WFOMC Manual

Guy Van den Broeck, Wannes Meert
July 12, 2013

WFOMC is an experimental tool to perform weighted model counting in a lifted manner. The tool is compatible with Markov Logic and Factor Graphs.

Table of Contents

1. Usage

Usage:

```
java -jar ./wfomc-2.0.jar [OPTIONS] input
```

For example, to perform inference on the example model `sickdeath.fg` for the query `death`, you use the following command:

```
java -jar ./wfomc-2.0.jar -q "death" models/sickdeath.fg
```

MLNs can be queried as follows:

```
java -jar ./wfomc-2.0.jar -q "smokes(Guy)" models/friendsmoker.mln
```

1.1. Options

Input files:

`--fg`	Force to read in file as a factor graph model
`--mln`	Force to read in file as MLN model
`--wmc`	Force to read in file as WMC model
`--train filename`	Database file with training data. (May be specified multiple times.)
`--test filename`	Database file with test data. (May be specified multiple times.)

Querying:

`-q query` `--query query`	Query atom (if none given, shows all marginals).
`--rcr`	Perform "Relax, Compensate and Recover" approximate inference (all marginals).

Learning:

`--wl`	Learn weights (expects a database file given)
`--ll`	Database likelihood (only for MLN)
`--pll`	Database pseudo likelihood (only for MLN, only for verification purposes)

Settings:

`--propinf`	Perform inference on the propositional level using the c2d compile of Darwiche. The c2d compiler command can be set with environment variable `C2DCMD` (default: `./c2d_linux`).
`--mln-nodist`	Associate the weight with the entire formula. Do not distribute weight over equivalent CNF formula. (not supported by weight learning yet)

Veryfication and feedback:

`--pdf`	Create a pdf visualizing the smoothed NNF. Requires pdflatex and graphviz dot to be in your path and the dot2texi package installed.

`--verify`	Verify the result of wfomc using the c2d compiler of Darwiche. The c2d compiler command can be set with environment variable `C2DCMD` (default: `./c2d_linux`).
`--verbose`	Verbose output on command line and in pdf

Output formats:

`--dimacs-out`	Output a given MLN to a ground DIMACS file.
`--fastinf-out`	Output a given MLN to the format used by the FastInf tool.
`--mln-out`	Output a given MLN to a ground MLN.

Various:

`-h` `--help`	Show help about the available flags.

1.2. Input formats

WFOMC supports three types of input formats. Internally, all formats are translated to the weighted model counting format:

- **Weighted Model Counting** format as defined below.
  The extension `.wmc` is associated with this filetype.
- **Markov Logic Networks** as defined for Alchemy.
  There are two possible transformation from MLNs to WMC. The first is the transformation to CNF as performed by Alchemy which distributes the weights over the equivalent CNF clauses. This is the default transformation. The second transformation assigns the weight to the formula in its whole and can be activated with the flag `--mln-nodist` (This is similar to how Primula system interprets MLN formulas).
  The extension `.mln` is associated with this filetype.
- **Factor Graphs** format as defined for FOPI.
  The extension `.fg` is associated with this filetype.

1.1. Dependencies

- For verifying the correctness of the results:
  - c2d compiler: Adnan Darwiche's c2d compiler.
    Used for propositional inference and verification. The binary is assumed to be installed as `./c2d_linux`. This can be overridden with the environment variable `C2DCMD`.
- For visualizing the d-DNNFs:
  - pdflatex
    `pdflatex` is assumed to be in your path.
  - dot2tex
    `dot2tex` is assumed to be in your path.
  - dot2texi
    LaTeX package assumed to be installed.
  - Graphviz
    The dot2tex tool expects Graphviz `dot` to be available on your path.
- Included in package:
  - Argot: Used for command line parsing (BSD licensed)
  - Factorie: Used for optimization algorithms (Apache 2.0 licensed)

2. Download

The WFOMC binaries can be downloaded from http://dtai.cs.kuleuven.be/wfomc/. Installing is not necessary, the zip-file contains a runnable jar-file.

3. Weighted Model Counting Input Format

More information about the semantics can be found in the publications mentioned below. A theory consists out of three parts:

- Domain declarations
- Predicate declarations
- Formulas

3.1. Domain Declarations

```
domain {Name} {size} \{{DomainElements}\}
```

Where `Name` is the name of the domain and should start with a capital letter and has a given size, `size`. It is also possible to give some domain elements explicitely by name as a comma separated list in `DomainElements`. For example,

```
domain D 10 {}
```

3.2. Predicate Declarations

```
predicate {predicate} {TrueWeight} {FalseWeight}
```

Where a predicate consists out of a predicate name (lowercase first letter) and optionally some arguments (e.g., `f(D,D)`). A weight can be given for when the predicate is true, `TrueWeight`, and when it is false, `FalseWeight`. For example,

```
predicate p(D)
predicate r
predicate f(D,D) 0.51 1
```

3.3. Formulas

The third part are the first-order formulas, one per line, in CNF. Optionally you can add domain constraints to a line separated by a comma. Logic variables should start with a capital letter.

Logic connectives:

- Negation: `!` or `¬`
- Disjunction: `v`, `V` or `|`
- Conjunction: `newline` (conjunctive normal form)

Domain connectives:

- Equality: `=`
- Inequality: `!=` or `≠`

For example:

```
!p(X) v !p(Y)   v !r v f(X,Y), X != Y
p(X)  v !f(X,Y)              , X != Y
p(Y)  v !f(X,Y)              , X != Y
r     v !f(X,Y)              , X != Y
```

4. Tutorial on Inference

```
# Example theory file
$ cat models/friendsmoker.mln
person = {Guy, Nima, Wannes, Jesse, Luc}
Friends(person,person)
Smokes(person)
2 Friends(x,y) ^ Smokes(x) => Smokes(y)
```

```
# Run a query on the theory
$ java -jar ./wfomc-2.0.jar -q "Smokes(Guy)" ./models/friendsmoker.mln
Reading file using MLN syntax.
Compilation took 815 ms
evidence nnf size = 18
evidence smooth nnf size = 24
query nnf size = 33
query smooth nnf size = 45

Inference took 15 ms
evidence logWmc = 68.63908810719217 = log(6.450259808376127E29)
query logWmc = 67.94594092663222 = log(3.225129904188057E29)

P(Some(Smokes(Guy))) = 0.49999999999999906

# Visualize the circuit in a pdf
$ java -jar ./wfomc-2.0.jar --pdf ./models/friendsmoker.mln
$ open ./nnfs/theory.smooth.nnf.pdf

# Visualize the circuit in a pdf in a verbose way (show all steps)
$ java -jar ./wfomc-2.0.jar --pdf --verbose ./models/friendsmoker.mln
$ open ./nnfs/theory.smooth.nnf.pdf
```

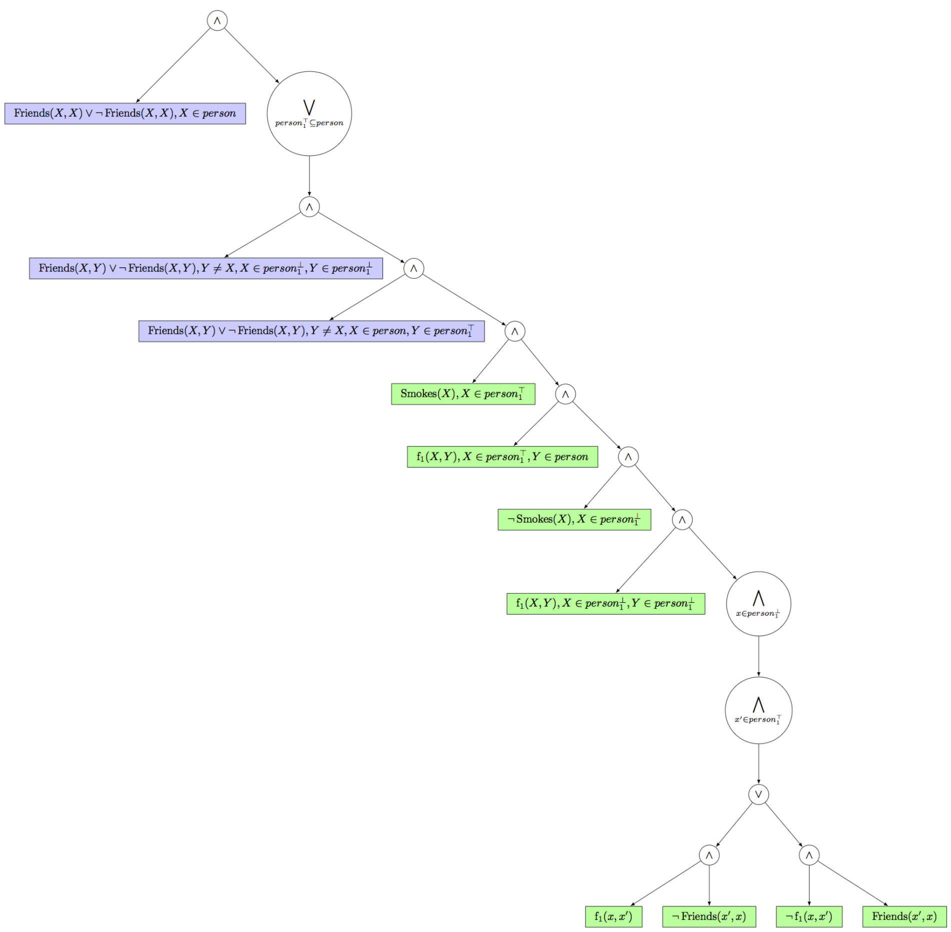

Figure 1: Part of the visualization of the compilation of friendsmokers (non-verbose).

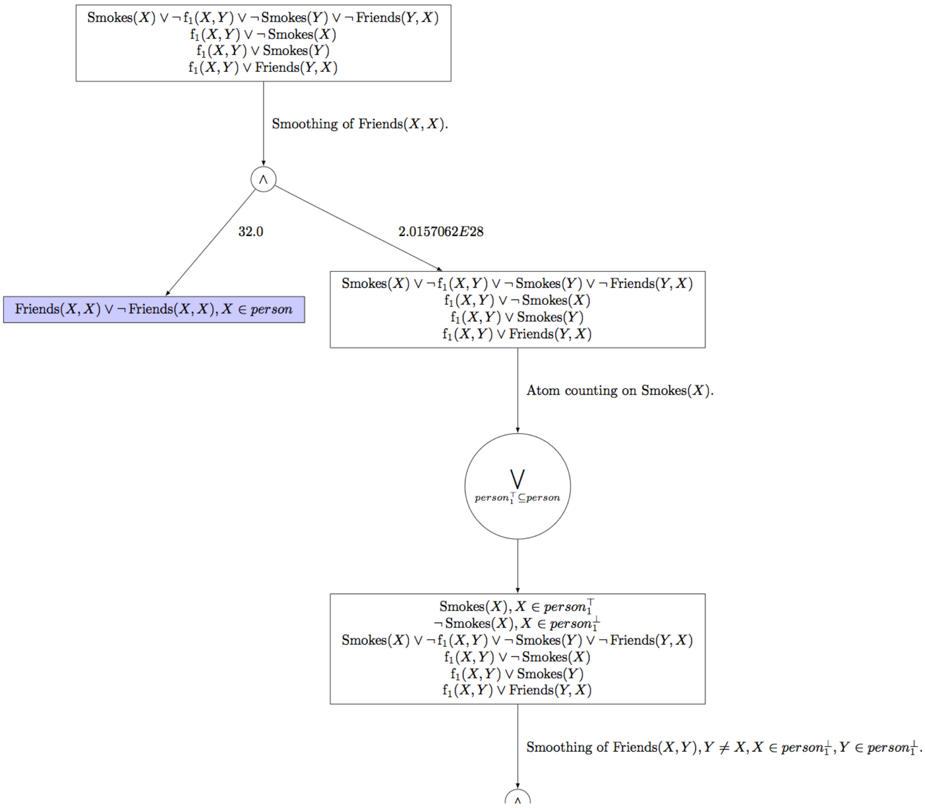

Figure 2: Part of the visualization of the compilation of friendsmokers (verbose).

5. Tutorial on Learning

To learn the weights of a theory, use the `--wl` flag and indicate the training database(s) with the `--train` flag.

```
$ java -jar ./wfomc-2.0.jar --wl
    --train models/learning/smoking/smoking-train.db
    models/learning/smoking/smoking.mln
```

6. Publications

The algorithms implemented are explained in the following publications:

- G. Van den Broeck, W. Meert and J. Davis. **Lifted Generative Parameter Learning**. In Proceedings of the 3rd International workshop on Statistical Relational AI (StarAI), held at the 27th AAAI Conference, 2013.
- G. Van den Broeck. **Lifted Inference and Learning in Statistical Relational Models**. PhD dissertation KU Leuven, 2013. https://lirias.kuleuven.be/handle/123456789/373041
- G. Van den Broeck and J. Davis. **Conditioning in First-Order Knowledge Compilation and Lifted Probabilistic Inference**. In Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI), 2012. https://lirias.kuleuven.be/handle/123456789/345667
- G. Van den Broeck, A. Choi, A. Darwiche. **Lifted relax, compensate and then recover: From approximate to exact lifted probabilistic inference**. In Proceedings of the conference on Uncertainty in Artificial Intelligence (UAI), 2012 https://lirias.kuleuven.be/handle/123456789/351575
- M. Jaeger, G. Van den Broeck. **Liftability of probabilistic inference: Upper and lower bounds**. In Proceedings of the 2nd International Workshop on Statistical Relational AI (StarAI), 2012. https://lirias.kuleuven.be/handle/123456789/352388
- W. Meert, G. Van den Broeck, N. Taghipour, D. Fierens, H. Blockeel, J. Davis, L. De Raedt. **Lifted inference for probabilistic programming**. In Proceedings of the NIPS Probabilistic Programming Workshop, 2012. https://lirias.kuleuven.be/handle/123456789/369419
- G. Van den Broeck. **On the completeness of first-order knowledge compilation for lifted probabilistic inference**. In Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS), 2011 https://lirias.kuleuven.be/handle/123456789/316338
- G. Van den Broeck, N. Taghipour, W. Meert, J. Davis, and L. De Raedt. **Tutorial on Lifted Inference in Probabilistic Logical Models**. On the 22th International Joint Conference on Artificial Intelligence (IJCAI), 2011. https://lirias.kuleuven.be/handle/123456789/317055
- G. Van den Broeck, N. Taghipour, W. Meert, J. Davis, and L. De Raedt. **Lifted probabilistic inference by first-order knowledge compilation**. In Proceedings of the 22th International Joint Conference on Artificial Intelligence (IJCAI), 2011. https://lirias.kuleuven.be/handle/123456789/308265

7. Contact

http://dtai.cs.kuleuven.be/wfomc/

Main contact:

Guy Van den Broeck
Departement of Computer Science, KU Leuven
Celestijnenlaan 200A

3001 Leuven
België
guy.vandenbroeck@cs.kuleuven.be
http://www.guyvdb.eu

Contributors:

- Guy Van den Broeck
- Wannes Meert
- Jesse Davis
- Jan Van Haaren

8. License

The Lesser GNU Public License, Version 3.0
http://www.gnu.org/licenses/lgpl-3.0.txt