Validated Model Counting Data
Version of February 28, 2023

Randal E. Bryant

Computer Science Department
Carnegie Mellon University, Pittsburgh, PA, United States

Benchmark machine

- 2021 MacBook Pro 18,4
  - 3.2 Ghz Apple M1 processor
  - 64 GB Ram
- Samsung T7 solid-state disk
  - 2TB capacity
  - Up to 1 GB/s R/W
  - Critical for performance. Wrote and read files of over 162 GB.

Methodology

1. Retrieved 200 public model counting benchmarks from 2022 Model Counting Competition website

   `https://mccompetition.org/2022/mc_description.html`

   - 100 public instances from Track 1 (unweighted model counting)
   - 100 public instances from Track 2 (weighted model counting)
2. Removed duplicates
   - 20 instances were duplicated in the two tracks
   - Reduces set of benchmarks to 180
3. Ran D4 on 180 instances with time limit of 4000 seconds
   - Completed for 124 instances
4. Generated POG representations from dDNNF graphs
   - Simplifications via constant propagation
     - Number of POG nodes around $0.90\times$ number of dDNNF nodes
   - Used heuristics to identify root node, and then put in topological order with root node at end.
   - Measure complexity by number of defining clauses D
     - Product or sum node with degree k requires $k + 1$ clauses
     - D also equals sum of number of edges and number of nodes
     - For 124 instances, Min = 304, Median = 848,784, Mean = 93,107,937, Max = 2,761,457,765
5. Ran program D2P to generate CRAT file for maximum of 10,000 seconds
   - Completed for 108 instances

- Exceeded clause limit for 1 instance (Our programs represent a clause ID as a 32-bit signed value)
- Ran out of memory for 1 instance
- Timed out for 14 instances

6. For completed CRAT files, ran CRAT-CHECK CRAT checker
   - Tests all conditions for correct CRAT
   - Most time spent performing RUP checks for clause addition and deletion
   - Completed successfully for all 108 instances
   - Time generally faster than D2P. (min = 0.005, max = 1.4, harmonic mean = 0.10)

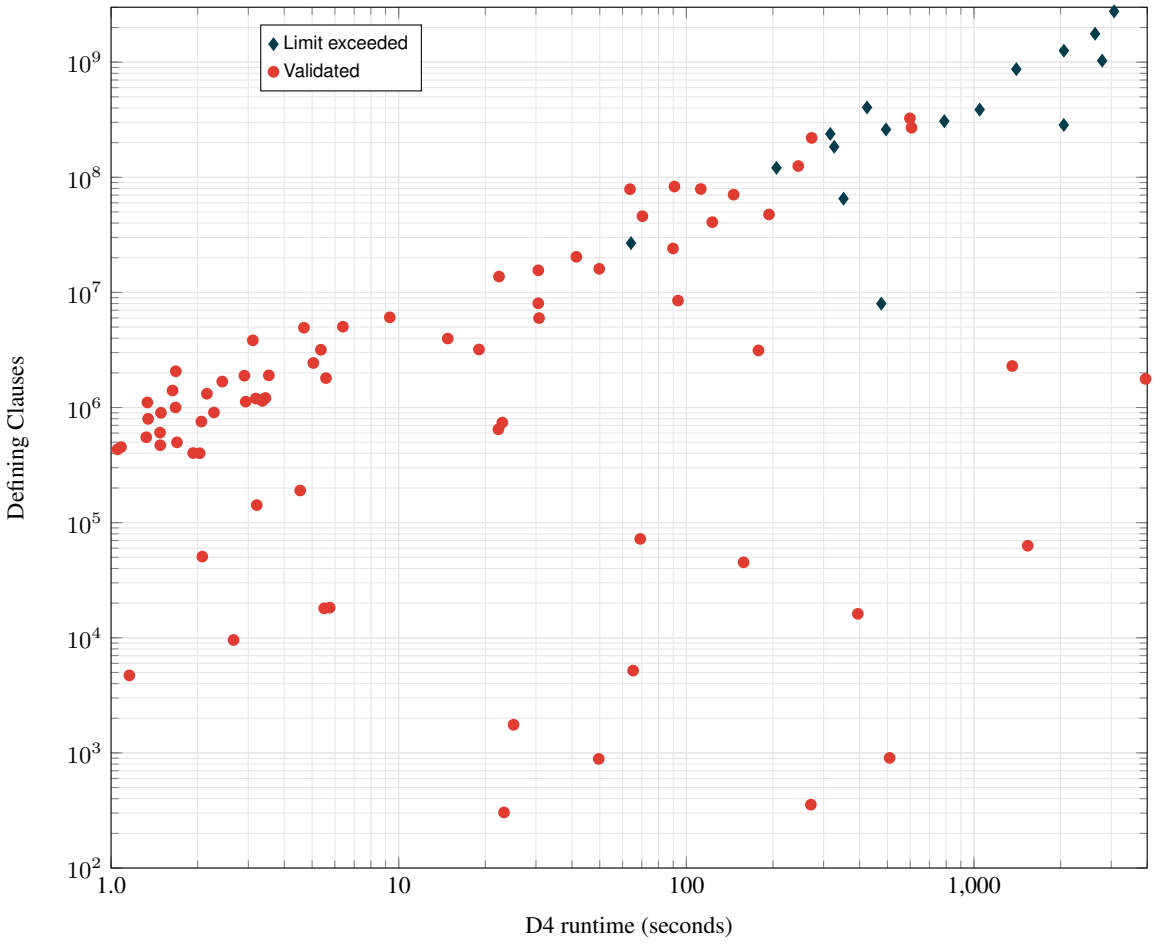

Fig. 1. Number of defining clauses in generated POG as function of D4 runtime

Generated CRAT Files

- Proof Clauses: Min 554, Median 1,719,245, Average 45,001,945, Max 770,382,773
- Ratio to number of defining clauses: Min 1.54, Harmonic Mean 3.13, Max 6072.96

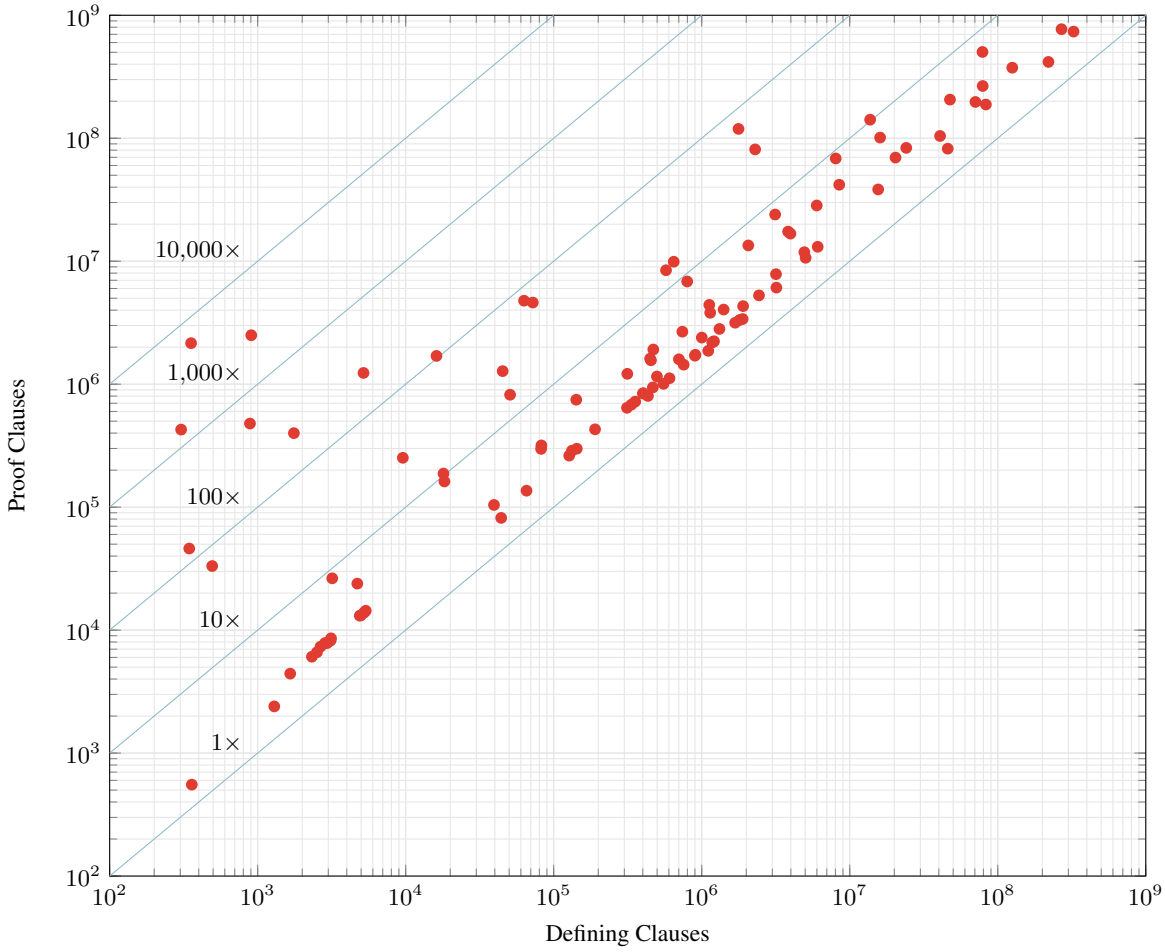

Fig. 2. Total number of clauses in CRAT file as function of number of defining clauses

Time to Generate and Validate CRAT

- Min 0.01, Average 1318.3, Max 13,144.8, Median 71.64
- Ratio to time for D4: Min 0.50, Harmonic Mean 5.50, Max 457.69

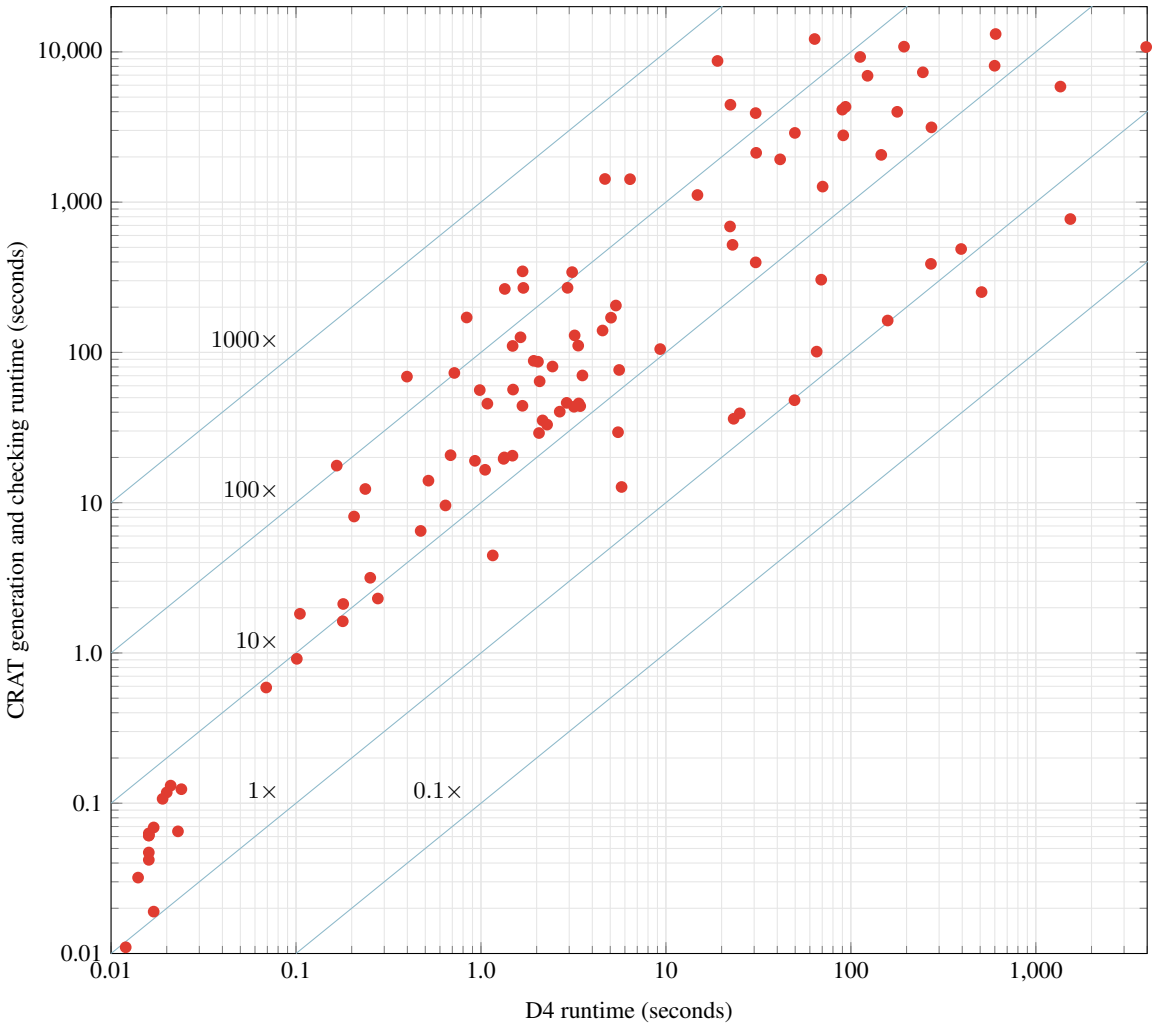

Fig. 3. Combined runtime for CRAT proof generation and checking as function of D4 runtime

Computing Counts and Weighted Counts

- Implemented package to represent numbers of form $a \cdot 2^b \cdot 5^c$.
  - For unweighted counts, first compute density and then scale by 2^n
  - For unweighted counts, perform exact decimal arithmetic
  - The set of numbers of this form is not closed under a reciprocal operation, but the weights in the competion are either of the form $w(x) + w(\overline{x}) = 1.0$ or $w(x) = w(\overline{x}) = 1$. The latter case sums to 2, with reciprocal $1/2$.
- Can implement having a be integer of arbitrary size and b and c being 32-bit integers.
- Weighted counts ranged up to 260,909 digits.

Benefits of Optimizations

Two optimizations were implemented within the CRAT framework:

Lemmas: Nodes with indegree greater than one were validated with separate lemmas, avoiding the possibly exponential expansion of the POG into a tree during the proof construction.

Literal grouping: For a product node with multiple literal arguments, those that cannot be validated by BCP are combined as arguments to a newly defined product node. The extension variable associated with this node is validated, and the proof is then used to validate the literals. This reduces the number of calls to the SAT solver to at most one per product node.

We conducted a set of experiments using the 80 benchmark instances for which the total number of proof clauses when both optimizations are performed was limited to 10^7. We then made three additional runs of the CRAT generator and checker, with each of the two optimizations disabled individually and in combination. Figures 4–5 illustrates the effect of the two optimizations. Each set of points shows the number of clauses when both optimizations are performed for the X value and the number with one or both of these optimizations disabled as the Y value.

The ratios between the number of clauses when either or both optimizations is disabled versus when both are enabled can be summarized as:

Without lemmas: Min 1.00, Max 52.54, Harmonic mean 2.11
Without literal grouping: Min 0.84, Max 39.60, Harmonic mean 1.18
Without either optimization: Min 0.99, Max 52.54, Harmonic mean 2.95

Here are other ways to summarize the impact on proof length:

- None of the 80 benchmarks require more than 10^7 proof clauses when both optimizations are used. Without lemmas, 19 rise above that threshold. Without literal merging, 7 due. Without either, 25 rise above the threshold. Two of the benchmarks require more than 10^8 proof clauses without lemmas.
- Of the 80 benchmark problems evaluated, and comparing with both optimizations disabled, 55 had the number of proof clauses reduced by at least 10% by using lemmas, 19 by using literal merging, and 69 by enabling both optimizations.

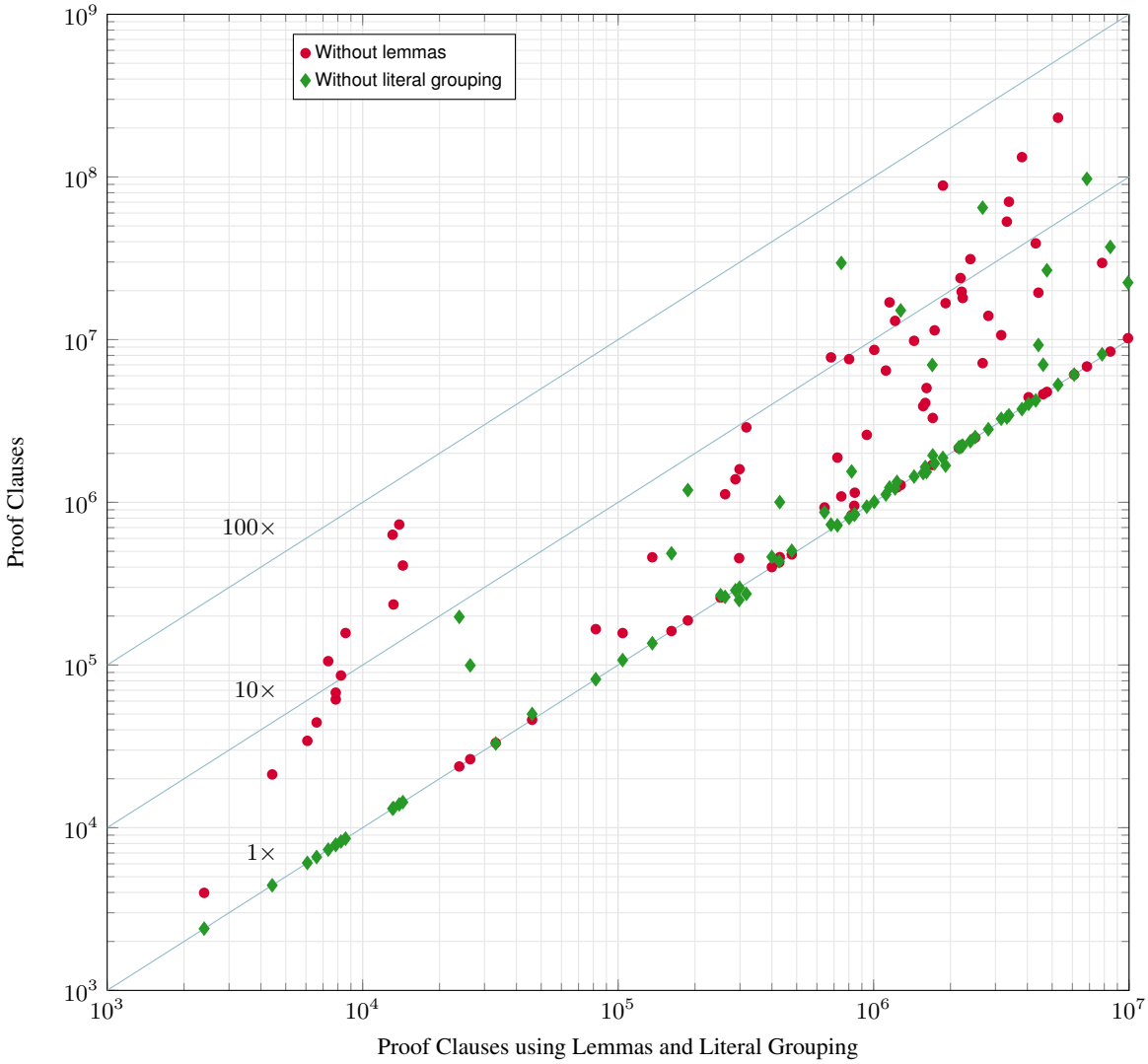

Fig. 4. Proof Clauses when Single Optimization Disabled

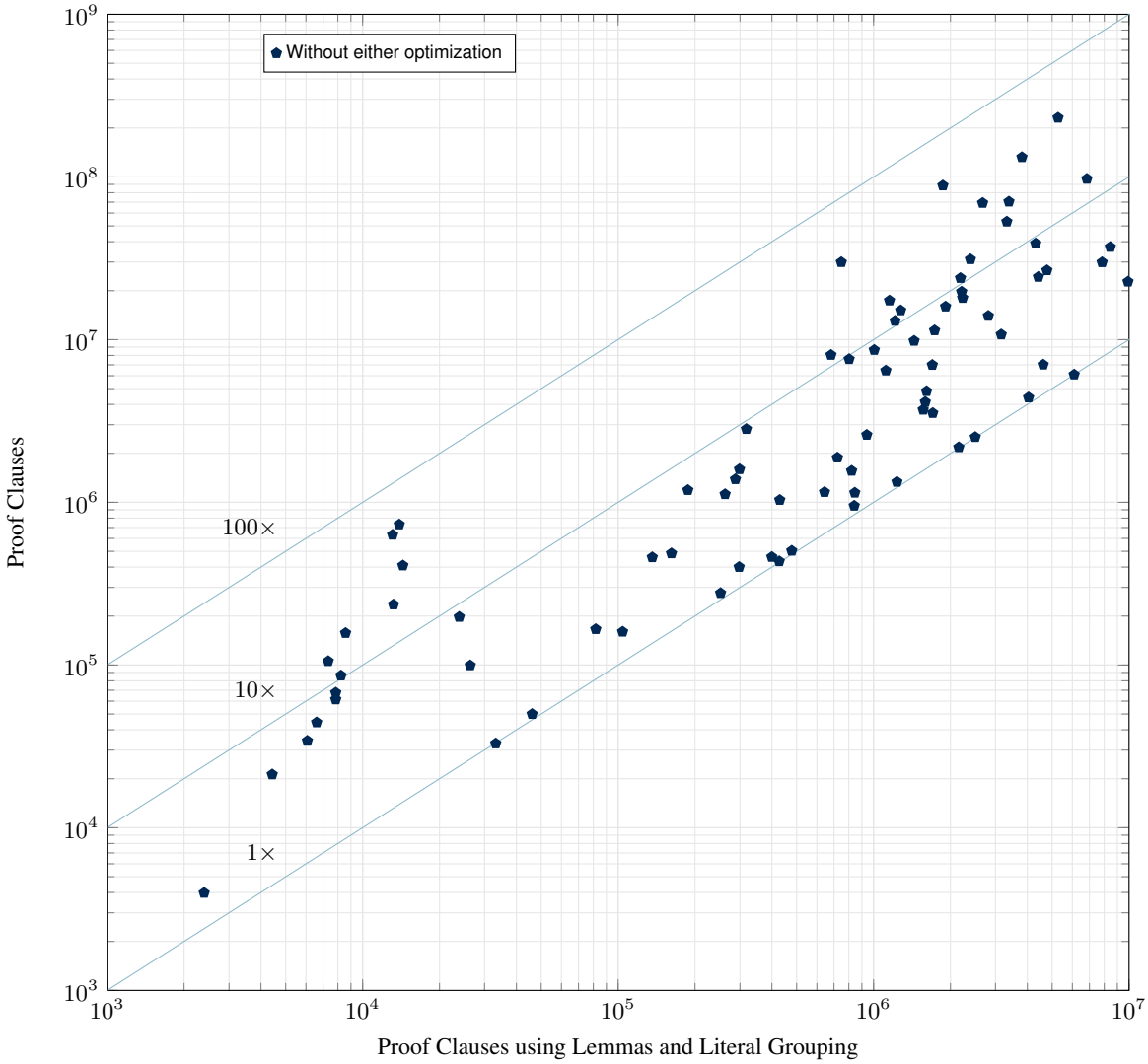

Fig. 5. Proof Clauses Both Optimizations are Disabled

The time performance is less dramatic, but still significant. Comparing the sum of CRAT generation and checking times for each of the four combinations of options, we get the following times relative to the one with both optimizations enabled:

Without lemmas: Min 0.85, Max 37.98, Harmonic mean 1.80
Without literal grouping: Min 0.83, Max 3.92, Harmonic mean 1.12
Without either optimization: Min 0.91, Max 41.97, Harmonic mean 2.19

Here is another way to summarize the impact on the combined time to generate and validate the CRAT file:

- Of the 80 benchmark problems evaluated, and comparing with both optimizations disabled, 50 had the time reduced by at least 10% by using lemmas, 27 by using literal merging, and 63 by enabling both optimizations.