Probabilistic Logic Programs

`Dealing with uncertainty by Probabilistic Reasoning`.

1. Introduction

Up to now, we discussed only about logic formula, that can be either true or false. However, sometimes, we do not have the ability to define their truth values; we have to take in account also the **probabilities** that an event occurs.

Prolog is a powerful programming language that allows us to represent and reason upon some given knowledge. *Why don't we expand Prolog to include probabilities as well*?

This is the basic idea about Probabilistic Logic Programming: since the notion of meta-interpreter, we can design a totally new Prolog machine that allows us to include also probabilities inside the reasoning schema.

There are already two systems that follow this approach:

- **ProbLog**.
- **LPAD** (Logic Programs with Annotated Disjunctions).

2. Probabilistic Logic Programming

A probabilistic logic program defines a **probability distribution** over normal logic programs, and each of them is a part of the distribution, named **possible world** or simply **world**.

In LPAD the clauses have a different structure. The **head** is extended with disjunctions and each **disjunct** is annotated with a probability (the head of the clause is the probability distribution of random variable).

Example

Assuming these sample points:

- *people with flu* also *sneeze in 70% of the cases*.
- *people with hay fever* also *sneeze in 80% of the cases*.

The respective snippet of code is:

```
sneezing(X): 0.7 ; null: 0.3 :- flu(X).
sneezing(X): 0.8 ; null: 0.2 :- hay_fever(X).
```

For each case we have an associated probability. But, *what happens if we get both*? *Which is the probability*? We add to the knowledge base other informations, such as:

- *people with flu* also *sneeze in 70% of the cases*.
- *people with hay fever* also *sneeze in 80% of the cases*.
- Bob has flu.
- Bob has hay fever.

Again, the snippet becomes:

```
sneezing(X): 0.7 ; null: 0.3 :- flu(X).
sneezing(X): 0.8 ; null: 0.2 :- hay_fever(X).
flu(bob).
hay_fever(bob).
```

Let's assume a Closed World Assumption, everything we know is already stated in the database program. We can obtain the *possible worlds* (normal logic programs that determine the probability distribution) by selecting an atom from the head of the clause (the clause must be already matched). The steps to follow are:

1. Ground the free variables with constants.
2. Select an atom from the head of each clause.
3. The total number of possible worlds is equal to 2^N (N is the amount of clauses stated).

Matching all the free variables with constants:

```
sneezing(bob) :- flu(bob).
sneezing(bob) :- hay_fever(bob).
flu(bob).
hay_fever(bob).
```

Defining all the possible worlds:

1st

```
null :- flu(bob).
sneezing(bob) :- hay_fever(bob).
flu(bob).
hay_fever(bob).
```

2nd

```
sneezing(bob) :- flu(bob).
null :- hay_fever(bob).
flu(bob).
hay_fever(bob).
```

3rd

```
null :- flu(bob).
sneezing(bob) :- hay_fever(bob).
```

```
flu(bob).
hay_fever(bob).
```

4th

```
null :- flu(bob).
null :- hay_fever(bob).
flu(bob).
hay_fever(bob).
```

Finally, we got more simple rules moving out the probabilities. However, we can infer the probability of each possible world, combining together the original ones:

- 1st $P(w_1) = 0.7 \times 0.8$.
- 2nd $P(w_2) = 0.7 \times 0.2$.
- 3rd $P(w_3) = 0.3 \times 0.8$.
- 4th $P(w_4) = 0.3 \times 0.2$.

These results are obtained by more formal steps, divided into:

- **Atomic Choice**. Selecting an atom from the head of the clause:

  $$(C, \theta, i)$$

  where:

  - C is the clause.
  - θ is the substitution of the free variables.
  - i is the index of the atom selected.

- **Compositive Choice**. The **set k** of atomic choices done, its probability is computed by:

  $$P(k) = \prod_{(C,\theta,i)} P(C, i).$$

- **Selection** σ. A **total** composite choice, we have chosen an atomic choice for each inner clause. This selection generates a possible world (a normal logic program part of the probability distribution). Its probability is given by the following formula:

  $$P(w_\sigma) = P(\sigma) = \prod_{(C,\theta,i)\in\sigma} P(C, i).$$

In general, given a query Q and a world w:

$$P(Q|w) = \begin{cases} 1 & \text{if } Q \text{ is true in } w \\ 0 & \text{altrimenti} \end{cases}$$

Given the query `sneezing(bob)`, we deduce that it's true in the first, second and third possible world. The probability of `sneezing(bob)` is equal to the sum of the possible worlds probabilities in which `sneezing(bob)` is true.

$$P(\text{sneezing(bob)}) = 0.56 + 0.21 + 0.14 = 0.93$$
