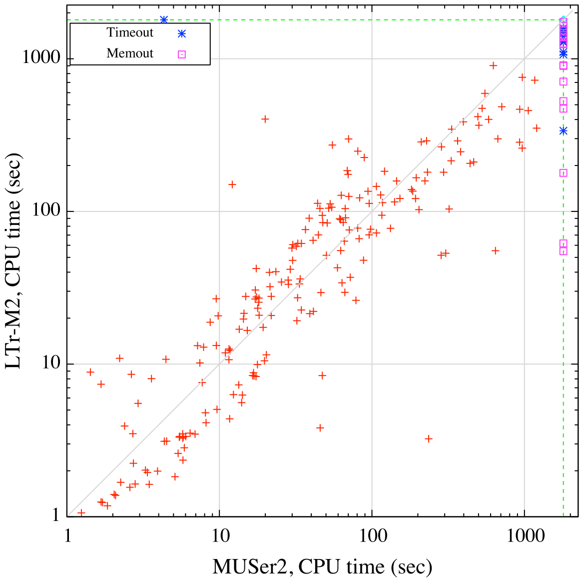
- M2: MUSer2
- HM: HaifaMUC
- Tr: Trimming
- LTr: Layered Trimming
- ITr: Iterative Trimming
- A/B: Approaches

	M2	HM	Tr-M2	Tr-HM	LTr-M2	LTr-HM	ITr-HM-A	ITr-HM-B
# solved	250	258	257	266	273	277	**280**	276
# TO/MO	26/48	40/26	39/28	51/7	32/19	47/0	44/0	48/0
Med. CPU	45.08	30.65	40.64	23.70	54.07	33.87	35.77	**23.16**
Avg. CPU	97.58	110.1	103.0	102.0	162.6	108.5	117.1	112.1

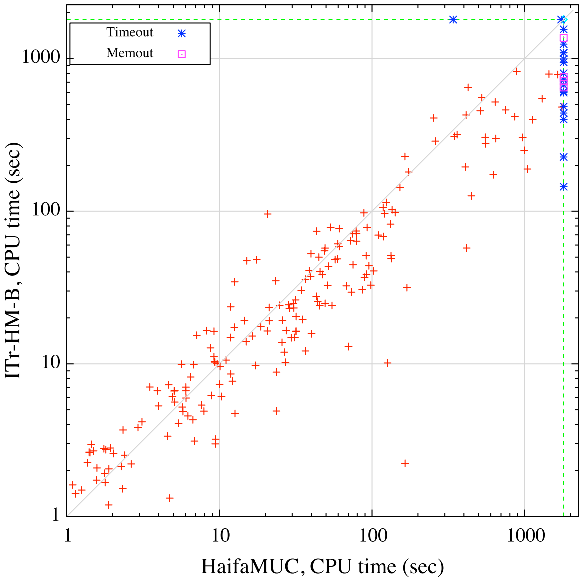
Experimental Results: Cactus Plot



Experimental Results: MUSer2



Experimental Results: HaifaMUC



Conclusions

Conclusions

Resolution graphs can be reconstructed from clausal proofs:

- Clausal proofs can easily be obtained from most solvers;
- Reconstruction requires only a fraction of the memory used to computing the graph during solving;
- Reconstruction typically removes many redundant clauses.

Our clausal proof approaches improve MUS extraction:

- Trimming is useful as preprocessing for MUS extraction;
- Iterative trimming is useful for resolution-based MUS tools;
- Layered trimming is useful for selector-based MUS tools;
- Speed-ups for both HaifaMUC and MUSer2.

Thanks!

Today @ 16:50: Wetzler, Heule, and Hunt, Jr. DRAT-trim: Efficient Checking and Trimming Using Expressive Clausal Proofs