Model Counting Competition Data Format (version 1.1)*

Johannes K. Fichte Markus Hecher Arijit Shaw

June 25, 2024

Abstract

Propositional model counting asks to compute the number of satisfying assignments (models) of a propositional formula. Over the past years, various solving techniques have been established and implemented into combinatorial solvers, such as SAT-based solving with caching, knowledge compilation, approximate solving utilizing sampling by SAT solvers, or dynamic programming. Following the improvements in solving the problem, a practical competition was conceived in 2019. The first competition started in the Spring of 2020 and has been repeated annually as part of the International Conference on Theory and Applications of Satisfiability Testing (SAT) Competitions. The 2021 iteration of the competition introduced a DIMACS-like data format that is still in use. This document extends the format and provides minor changes, including support for additional versions of the counting problem.

More information on the competition can be found at `modelcounting.org`.

1 Introduction

The input format remains compatible with the 2021 Competition format [FH21]. In addition, we introduce the following updates for the 2024 iteration:

- Input supports projected weighted model counting instances.
  We introduced an experimental track already in 2022 and use that format.
- The solver needs to support the type line and either handle the problem according to the type line or output an error.
  Supporting the problem type only via commandline parameters is considered insufficient.
- We allow negative and zero weights for weighted instances. If the solver cannot handle negative weights, a format error must be returned.

*Last updated June 6, 2024 for the Bonus Track on Weighted Model Counting of the 2024 Model Counting Competition.

- The output format supports fractional numbers.
- Approximation guarantees are included in the output format.

In the following description, we copy the previous format description and color changes in blue.

2 Input Format (DIMACS-like)

In the Model Counting Competition 2021, we use a DIMACS CNF-like input format [TCC+93]. The format extends the format used in SAT competitions [JBRS12, BFH+20][1] by introducing statements for weights and projections similar as in Cachet [KS05] or Ganak [SRSM19].

The following description gives an idea on the expected input format:

```
c c this is a comment and will be ignored
c c REPRODUCIBILITY LINE MANDATORY FOR SUBMITTED BENCHMARKS
c r originUrl/doi descUrl/doi [generatorUrl/doi]
c c HEADER AS IN DIMACS CNF
p cnf n m
c c OPTIONAL MODEL COUNTING HEADER
c t mc|wmc|pmc|pwmc
c c PROBLEM SPECIFIC LINES for wmc|pmc
c p ...
c p ...
c c CLAUSES AS IN DIMACS CNF
-1 -2 0
2 3 -4 0
c c this is a comment and will be ignored
4 5 0
4 6 0
```

In more details (note that we print symbols in typewriter font):

- Line separator is the symbol \n. Expressions are separated by space.
- A line starting with character `c r` is a comment. The solver may ignore it. The line aims for open data and reproducibility of the submitted instances expressing the origin of the benchmarks. The line will be added after competition when publishing the instances. `originUrl/doi` states where the instance can be downloaded (either as URL or DOI). `descUrl/doi` links to a description of the benchmarks. `generatorUrl/doi` provides an optional URL/DOI to where a problem generator can be found.
- A line starting with character `c t` is a comment. The starting character will be followed by `mc`, `wmc`, `pmc`, or `pwmc` indicating possible problem

[1] See, for example, `http://www.satcompetition.org/2009/format-benchmarks2009.html`

specific lines starting with `cs` in the file. If present, the line will occur prior to problem specific line. The solver needs to support the line and either handle the problem or output an error.

- A line starting with character `c p` provides problem specific details for weighted or projected model counting. Lines may occur anywhere in the file. We assume that lines are consistent and provide no contradicting information. We provide more details on its meaning below.
- Lines starting with character `c` *followed by* a second *character differing* from `p` and `s` are comments and can occur anywhere in the file. For convenience, we provide comments by lines starting with `c c`.
- Variables are consecutively numbered from `1` to `n`.
- The problem description is given by a unique line of the form `p cnf NumVariables NumClauses` that we expect to be the first line (except comments). More precisely, the line starts with character p (no other line may start with p), followed by the problem descriptor `cnf`, followed by number `n` of variables followed by number `m` of clauses each symbol is separated by space each time.
- The remaining lines indicate clauses consisting of decimal integers separated by space. Lines are terminated by character `0`. The Line `2 -1 3 0\n` indicates the clause "2 or not 1 or 3" ($v_2 \vee \neg v_1 \vee v_3$). If more lines than announced are present, the solver shall return a parser error and terminate without solving the instance.
- Empty lines or lines consisting of spaces/tabs only may occur and can be ignored.

Weighted Model Counting For weighted model counting, we introduce optional problem specific lines:

```
c p weight  1 0.4 0
c p weight -1 0.6 0
c p ...
```

In more details (note that we print symbols in typewriter font):

- The weight function is given by lines of the form `c p weight` ℓ w_ℓ `0` defining the weight w_ℓ for literal ℓ, where $0 \leq w_\ell$. The weight will be given as floating point (e.g., `0.0003`) with at most 9 significant digits after the decimal point, or in 32-bit scientific floating point notation (e.g., `1.23e+4`), or as fraction (e.g., `3/10`) consisting of two integers separated by the symbol `/`.
- We expect that the submission tests instances and output if it cannot be handled correctly. For example, if an instance specifies an unexpected

number of significant digits, larger floating point values, or large fractional values. In such a case, the solver should output a parsing error as specified in the return code section.

- We will provide only instances where both w_ℓ and $w_{\neg\ell}$ are given.
- ~~We will provide only instances where $0 \leq w_\ell \leq 1$ and $w_{\neg\ell} + w_\ell = 1$.~~
- If a weight $0 < w_\ell < 1$ is defined for a literal ℓ but $\neg\ell$ is not given, we assume $w_{\neg\ell} = 1 - w_\ell$ (assume that $\neg\neg a = a$). In other words:

  ```
  c p weight  1 0.4 0
  ```

  or

  ```
  c p weight -1 0.6 0
  ```

  are the same as

  ```
  c p weight  1 0.4 0
  c p weight -1 0.6 0
  ```

  The solver should output a warning that explicitly states how the weight is set.
- If a weight $w_\ell \leq 0$ is defined for a literal ℓ, the weight for literal $\neg\ell$ *must* be given as well. Otherwise, the solver *must* output a format error and abort.
- If a weight $w_\ell = 0$ is given, the solver *should* should output a warning stating which weight is 0.
- The solver should output an error on instances that cannot be handled or are not properly tested, e.g., if the solver can only handle weights w_ℓ, $w_{\neg\ell} > 0$ and $w_\ell + w_{\neg\ell} = 1$.
- ~~If the solver can handle weights $w_\ell > 1$ or $w_\ell + w_{\neg\ell} > 1$ unless $w_\ell = w_{\neg\ell} = 1$, the solver must output a warning. Otherwise, the solver must output an error on those instances.~~
- For compatibility, with #SAT we say that if for a variable x, there is neither a weight for x nor $\neg x$ given, it is considered 1.
  Note: this differs from the format used in Cachet.

Projected Model Counting For projected model counting, we introduce optional problem specific lines:

```
c c INDICATES VARIABLES THAT SHOULD BE USED
c p show varid1 varid2 ... 0
....
c c MORE MIGHT BE GIVEN LATER
c p show varid7 varid23 ... 0
```

In more details (note that we print symbols in typewriter font):

- Projection variables will be given by lines starting with `c p show` followed by the identifier of the variables. Lines describing to add variables to a projection set may occur anywhere in the files and will be terminated by symbol `0`.

Note: if all variables are stated using `show`, we consider model counting. if no variables are stated the problem is simply to decide satisfiability.

3 Output Format

We expect that the solver outputs result to `stdout` in the following format.

```
c o This output describes a result of a run from
c o a [weighted|projected|projected weighted] model counter.
c o
c o The following line keeps backwards compatibility
c o with SAT solvers and avoids underflows if result is 0.
c o MANDATORY
s SATISFIABLE|UNSATISFIABLE|UNKNOWN
c o The following solution line is optional.
c o It allows a user to double check whether a solver
c o that provides multiple options was called correct.
c o
c o MANDATORY
c s type [mc|wmc|pmc|pwmc]
c o The solver has to output an estimate in scientific notation
c o on the solution size, even if it was an exact solver.
c o MANDATORY
c s [log10-estimate|neglog10-estimate] VALUE
c o The solver outputs the solution in its highest precision.
c o MANDATORY
c s SOLVERTYPE PRECISION NOTATION VALUE
c o OPTIONAL
c s pac guarantees epsilon: E delta: D
c o InternalValueForOpt=X
c o Internal=X
```

In more details:

- While the solver may use a line staring with `c` for comments in the output, we suggest to use lines starting with `c o` to indicate a comment.
- The solver has to announce whether the instance is satisfiable or unsatisfiable by a line starting with `s` followed by `SATISFIABLE` or `UNSATISFIABLE`. The solver may output `UNKNOWN`, but is not allowed to output any other value than these three if a line starting with `s` is present.
- The solver has to announce an estimate on the solution by a line starting with `c s [neg]log10-estimate VALUE` where `VALUE` is a string representing the result `cnt` (model count/ weighted model count/ projected model count/ projected weighted model count) in $\log_{10}$-Notation (see Section 4) of double precision, i.e., 15 significant digits. More precisely:
  We let `VALUE` $:= \log_{10}(|$`cnt`$|)$
  If `cnt` > 0, output "`c s log10-estimate VALUE`".
  If `cnt` $= 0$, output "`c s log10-estimate -inf`".
  If `cnt` < 0, output "`c s neglog10-estimate VALUE`".
- The output is has to be announced by a line starting with `c s` followed by strings indicating the solver type, the precision, the notation, and the result. The solver developer may use for `SOLVERTYPE` the following strings: `approx`, `exact`, heuristic. In place of `PRECISION`, the solver developer has to specify which the internal precision the solver used, allowed values are `arb`, `single`, `double`, `quadruple`, or other values according to IEEE754 Standard. For `NOTATION`, the developer may chose `log10`, `float`, `prec-sci`, `int`, or `frac`. Then, in place of `VALUE`, the solver has to output its computed result.
  If the solver outputs an approximate solution, we suggest that the solver additionally provides the guarantees. For example, by outputting the following line: `c s pac guarantees epsilon: 0.1 delta: 0.2`
- The solver may output a result in $\log_{10}$ *notation* which is similar to the format used in the probabilistic inference competitions UAI [GRS+16].
- If the solver consists of a run script, which calls a pre-processor, consists of multiple phases, or consists of a solving portfolio, we expect the developer to output by a comment line which tool was started, what parameters it used, and when the tool ended. Preferably as follows:

```
c o CALLS(1) ./preproc -parameters
c o stat CALL1 STARTED RFC3339-TIMESTAMP
...
c o stat CALL1 FINISHED RFC3339-TIMESTAMP
c o CALLS(2) ./postproc -parameters
...
```

- We suggest that the solver outputs internal statistics by lines starting with `c o DESCRIPTION=VALUE` or `c DESCRIPTION : VALUE`.

4 Examples

The following sections provide a few brief examples for each track with expected input and output.

Model Counting

Example 1. *The following text describes the CNF formula (set of clauses)*

$$\{\{\neg x_1, \neg x_2\}, \{x_2, x_3, \neg x_4\}, \{x_4, x_5\}, \{x_4, x_6\}\}.$$

```
c c This file describes a DIMACS-line CNF in MC 2021 format
c c The instance has 6 variables and 4 clauses.
p cnf 6 4
c t mc
-1 -2    0
 2  3 -4 0
c c This line is a comment and can be ignored.
4 5 0
c The line contains a comment and can be ignored as well.
4 6 0
```

A solution is given as follows, but can also be modified according to the technique of the solver:

```
c o This file describes a solution to a model counting instance.
s SATISFIABLE
c s type mc
c o The solver log10-estimates a solution of 22.
c s log10-estimate 1.342422680822206
c o Arbitrary precision result is 22.
c s exact arb int 22
```

Weighted Model Counting

Example 2. *The following text describes the CNF formula (set of clauses)*

$$\{\neg x_1, \neg x_2\}, \{x_2, x_3, \neg x_4\}, \{x_4, x_5\}, \{x_4, x_6\}\}$$

with weight function $\{x_1 \mapsto 0.4, \neg x_1 \mapsto 0.6, x_2 \mapsto 0.5, \neg x_2 \mapsto 0.5, x_3 \mapsto 0.4, \neg x_3 \mapsto 0.6, x_4 \mapsto 0.3, \neg x_4 \mapsto 0.7, x_5 \mapsto 0.5, \neg x_5 \mapsto 0.5, x_6 \mapsto 0.7, \neg x_6 \mapsto 0.3\}$.

```
c c This file describes a weighted CNF in MC 2021 format
c c with 6 variables and 4 clauses
p cnf 6 4
c t wmc
c c Weights are given as follows, spaces may be added
```

```
c c to improve readability.
c p weight  1 0.4 0
c p weight  2 0.5 0
c p weight  3 0.4 0
c p weight  4 0.3 0
c p weight  5 0.5 0
c p weight  6 0.7 0
-1 -2 0
 2  3 -4 0
c this is a comment and will be ignored
 4  5 0
 4  6 0
c same
```

The solution should be given in the following format (modified according to the technique of the solver):

```
c o This file describes a solution to a weighted
c o model counting instance.
s SATISFIABILE
c s type wmc
c o This file describes that the weighted model count is 0.345
c o
c s log10-estimate -0.460924
c s exact double float 0.346
```

Example 3 (Optional)**.** *The following text describes the CNF formula (set of clauses)*

$$\{\neg x_1, x_2\}, \{x_3, \neg x_2\}, \{x_2, x_1\}, \{x_3, x_2\}\}$$

with weight function $\{x_1 \mapsto 0.1, \neg x_1 \mapsto 0.1, x_2 \mapsto 0.1, \neg x_2 \mapsto 0.9, x_3 \mapsto 0.0235, \neg x_3 \mapsto 0.0125\}$ *including a problem description line and two comments.*

```
c c This file describes a weighted CNF in MC 2021 format
c c with 3 variables and 4 clauses
p cnf 3 4
c t wmc
-1  2 0
 3 -2 0
 2  1 0
 3  2 0
c p weight  1 0.1
c p weight -1 0.1
c p weight  2 0.1
c p weight  3 0.0235
c p weight -3 0.0125
```

The solution should be given in the following format:

```
c o WARNING
c o  L9:Sum of positive and negative literal is not equal to 1.
c o WARNING
c o  L12:Sum of positive and negative literal is not equal 1.
c o This file describes a solution to a weighted
c o model counting instance.
s SATISFIABILE
c s type wmc
c o This file describes that the weighted model count is
c o 0.000470000000000000532907051822
c s type wmc
c s log10-estimate -3.327902142064282
c s exact arb log10 -3.32790214206428248634352698 91
```

Projected Model Counting

Example 4. *The following text describes the CNF formula (set of clauses)*

$$\{\neg x_1, \neg x_2\}, \{x_2, x_3, \neg x_4\}, \{x_4, x_5\}, \{x_4, x_6\}\}$$

with projection set $\{x_1, x_2\}$ *including a problem description line and two comments.*

```
c c This file describes a projected CNF in MC 2021 format
c c with 6 variables and 4 clauses and 2 projected variables
p cnf 6 4 2
c t pmc
c p show 1 2
-1 -2 0
 2  3 -4 0
cc this is a comment and will be ignored
 4  5 0
 4  6 0
```

A solution can be given in the following format:

```
c o This file describes that the projected model count is 3
s SATISFIABILE
c s log10-estimate 0.47712125471966
c s type pmc
c s exact arb int 3
```

Output $\log_{10}$-Notation

We say that an output is in $\log_{10}$-Notation if the result of the problem is the value v, but the solver outputs $\log_{10}(v)$.

References

[BFH+20] Tomáš Balyo, Nils Froleyks, Marijn J.H. Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors. *Proceedings of SAT Competition 2020: Solver and Benchmark Descriptions*. University of Helsinki, Department of Computer Science, 2020.

[FH21] Johannes K. Fichte and Markus Hecher. Model counting competition 2021: Call for benchmarks/participation. `https://mccompetition.org/assets/files/2021/competition2021.pdf`, 2021.

[GRS+16] Vibhav Gogate, Tahrima Rahman, Somdeb Sarkhel, David Smith, and Deepak Venugopal. Uai 2016 inference evaluation. `http://www.hlt.utdallas.edu/~vgogate/uai16-evaluation/tuning.html`, 2016.

[JBRS12] Matti Järvisalo, Daniel Le Berre, Olivier Roussel, and Laurent Simon. The international SAT solver competitions. In *AI Magazin*. The AAAI Press, 2012.

[KS05] Henry Kautz and Tian Sang. Model counting using component caching and clause learning. `https://www.cs.rochester.edu/u/kautz/Cachet/cachet-wmc-1-21.zip`., 2005.

[SRSM19] Shubham Sharma, Subhajit Roy, Mate Soos, and Kuldeep S. Meel. GANAK: A scalable probabilistic exact model counter. In Sarit Kraus, editor, *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19)*, pages 1169–1176, Macao, China, 2019. IJCAI.

[TCC+93] Michael Trick, Vavsek Chvatal, Bill Cook, David Johnson, Cathy McGeoch, and Bob Tarjan. The 2nd DIMACS implementation challenge: 1992–1993 on NP hard problems: Maximum clique, graph coloring, and satisfiability. `http://archive.dimacs.rutgers.edu/pub/challenge/sat/benchmarks/`, 1993.