Gibbs Sampling in Probabilistic Description Logics with Deterministic Dependencies

Oliver Gries and Ralf Möller

Hamburg University of Technology*
21073 Hamburg, Germany

Abstract. In many applications there is interest in representing both probabilistic and deterministic dependencies. This is especially the case in applications using Description Logics (DLs), where ontology engineering usually is based on strict knowledge, while there is also the need to represent uncertainty. We introduce a Markovian style of probabilistic reasoning in first-order logic known as Markov logic and investigate the opportunities for restricting this formalism to DLs. In particular, we show that Gibbs sampling with deterministic dependencies specified in an appropriate fragment remains correct, i.e., probability estimates approximate the correct probabilities. We propose a Gibbs sampling method incorporating deterministic dependencies and conclude that this incorporation can speed up Gibbs sampling significantly.

1 Introduction

While probabilistic languages often represent uncertain evidence and exceptions, there is also the need to further represent deterministic knowledge. As a tradeoff, there are probabilistic formalisms that allow for the representation of near-deterministic knowledge, i.e., knowledge represented with probabilities approximating 0 or 1. However, since knowledge representation with logic in its origin is based on strict formulas, we believe it is important to consider the perspective in which uncertainty is an additional feature to deterministic knowledge. This perspective is especially suited for Description Logics (DLs) [3], where ontology engineering usually is based on specifying strict taxonomies with strict disjointness and strict domain and range restrictions on roles, and where the tradeoff between complexity and expressivity is well-studied.

Gibbs sampling [2, 4, 5] is a Markov chain Monte Carlo (MCMC) method to estimate a conditional probability distribution by generating samples from a simpler distribution. Since Markov chains constructed with Gibbs sampling are known to be regular, estimates of probabilities are known to be correct in the long run. Though this method is often used in practice, it is insufficient for logic in the presence of deterministic dependencies, since in this case the correct flow of the

* This work is supported by the European Union project CASAM (FP7-217061).

chain usually is broken down. Similarly, Gibbs sampling with near-determinism involves transitions with probabilities approximating 0, leading to unacceptably long convergence times (cf. [7]). In order to solve this problem, in [7] the MC-SAT algorithm is proposed. MC-SAT samples new states with respect to a set of auxiliary variables. However, while MC-SAT is a powerful method to compute conditional probabilities, it is restricted to near-determinism.

In this paper, we propose a formalism based on Markov logic [6] and show that by restricting the deterministic part of the knowledge to a fragment of $\mathcal{ALH}$, Markov chains constructed with Gibbs sampling remain regular. To the best of our knowledge, until now there has not been a language discovered allowing for global deterministic dependencies in Gibbs sampling. We present a Gibbs sampling method incorporating these dependencies and conclude that this incorporation can speed up Gibbs sampling significantly.

In Sect. 2, we introduce probabilistic knowledge representation and Markov networks. In Sect. 3, the formalism of Markov logic [6] is presented and extended to incorporate deterministic dependencies. Further, we define the language $\mathcal{ALH}^-$ for the representation of determinism. Then, in Sect. 4, we introduce to problems with Gibbs sampling in knowledge representation. In Sect. 5, we propose a Gibbs sampling algorithm incorporating deterministic dependencies in $\mathcal{ALH}^-$. Finally, in Sect. 6 we summarize the results.

2 Preliminaries

For the representation of probabilistic and deterministic knowledge, we will focus on Description Logics (DLs). We assume the reader to be familiar with the syntax and semantics of DLs [3] and first-order logic [4].

2.1 Probabilistic Knowledge Representation

The basic notion of probabilistic knowledge representation formalisms is the so-called *random experiment*. A *random variable* X is a function assigning a value to the result of a random experiment. In the sequel, we will use only Boolean random variables with values 1 or 0 (*true* or *false*, respectively).

Let $\boldsymbol{X} = \{X_1, ..., X_n\}$ be the ordered set of all random variables of a random experiment. An *event* $\boldsymbol{X} = \boldsymbol{x}$ is an assignment $X_1 = x_1, ..., X_n = x_n$ to all random variables, and a certain vector of values $\boldsymbol{x}$ is referred to as a *possible world*. If it is clear from the context, we write x as an abbreviation for $X = \mathit{true}$ and $\neg x$ as an abbreviation for $X = \mathit{false}$. A possible world can be associated with a probability $P(\boldsymbol{X} = \boldsymbol{x}) = p$, where p is a real value in $[0, 1]$.[1]

A *distribution* $\mathbf{P}(X)$ of a random variable X is a mapping from the domain of X to probability values in $[0, 1]$ such that the values of X sum up to 1. Distributions can be defined for (ordered) sets of random variables as well. A mapping $\mathbf{P}(X_1, \ldots, X_k)$ from the domain of a set of random variables to the

[1] We assume the reader to be familiar with Kolmogorov's axioms of probability.

k-dimensional cross product of $[0,1]$ such that all combinations of values sum up to 1 is called *joint distribution*. A *full joint distribution* $\mathbf{P}(X_1,\ldots,X_n)$ is a joint distribution where all random variables of a random experiment are involved. Let $\Omega = \{\boldsymbol{x}_1,...,\boldsymbol{x}_r\}$ be the set of all possible worlds. In order to specify a full joint distribution $\mathbf{P}(X_1,\ldots,X_n)$, probabilities $P(\boldsymbol{X}=\boldsymbol{x}_i)$ for all $\boldsymbol{x}_i$ must be given such that $\sum_{i=1}^{r} P(\boldsymbol{X}=\boldsymbol{x}_i) = 1$. The expression $\mathbf{P}(X_1,\ldots,X_m,x_{m+1},\ldots,x_l)$ denotes an m-dimensional distribution where the values of $X_{m+1},...,X_l$ have been fixed. In slight misuse of notation, we sometimes write $\boldsymbol{e}$ for these fixed values.

The *conditional probability distribution of X given evidence* $\boldsymbol{e}$ is defined by $\mathbf{P}(X \mid \boldsymbol{e}) = \frac{\mathbf{P}(X,\boldsymbol{e})}{P(\boldsymbol{e})} = \alpha\,\mathbf{P}(X,\boldsymbol{e}) = \alpha <P(x,\boldsymbol{e}), P(\neg x,\boldsymbol{e})>$, where α is a normalizing constant. Note that $\mathbf{P}(X \mid \boldsymbol{e})$ is only defined, if $P(\boldsymbol{e}) > 0$.

The most common query types in probabilistic knowledge representation are the *conditional probability query*, i.e., the computation of $\mathbf{P}(X \mid \boldsymbol{e})$ and the *maximum a posteriori (MAP) query*, where the objective is to find the most likely assignment of values to random variables $X_1,...,X_m$ given evidence $\boldsymbol{e}$, i.e., the computation of $argmax_{X_1,\ldots,X_m}\mathbf{P}(X_1,...,X_m \mid \boldsymbol{e})$. In this paper, we will focus on the former query type.

2.2 Markov Networks

Since the representation of $\mathbf{P}(X_1,\ldots,X_n)$ in principle requires the specification of 2^n probability values, there is interest in formalisms with a less complex representation. The main idea to achieve a more compact representation is that one can exploit independence assumptions. Usually, this is the case in graph-based formalisms, where the representation of the full joint probability distribution can be decomposed into different factors. Besides Bayesian networks, the most important graph-based formalism is the formalism of Markov networks [1,2].

A *Markov network graph* is a tuple $\boldsymbol{G} = (\boldsymbol{X},\boldsymbol{E})$, where $\boldsymbol{X} = \{X_1,...,X_n\}$ is a set of nodes corresponding to the random variables of the domain and $\boldsymbol{E}$ is a set of undirected edges (X_i,X_j), $i \neq j$, between these nodes. A *clique* C is a subgraph of G, whose nodes are all adjacent to each other. Let $\boldsymbol{X}_C$ be the set of nodes contained in C. A clique C is called maximal, if there is no other clique C_i with $\boldsymbol{X}_C \subset \boldsymbol{X}_{C_i}$. Further, $\boldsymbol{C} = \{C_1,...,C_m\}$ is a set of cliques of $\boldsymbol{G}$ consisting of all nodes of $\boldsymbol{X}$, i.e., $\boldsymbol{X}_{C_1} \cup ... \cup \boldsymbol{X}_{C_m} = \boldsymbol{X}$.

A *Markov network* $\mathcal{M} = (\boldsymbol{G},\boldsymbol{F})$ consists of a Markov network graph $\boldsymbol{G}$ and a set $\boldsymbol{F}$ which is comprised of non-negative real-valued functions f_i for each clique $C_i, i = 1,...,m$ in $\boldsymbol{G}$. A full joint probability distribution to be expressed can be decomposed into factors f_i (cf. [1,2]) such that

$$P(\boldsymbol{X}=\boldsymbol{x}) = \frac{1}{Z}\prod_{i}^{m} f_i(\boldsymbol{x}_{C_i}) \tag{1}$$

where Z is a normalizing constant summing over the products in (1) for all possible worlds $\boldsymbol{x}$ ensuring that $\sum_{k=1}^{r} P(\boldsymbol{X}=\boldsymbol{x}_k) = 1$. Note that each f_i does only depend on the values of random variables corresponding to its clique C_i.

3 Markov Description Logics

3.1 Markov Logic

The formalism of Markov logic [6] provides a means to combine the formalism of Markov networks with the expressivity of first-order logic. A knowledge base in Markov logic is called a *Markov logic network* $MLN = (\mathcal{F}, \mathcal{W})$. It consists of a multiset of first-order formulas $\mathcal{F} = \{F_1, ..., F_p\}$ and a multiset of real number weights $\mathcal{W} = \{w_1, ..., w_p\}$ such that each w_i is associated to F_i. For simplicity, we use the notation of a set of weighted formulas $w_i\, F_i$. An example Markov logic network is $MLN_1 = \{4\, \forall x\, P(x) \rightarrow Q(x), 1.1\, P(i)\}$, where i is a constant.

Let $\Gamma = \{c_1, ..., c_s\}$ be the set of all constants mentioned in $\mathcal{F}$. A *grounding* of a formula F_i is a substitution of all variables in the matrix of F_i with constants from Γ (this corresponds to a domain closure).[2] A Markov logic network MLN can be converted to a (finite) set of weighted ground clauses $Cl = \{cl_1, ..., cl_m\}$. Each atom appearing in Cl is referred to as a *ground atom*. The set of all these ground atoms corresponds to a set of Boolean random variables $\boldsymbol{X} = \{X_1, ..., X_n\}$. Consequently, for each MLN with a fixed set of constants Γ, there is a set of possible worlds $\boldsymbol{x}$. When a world $\boldsymbol{x}$ does not satisfy a formula, the idea is to ensure that this world is less probable rather than impossible as in first-order logic.

For each MLN there is a corresponding Markov network $\mathcal{M} = (\boldsymbol{G}, \boldsymbol{F})$, with $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{E}_{MLN})$, where $\boldsymbol{E}_{MLN}$ is the set of pairs of ground atoms (X_i, X_j) appearing together in at least one cl_i and a function $f_i \in \boldsymbol{F}$ for each $cl_i \in Cl$. Note that each weighted ground clause cl_i corresponds to a (not necessarily maximal) clique C_i. In notation slightly differently from [6], we specify the full joint distribution of a MLN with (1), where

$$f_i(\boldsymbol{x}_{C_i}) = \begin{cases} exp(w_i), & \text{if } \boldsymbol{x}_{C_i} \text{ satisfies } cl_i \\ 1, & \text{otherwise} \end{cases} \tag{2}$$

Note that $\prod_{i=1}^{m} exp(w_i) = \prod_{i=1}^{m} exp(ln\, f_i(\boldsymbol{x}_{C_i})) = exp(-\sum_{i=1}^{m} -ln\, f_i(\boldsymbol{x}_{C_i}))$, where the last term often is used in the context of statistical physics with $-ln\, f_i(\boldsymbol{x}_{C_i})$ called an energy function [2, 5]. The advantage of using exp in Markov logic is that $P(\boldsymbol{X} = \boldsymbol{x}) > 0$ and that it is possible to specify $w \in \mathbb{R}$.

3.2 Exploiting DLs: Incorporating Deterministic Constraints

There is often interest to represent a domain with both deterministic and probabilistic dependencies. While in [6] deterministic dependencies are approximated by assigning large weights to formulas, we propose to incorporate deterministic constraints to Markov logic in the context of DLs.

Definition 1. *A Markov DL knowledge base* KB_M *is a tuple* $(\mathcal{T}, \mathcal{A})$*, where* $\mathcal{T}$ *is comprised of sets* $\mathcal{T}_{det}$ *and* $\mathcal{T}_w$ *of deterministic resp. weighted axioms and* $\mathcal{A}$ *is comprised of sets* $\mathcal{A}_{det}$ *and* $\mathcal{A}_w$ *of deterministic resp. weighted assertions.*

[2] An existentially quantified formula is replaced by a disjunction of its groundings.

Under consideration of a domain closure, corresponding Markov networks are similar to the ones introduced in Sect. 3.1. However, in Markov DL knowledge bases $Cl = \{cl_1, ..., cl_m\}$ is the set of weighted and deterministic ground clauses. We specify the full joint probability distribution of KB_M with (1), where

$$f_i(\boldsymbol{x}_{C_i}) = \begin{cases} exp(w_i), & \text{if } cl_i \text{ is weighted and } \boldsymbol{x}_{C_i} \text{ satisfies } cl_i \\ 0, & \text{if } cl_i \text{ is deterministic and } \boldsymbol{x}_{C_i} \text{ does not satisfy } cl_i \\ 1, & \text{otherwise.} \end{cases} \quad (3)$$

An advantage of having both deterministic and probabilistic dependencies is that initial ontology engineering is done as usual with standard reasoning support and with the possibility to add weighted axioms and weighted assertions on top of the strict fundament. Since lots of possible worlds do not have to be considered because their probability is known to be 0, probabilistic reasoning can be significantly faster.

The language for representing $\mathcal{T}_w$ and $\mathcal{A}_w$ can be any DL in which it is reasonable to assume a fixed set of constants $\Gamma = \{c_1, ..., c_s\}$ such that it is possible to compute a finite set of weighted ground clauses. For the representation of deterministic knowledge, in this paper we use the language $\mathcal{ALH}^-$, a rather simple DL without an assertional component (i.e., we are not allowing for deterministic assertions). An $\mathcal{ALH}^-$ *knowledge base KB* consists of a set $\mathcal{T}$ of terminological axioms as depicted in Table 1, where A, B are atomic concepts and R, S are atomic roles with the additional restriction that equivalences as well as terminological cycles are not allowed. The semantics is defined as usual.

Table 1. Terminological axioms in $\mathcal{ALH}^-$

$A \sqsubseteq B$	concept inclusion	$\exists R.\top \sqsubseteq A$	domain restriction on roles
$A \sqsubseteq \neg B$	concept disjointness	$\top \sqsubseteq \forall R.A$	range restriction on roles
$R \sqsubseteq S$	role inclusion		

Example 1. Let $KB_M = (\{Lynx \sqsubseteq Animal, Animal \sqsubseteq \neg Plant\}, \{1.1\ Lynx(i), 0.6\ Plant(i)\})$, where $\mathcal{T}_w = \mathcal{A}_{det} = \{\}$. Under domain closure there are 2^3 possible worlds $\boldsymbol{x}_k$ (where e.g. *Lynx* is abbreviated with L):

$$\begin{aligned} \boldsymbol{x}_1 &= <L(i), A(i), P(i)> & \boldsymbol{x}_5 &= <\neg L(i), A(i), P(i)> \\ \boldsymbol{x}_2 &= <L(i), A(i), \neg P(i)> & \boldsymbol{x}_6 &= <\neg L(i), A(i), \neg P(i)> \\ \boldsymbol{x}_3 &= <L(i), \neg A(i), P(i)> & \boldsymbol{x}_7 &= <\neg L(i), \neg A(i), P(i)> \\ \boldsymbol{x}_4 &= <L(i), \neg A(i), \neg P(i)> & \boldsymbol{x}_8 &= <\neg L(i), \neg A(i), \neg P(i)> \end{aligned}$$

The full joint probability distribution is specified with respect to the set $Cl = \{\neg Lynx(i) \vee Animal(i),\ \neg Animal(i) \vee \neg Plant(i),\ 1.1\ Lynx(i), 0.6\ Plant(i)\}$. The normalizing constant $Z = 0 + exp(1.1) + 0 + 0 + 0 + 1 + exp(0.6) + 1 \approx 6.83$ such that e.g. $P(\boldsymbol{X} = \boldsymbol{x}_7) \approx \frac{exp(0.6)}{6.83} \approx 0.267$. In order to keep the example simple, we do not consider roles (but the general structure would be the same).

4 Gibbs Sampling for Reasoning about Knowledge

Answering conditional probability queries $\mathbf{P}(X \mid \boldsymbol{e})$ by simply applying the full joint probability distribution is intractable [4]. *Sampling* or *Monte Carlo*-algorithms avoid this problem by generating samples from a probability distribution which is much easier to compute. The objective is to approximate $\mathbf{P}(X \mid \boldsymbol{e})$. Sampling is like "coin flipping": Random numbers in $[0, 1]$ are generated and a variable X_i is assigned *true* resp. *false*, if the corresponding number is lower resp. greater than the respective probability in the distribution of X_i.

A *Markov chain (of length k)* is a sequence of states $\boldsymbol{x}_1, ..., \boldsymbol{x}_k$ (a state corresponds to a possible world) where each successing state only depends on the current state. *Markov chain Monte Carlo (MCMC)* algorithms [2,4] are a powerful class of sampling algorithms walking through the state space $\Omega = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_r\}$. With respect to an arbitrary order, each non-evidence variable $X_i, i = 1, ..., m$ is sampled. This process is repeated N-times such that $k = m \cdot N$. After $k - 1$ steps, the fractions of the number of states visited with $X = x$ resp. $X = \neg x$ are taken as the estimated probabilities of $\mathbf{P}(X \mid \boldsymbol{e})$.

Let $\overline{\boldsymbol{x}_i}$ be an assignment to $X_1, ..., X_{i-1}, X_{i+1}, ..., X_n$. *Gibbs sampling* [2,4,5] is a special case of MCMC, where the probability of a transition $T(\boldsymbol{x}_t \rightarrow \boldsymbol{x}_{t+1})$ is $T((x_i, \overline{\boldsymbol{x}_i}) \rightarrow (x_i', \overline{\boldsymbol{x}_i})) = P(x_i' \mid \overline{\boldsymbol{x}_i})$. In graph-based formalisms such as Markov networks, Gibbs sampling can be optimized by exploiting the graph structure: Let $C_1, ..., C_s$ be all cliques containing the node X_i. The *Markov boundary* of X_i is the set of its neighbours $MB(X_i) = \boldsymbol{X}_{C_1} \cup \ldots \cup \boldsymbol{X}_{C_s} \setminus \{X_i\}$. If the values of $MB(X_i)$ are known, denoted $mb(X_i)$, it shields X_i from influences of all other nodes in $\boldsymbol{G}$, i.e., $\mathbf{P}(X_i \mid x_1, ..., x_{i-1}, x_{i+1}, ..., x_n) = \mathbf{P}(X_i \mid mb(X_i)) = \alpha <P(x_i, mb(X_i)), P(\neg x_i, mb(X_i))>$.

A Markov chain is *regular* if there is a number v such that for all pairs of states $\boldsymbol{x}_i, \boldsymbol{x}_j$ the probability of getting from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$ in v steps is greater than 0 [2]. If a Markov chain is regular (in Markov networks if $f_i > 0$ for all $f_i \in \boldsymbol{F}$) the probability distribution of the states converges to a unique *stationary distribution* π. Finally, in the case of Gibbs Sampling, $\pi(\boldsymbol{x}_i)$ is known to be equal to $P(\boldsymbol{x}_i \mid \boldsymbol{e})$. Markov chains through Gibbs Sampling and deterministic dependencies usually are not regular and can break down the state graph into disconnected regions:

Example 2. Consider two ground atoms X_1, X_2 with the deterministic constraint $X_1 \equiv X_2$. The state graph for applying Gibbs sampling is depicted in Fig. 1. If the chain starts with $(0, 0)$, it will never reach $(1, 1)$. Consequently, $\pi(0, 0) = 1$ and $\pi(1, 1) = 0$, though there is no information of any preference.

Similarly, Gibbs sampling with near-determinism involves transitions with probabilities approximating 0, leading to unacceptably long convergence times (cf. [7]). Thus, since (near-)determinism is required in many applications, Gibbs sampling in general is not sufficient for reasoning about knowledge.

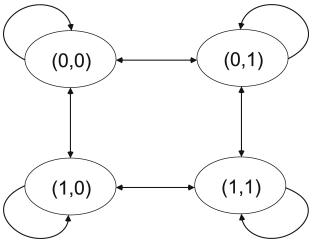

Fig. 1. Gibbs sampling state graph of two Boolean random variables

5 Gibbs Sampling with Deterministic Dependencies

Definition 2. *Let $\boldsymbol{D}$ be a set of deterministic dependencies. A Markov chain is regular with respect to $\boldsymbol{D}$ if there is a number v such that for all pairs of states $\boldsymbol{x}_i, \boldsymbol{x}_j$ both satisfying $\boldsymbol{D}$, the probability of getting from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$ in v steps is greater than* 0.

A Markov chain being regular with respect to $\boldsymbol{D}$ has a unique stationary distribution π, if the initial state x_1 satisfies $\boldsymbol{D}$, i.e., $P(\boldsymbol{X} = \boldsymbol{x_1}) > 0$. The difference to regular Markov chains defined in the previous section is that there are states $\boldsymbol{x}_i$ with $P(\boldsymbol{X} = \boldsymbol{x}_i) = 0$ that are simply not stepped into.

As can be seen from Example 2, the regularity of the Markov chain is broken, if X_1 and X_2 are constrained to be equal. This is also the case if they are constrained to be different. We will now show that it is possible to run a Markov chain constructed with Gibbs sampling that is regular with respect to deterministic constraints in $\mathcal{ALH}^-$.

Theorem 1. *Let KB_M be a Markov DL knowledge base where $\mathcal{T}_{det}$ is represented with $\mathcal{ALH}^-$ and $\mathcal{A}_{det}$ is empty. Then, a Markov chain constructed with Gibbs sampling is regular with respect to $\mathcal{T}_{det}$.*

Proof. In order to prove the regularity, it is important to figure out states satisfying or falsifying $\mathcal{T}_{det}$. Ground clauses derived from disjointness axioms are of the form $\neg At_1 \vee \neg At_2$, and ground clauses derived from all other axioms in $\mathcal{ALH}^-$ are of the form $\neg At_1 \vee At_2$, where At_1 and At_2 are ground atoms. Therefore, clauses derived from $\mathcal{T}_{det}$ are falsified only if At_1 is assigned 1 (*true*) and At_2 assigned 0 (*false*) or if both At_1 and At_2 are assigned 1. Since in $\mathcal{ALH}^-$ there are no cycles, an order $X_1 < ... < X_n$ can be defined for all ground atoms such that for all ground clauses $\neg At_1 \vee At_2$ no ground atom At_2 is lower in the order than At_1. Atoms only mentioned in clauses $\neg At_1 \vee \neg At_2$ can be added arbitrarily. Then, it is possible to construct a tree, where each node respects the defined order and corresponds to a state $(x_1, ..., x_n)$, and where the root node is a state where all atoms are assigned with 0, i.e., $(0, ..., 0)$. For every 0 in $(0, ..., 0)$, there is exactly one child node where the corresponding 0 is changed to 1. Then, for every 0 preceding the leftmost 1 in a node, there is also exactly one child node where the corresponding 0 is changed to 1. After constructing all children

nodes, the tree contains exactly 2^n nodes (i.e., one node for each state). This tree has the following property: If a node represents a state satisfying $\mathcal{T}_{det}$, then its parent node also represents a state satisfying $\mathcal{T}_{det}$. Since in $\mathcal{ALH}^-$ the state $(0, ..., 0)$ satisfies $\mathcal{T}_{det}$, for each node it holds that if it satisfies $\mathcal{T}_{det}$ then there is a path from this node to $(0, ..., 0)$ satisfying $\mathcal{T}_{det}$. □

We will now specify a Gibbs sampling algorithm with deterministic dependencies in $\mathcal{ALH}^-$. Instead of answering conditional probability queries $\mathbf{P}(X \mid \boldsymbol{e})$, with this algorithm it is possible to answer probability queries $\mathbf{P}(X)$ conditioned on the ground clauses obtained from $\mathcal{T}_{det}$.

Our objective is to exploit the fact that – due to restrictions in $\mathcal{T}_{det}$ – there are a lot of state transition possibilities where the bit-flip probability (the probability that a random variable will change its value) is 0, i.e., a lot of cases in which one does not need to sample at all. We propose to assign an integer γ_i to each random variable X_i. γ_i is initially 0 and is increased whenever a bit-flip occurs for a variable $X_j, i \neq j$ that restricts the bit-flip probability of X_i to be 0 ("set"), and γ_i is decreased whenever this specific restriction does not hold any more ("release"). As long as $\gamma_i > 0$, X_i is not sampled and we say that X_i is "blocked". Settings $(+1)$ and releases (-1) are depicted in Table 2 for groundings of all $\mathcal{ALH}^-$-axioms with individuals i, j. Consider e.g. the first row in the second column of Table 2: If the ground atom $A(i)$ is 0 in $\boldsymbol{x}_t$ and is 1 in $\boldsymbol{x}_{t+1}$ then $B(i)$ is known to be 1 (otherwise $\boldsymbol{x}_{t+1}$ would violate this constraint) and $\gamma_{B(i)}$

Table 2. Setting and releasing blockings due to $\mathcal{ALH}^-$-axioms

$A \sqsubseteq B$

$A(i)_t$	$A(i)_{t+1}$	$\gamma_{B(i)}$
0	1	+1
1	0	−1

$B(i)_t$	$B(i)_{t+1}$	$\gamma_{A(i)}$
0	1	−1
1	0	+1

$A \sqsubseteq \neg B$

$A(i)_t$	$A(i)_{t+1}$	$\gamma_{B(i)}$
0	1	+1
1	0	−1

$B(i)_t$	$B(i)_{t+1}$	$\gamma_{A(i)}$
0	1	+1
1	0	−1

$R \sqsubseteq S$

$R(i,j)_t$	$R(i,j)_{t+1}$	$\gamma_{S(i,j)}$
0	1	+1
1	0	−1

$S(i,j)_t$	$S(i,j)_{t+1}$	$\gamma_{R(i,j)}$
0	1	−1
1	0	+1

$\exists R.\top \sqsubseteq A$

$R(i,j)_t$	$R(i,j)_{t+1}$	$\gamma_{A(i)}$
0	1	+1
1	0	−1

$A(i)_t$	$A(i)_{t+1}$	$\gamma_{R(i,j^*)}$
0	1	−1
1	0	+1

$\top \sqsubseteq \forall R.A$

$R(i,j)_t$	$R(i,j)_{t+1}$	$\gamma_{A(j)}$
0	1	+1
1	0	−1

$A(j)_t$	$A(j)_{t+1}$	$\gamma_{R(i^*,j)}$
0	1	−1
1	0	+1

is increased, i.e., $B(i)$ is blocked by $A(i)$. Individuals with * indicate that every individual of the closed domain has to be considered separately.

The Markov chain has to start with a state $\boldsymbol{x}_1$ satisfying $\mathcal{T}_{det}$. Such a state can be found with MaxWalkSat [9], a local search algorithm for the weighted satisfiability problem. All deterministic clauses are assigned with a weight greater than the sum l of all weights of clauses obtained from $\mathcal{T}_w \cup \mathcal{A}_w$. Then, a state violating clauses of total weight l or less satisfies $\mathcal{T}_{det}$ (cf. [9]). The state $(0, ...0)$ is known to satisfy $\mathcal{T}_{det}$ such that the chain could also start from this node, but MaxWalkSat will find a node that will be (near to) the mode of the distribution such that the chain will approximate faster. Since all transitions to states not satisfying $\mathcal{T}_{det}$ have probability 0, the whole chain does only involve states satisfying $\mathcal{T}_{det}$. In other words, it is guaranteed that $P(mb(X_i)) > 0$ such that the transition distribution $\mathbf{P}(X_i \mid mb(X_i))$ is defined. Given KB_M, this distribution is computed with (1) where functions f_i are defined according to (3).

The Gibbs-$\mathcal{ALCH}^-$-algorithm is depicted in Fig. 2. If the value of X_i is flipped, the method *setRelease* is called with parameters X_i, the old and new value of X_i, x_i resp. x_i', and the set Cl_i of deterministic ground clauses containing X_i. This method sets and releases blockings as specified in Table 2. At the beginning of the process, it has to be ensured that all initial blockings are set. This is done by calling *set*, a function similar to *setRelease*, but only increasing γ_i. Then, the settings and releases depicted in Table 2 ensure that for each X_i, $\gamma_i > 0$ if and only if either $P(x_i \mid mb(X_i)) = 0$ or $P(\neg x_i \mid mb(X_i)) = 0$. Finally, after $n \cdot N$ samples, the Boolean vector $\mathbf{N}[X]$ of counts over X is normalized by α and the result represents the estimated distribution of $\mathbf{P}(X)$. The method can further be optimized by restricting $\boldsymbol{X}$ to a set of ground atoms assumed to be relevant for the query answer.

Algorithm Gibbs-$\mathcal{ALCH}^-(KB_M, X, N)$
Output An estimate of $\mathbf{P}(X)$
  local variables:
    $\mathbf{N}[X]$, a vector of counts over X
    Cl, a set of all (weighted) ground clauses obtained from KB_M
    $\boldsymbol{X} = \{X_1, ..., X_n\}$, the set of all random variables
    $\boldsymbol{\gamma} = \{\gamma_1, ..., \gamma_n\}$, a set of integers initially assigned with 0
    $\boldsymbol{x} = (x_1, ..., x_n)$, an initial state satisfying $\mathcal{T}_{det}$
  for i = 1 to n **do if** $(\gamma_i = 0)$ set(X_i, $\neg x_i$, x_i, Cl_i);
  for j = 1 to N **do**
    for i = 1 to n **do**
      if $(\gamma_i = 0)$
        sample the value x_i' of X_i in $\boldsymbol{x}$ from $\mathbf{P}(X_i \mid mb(X_i))$;
        if $(x_i \neq x_i')$ setRelease(X_i, x_i, x_i', Cl_i);
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in $\boldsymbol{x}$
  return α <$\mathbf{N}[1], \mathbf{N}[0]$>

Fig. 2. The Gibbs-$\mathcal{ALCH}^-$ algorithm

6 Conclusion

We have shown that deterministic dependencies in $\mathcal{ALH}^-$ retain the regularity of Markov chains constructed with Gibbs sampling. Further, we proposed a method incorporating these dependencies. Since lots of redundant samples are not generated by this method, we conclude that significant efficiency is gained. This is in contrast to previous results, where Gibbs sampling with (near-)determinism is known to give poor results. While $\mathcal{T}_{det}$ is specified with $\mathcal{ALH}^-$, the sets $\mathcal{T}_w$ and $\mathcal{A}_w$ of weighted axioms resp. weighted assertions can e.g. be specified in the expressive DL $\mathcal{SHQ}$. In [7] it is shown that Gibbs sampling slows down more and more when clauses are assigned with weights beyond 4. Note that a world not satisfying a ground clause with weight 4 is as probable as a world not satisifying e^3 ground clauses with weight 1. Thus, for ground clauses not intended to represent deterministic knowledge, a weight around 4 usually will suit the requirements of a knowledge engineer. However, often, in addition to axioms in $\mathcal{ALH}^-$, there is also the need to represent deterministic assertions in $\mathcal{A}_{det}$ as well as deterministic functional- and transitive roles. Further research is required, in order to also incorporate these dependencies. Another open task is the implementation of the proposed method in order to compare its runtime performance with other approaches estimating probability distributions under consideration of (near-)deterministic knowledge (such as MC-SAT [7] or SampleSearch [8]).

References

1. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Mateo, CA, 1988.
2. Koller, D., Levy, A., Pfeffer, A., Getoor, L., Taskar, B.: Graphical Models in a Nutshell. In: Introduction to Statistical Relational Learning, pages 13–55, Cambridge, MA: MIT Press, 2007.
3. Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, January 2003.
4. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach (Second Edition) Prentice Hall, 2003.
5. Geman, S., Geman, D.: Stochastic Relaxation, Gibbs Distribution and Bayesian Restoration of Images. IEE Transactions on Pattern Analysis and Machine Intelligence 6, pages 721–741, 1984.
6. Domingos, P., Richardson, M.: Markov Logic: A Unifying Framework for Statistical Relational Learning. In: Introduction to Statistical Relational Learning, pages 339–371, Cambridge, MA: MIT Press, 2007.
7. Poon, H., Domingos, P.: Sound and Efficient Inference with Probabilistic and Deterministic Dependencies. In Proceedings of AAAI-06, Boston, Massachusetts, July 2006.
8. Gogate, V., Dechter, R.: SampleSearch: Importance Sampling in presence of Determinism. ICS technical report (Submitted), July 2009.
9. Jiang, Y., Kautz, H., Selman, B.: Solving Problems with Hard and Soft Constraints using a Stochastic Algorithm for MAX-SAT. In First International Joint Workshop on Artificial Intelligence and Operations Research, 1995.