Theorem 5. Let \( KB = (O, M) \) be a TCP ontology, where \( M \) is a cMLN. Deciding \( Pr_{KB}(a) \geq k \) is PP-hard in the data complexity.

The complexity class PP contains problems decidable by a probabilistic Turing machine in polynomial time, with error probability less than 1/2; like \( \# P \), a polynomial time Turing machine with a PP oracle can solve all problems in the polynomial hierarchy [21]. This negative result does not prevent us, however, from tractably comparing the probabilities of two worlds.

Proposition 1. Let \( M \) be a cMLN, and \( \lambda_1 \) and \( \lambda_2 \) be possible worlds. Deciding whether \( Pr_M(\lambda_1) \leq Pr_M(\lambda_2) \) is in \( PTIME \) in the data complexity.

The basic intuition behind this result is that we can compute the term \( n_i(x) \) in Equation 1 in polynomial time; since the denominator in this equation is the same for all worlds, \( Pr_M(\lambda_1) \leq Pr_M(\lambda_2) \) can be decided by only computing the numerators in this equation.

Example 5. Consider the following worlds relative to the MLN in Example 2 and the grounding from Example 4:

\[
\begin{align*}
\lambda_1 &= \{ \text{canLabel}(f, \ell), \text{hor}(\ell, f), \neg\text{left}(\ell, f), \text{adj}(\ell, f), \\
&\quad \text{right}(\ell, f), \neg\text{top}(\ell, f), \neg\text{nw}(\ell, f), \neg\text{ver}(\ell, f) \}; \\
\lambda_2 &= \{ \text{canLabel}(f, \ell), \neg\text{hor}(\ell, f), \neg\text{left}(\ell, f), \text{adj}(\ell, f), \\
&\quad \neg\text{right}(\ell, f), \text{top}(\ell, f), \neg\text{nw}(\ell, f), \text{ver}(\ell, f) \}.
\end{align*}
\]

A quick inspection of the formulas in the MLN allows us to conclude that \( \lambda_1 = \neg f_1 \land f_2 \land \neg f_3 \land f_4 \), while \( \lambda_2 = \neg f_1 \land \
eg f_2 \land f_3 \land \neg f_4 \). Therefore, taking into account the weights of each formula, we can see that \( Pr(\lambda_1) > Pr(\lambda_2) \); in fact, even though we cannot (tractably) compute the actual probabilities, we can conclude that \( Pr(\lambda_1) = \frac{2^n}{12} Pr(\lambda_2) \), or that \( \lambda_1 \) is approximately 2.7 times more probable than \( \lambda_1 \).

4.2 An Anytime Algorithm

The results in the previous section point the way towards Algorithm anytimeRank for answering ranking queries; the pseudocode for this algorithm is shown in Figure 4; in the rest of this section, we will discuss its properties.

Algorithm anytimeRank takes a probabilistic ontology in which the MLN is assumed to be a ground cMLN (recall that the grounding can be computed in polynomial time in the data complexity); the other input corresponds to a stopping condition that can be based on whatever the user considers important (time, number of steps, number of inspected worlds, etc); cf. Section 4.2.2 for a discussion on ways to define the stopping condition based on properties of the output offering correctness guarantees.

The main while loop in line 3 iterates through the set of equivalence classes relative to \( M \). Subroutine compMostProbEqClass, invoked in line 4, computes the \( i \)-th most probable equivalence class. Note that this can simply be done by taking the formulas in \( M \) and sorting them with respect to their weights; the classes are then generated by keeping track of a Boolean vector of which formulas are true and which are false. The next while loop, in line 7, is in charge of going through the current equivalence class. Subroutine computePossWorld takes the current class and a set of already inspected worlds and computes a new world (not in \( S \)). This can be done as described in the proof sketch of Theorem 3; in particular, the possible combinations of atoms in formula neg can be traversed in order, without the need to explicitly keep track of a set like \( S \). The final lines of this loop take the computed world and obtain the atomic consequences from the (non-probabilistic) induced subontology (line 10), and adds the score to each such atom (lines 11 and 12). The score of a class consists of \( e \) to the power of the sum of the weights of formulas that are true in that class. Line 13 updates the output set of atoms. Finally, set out is returned in decreasing order of score.

Example 6. Consider the probabilistic \( \mathcal{EL}^{**} \) ontology \( \Phi = (O, M) \), where \( O \) is given as follows:

\[
\forall X p(X) \rightarrow q(X) : \{ \{m(X), 1\}, \{n(X), 0\} \};
\]

\[
p(a) : \{ \{m(a), 1\} \};
\]

\[
p(b) : \{ \{n(b), 1\} \};
\]

\[
p(c) : \{ \{m(c), 1\}, \{n(c), 0\} \},
\]

and \( M \) is given by \( \{ \{m(X), 1.5\}, \{n(X), 0.8\} \} \). Suppose we ground \( M \) with the set of constants \( \{a, b, c\} \), yielding:

\[
f_1 : (m(a), 1.5), \quad f_2 : (n(a), 0.8), \quad f_3 : (m(b), 1.5), \quad f_4 : (n(b), 0.8), \quad f_5 : (m(c), 1.5), \quad f_6 : (n(c), 0.8).
\]

This setup therefore yields \( 2^6 = 64 \) equivalence classes (in this case, we have exactly one world per class). Figure 5 shows a subset of these classes in decreasing order of score (as they will be inspected by Algorithm anytimeRank).

The algorithm will proceed as follows; cf. Figure 5 for the description of the classes (we use classes \( C_i \) to denote the single world \( \lambda_i \) in that class):

\[
\begin{align*}
O_{C_1} &= \{ p(a), p(b) \}; \quad \text{add } e^{0.9} \text{ to } score(p(a)) \text{ and } score(p(b)); \\
O_{C_2} &= \{ p(a), p(b), p(c), q(c) \}; \quad \text{add } e^{0.1} \text{ to } score(p(a)), score(p(b)), score(p(c)), \text{ and } score(q(c)); \\
O_{C_3} &= \{ p(a) \}; \quad \text{add } e^{0.5} \text{ to } score(p(a)); \\
O_{C_4} &= \{ p(a), p(c), q(c) \}; \quad \text{add } e^{0.1} \text{ to } score(p(a)), score(p(c)), \text{ and } score(q(c)); \\
O_{C_5} &= \{ p(a), p(b), q(a) \}; \quad \text{add } e^{0.3} \text{ to } score(p(a)), score(p(b)), \text{ and } score(q(a)).
\end{align*}
\]

If the algorithm is stopped at this point, the scores are as follows (approximate, and in descending order):

\[
p(a) : 2530.18, \quad p(b) : 1638.47, \quad p(c) : 891.71, \quad q(c) : 891.71
\]

4.2.1 Correctness and Running Time of anytimeRank

The correctness of this algorithm lies in the fact that if all classes are inspected, the returned output set is clearly the answer to the ranking query. In the general case, only a