cplint and PITA

Fabrizio Riguzzi

cplint [5] and PITA [7] allow various forms of Probabilistic Logic Programming. They accepts the language of Logic Programs with Annotated Disjunctions (LPADs)[11, 12] and CP-logic programs [9, 10]. They share the following syntax for input programs.

1 Syntax

Disjunction in the head is represented with a semicolon and atoms in the head are separated from probabilities by a colon. For the rest, the usual syntax of Prolog is used. For example, the LPAD clause $h_1 : p_1 \vee \ldots \vee h_n : p_n \leftarrow b_1, \ldots, b_m, \neg c_1, \ldots, \neg c_l$ is represented by

```
h1:p1 ; ... ; hn:pn :- b1,...,bm,\+ c1,....,\+ cl
```

No parentheses are necessary. The pi are numeric expressions. It is up to the user to ensure that the numeric expressions are legal, i.e. that they sum up to less than one.

If the clause has a single head with probability 1, the annotation can be omitted and the clause takes the form of a normal prolog clause.

The coin example of [12] is represented as

```
heads(Coin):1/2 ; tails(Coin):1/2:-  toss(Coin),\+biased(Coin).
heads(Coin):0.6 ; tails(Coin):0.4:-  toss(Coin),biased(Coin).
fair(Coin):0.9 ; biased(Coin):0.1.
toss(coin).
```

2 cplint

cplint consists of three Prolog modules for answering queries using goal-oriented procedures.

lpadsld.pl: computes the probability of a query using the top-down procedure described in in [5] and [6]. It is based on SLDNF resolution and is an adaptation of the interpreter for ProbLog [2].

lpad.pl: computes the probability of a query using a top-down procedure based on SLG resolution [1]. As a consequence, it works for any sound LPADs, i.e., any LPAD such that each of its instances has a two valued well founded model.

cpl.pl: computes the probability of a query using a top-down procedure based on SLG resolution and moreover checks that the CP-logic program is valid, i.e., that it has at least an execution model.

2.1 Installation

cplint is distributed in source code in the git version of Yap. It includes Prolog and C files. Download it by following the instruction in http://www.ncc.up.pt/~vsc/Yap/downloads.html.

cplint uses GLib 2.0 and CUDD. GLib is a standard GNU package so it is easy to install it using the package management software of your Linux distribution.

To install CUDD, follow the instructions at `http://vlsi.colorado.edu/~fabio/CUDD/` to get the package. After decompressing, you will have a direcory `cudd-2.4.2` with various subdirectories. Compile CUDD following the included instructions.

Install Yap together with `cplint`: when compiling Yap following the instuction of the `INSTALL` file in the root of the Yap folder and use the `configure` option `--enable-cplint=DIR` where `DIR` is the path to the directory `cudd-2.4.2` (including `cudd-2.4.2`).

After having performed `make install` you can do `make installcheck` that will execute a suite of tests of the various programs. If no error is reported you have a working installation of `cplint`.

2.2 Commands

All modules accept the same commands for reading in files and answering queries. The LPAD or CP-logic program must be stored in a text file with extension `.cpl`. Suppose you have stored the example above in file `coin.cpl`. In order to answer queries from this program, you have to run Yap, load one of the modules (such as for example `lpadsld.pl`) by issuing the command

```
:- use_module(library(lpadsld)).
```

at the command prompt. Then you must parse the source file `coin.cpl` with the command

```
:- p(coin).
```

if `coin.cpl` is in the current directory.

At this point you can pose query to the program by using the predicate `s/2` (for solve) that takes as its first argument a conjunction of goals in the form of a list and returns the computed probability as its second argument. For example, the probability of the conjunction `head(coin),biased(coin)` can be asked with the query

```
:- s([head(coin),biased(coin)],P).
```

For computing the probability of a conjunction given another conjunction you can use the predicate `sc/3` (for solve conditional) that take takes as input the query conjunction as its first argument, the evidence conjunction as its second argument and returns the probability in its third argument. For example, the probability of the query `heads(coin)` given the evidence `biased(coin)` can be asked with the query

```
:- sc([heads(coin)],[biased(coin)],P).
```

3 PITA: Probabilistic Inference with Tabling and Answer subsumption

"Probabilistic Inference with Tabling and Answer subsumption" (PITA) [7, 8] allows various forms of Probabilistic Logic Programming and Possibilistic Logic Programming. It accepts the language of LPADs and CP-logic programs.

PITA computes the probability of queries by tranforming the input program into a normal logic program and then calling a modified version of the query on the transformed programs.

3.1 Installation

PITA, as `cpliint`, uses GLib 2.0 and CUDD. Follow the instructions in Section 2.1 to install them.

PITA is a package of XSB Prolog. To install PITA, run XSB `configure` in the `build` directory with the option `--with-pita=DIR` where `DIR` is the folder where CUDD is.

3.2 Commands

If you want to use inference on LPADs, load PITA in XSB with

```
:- [pita].
```

load you program, say `coin.cpl`, with

```
:- load(coin).
```

and compute the probability of query atom `heads(coin)` by

```
:- prob(heads(coin),P).
```

`load(file)` reads `file.cpl`, translates it into a normal program, writes the result in `file.P` and loads `file.P`.

PITA offers also the predicate `parse(infile,outfile)` which translates the LPAD in `infile` into a normal progam and writes it to `outfile`.

Moreove, you can use `prob(goal,P,CPUTime,WallTime)` that returns the probability of goal `P` together with the CPU and wall time used.

In case the modeling assumptions of PRISM hold, i.e.: (1) the probability of a conjunction (A, B) is computed as the product of the probabilities of A and B (independence assumption), (2) the probability of a disjunction $(A; B)$ is computed as the sum of the probabilities of A and B (exclusiveness assumption), you can perform faster inference with an optimized version of PITA in package `pitaindexc.P`. It accepts the same commands of `pita.P`. `pitaindexc.P` simulates PRISM and does not need CUDD and GLib.

If you want to compute the Viterbi path and probability of a query (the Viterbi path is the explanation with the highest probability) as with the predicate `viterbif/3` of PRISM, you can use package `pitavitind.P`.

The package `pitacount.P` can be used to count the explanations for a query, provided that the independence assumption holds. To count the number of explanations for a query use

```
:- count(heads(coin),C).
```

`pitacount.P` does not need CUDD and GLib.

3.3 Possibilistic Logic Programming

PITA can be used also for answering queries to possibilistic logic program [4], a form of logic progamming based on possibilistic logic [3]. The package `pitaposs.P` provides possibilistic inference. You have to write the possibilistic program as an LPAD in which the rules have a single head whose annotation is the lower bound on the necessity of the clauses. To compute the highest lower bound on the necessity of a query use

```
:- poss(heads(coin),P).
```

`pitaposs.P` does not need CUDD and GLib.

References

[1] Weidong Chen and David Scott Warren. Tabled evaluation with delaying for general logic programs. *J. ACM*, 43(1):20–74, 1996.

[2] L. De Raedt, A. Kimmig, and H. Toivonen. ProbLog: A probabilistic Prolog and its application in link discovery. pages 2462–2467, 2007.

[3] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of logic in artificial intelligence and logic programming,vol. 3*, pages 439–514. Oxford University Press, 1994.

[4] Didier Dubois, Jérôme Lang, and Henri Prade. Towards possibilistic logic programming. In *International Conference on Logic Programming*, pages 581–595, 1991.

[5] Fabrizio Riguzzi. A top down interpreter for LPAD and CP-logic. In *Congress of the Italian Association for Artificial Intelligence*, volume 4733 of *LNAI*, pages 109–120. Springer, 2007.

[6] Fabrizio Riguzzi. Extended semantics and inference for the Independent Choice Logic. *Logic J. of the IGPL*, 17(6):589–629, 2009.

[7] Fabrizio Riguzzi and Terrance Swift. Tabling and Answer Subsumption for Reasoning on Logic Programs with Annotated Disjunctions. In *Technical Communications of the International Conference on Logic Programming*, volume 7 of *LIPIcs*, pages 162–171. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010.

[8] Fabrizio Riguzzi and Terrance Swift. The PITA system: Tabling and answer subsumption for reasoning under uncertainty. *Theory and Practice of Logic Programming, 27th Int'l. Conference on Logic Programming (ICLP'11) Special Issue*, 2011.

[9] J. Vennekens, M. Denecker, and M. Bruynooghe. Representing causal information about a probabilistic process. In *Proceedings of the 10th European Conference on Logics in Artificial Intelligence*, LNAI. Springer, September 2006.

[10] J. Vennekens, Marc Denecker, and Maurice Bruynooghe. CP-logic: A language of causal probabilistic events and its relation to logic programming. *Theory Pract. Log. Program.*, 9(3):245–308, 2009.

[11] J. Vennekens and S. Verbaeten. Logic programs with annotated disjunctions. Technical Report CW386, K. U. Leuven, 2003.

[12] J. Vennekens, S. Verbaeten, and M. Bruynooghe. Logic programs with annotated disjunctions. volume 3131 of *LNCS*, pages 195–209. Springer, 2004.